А.С.Подколзин

Компьютерное моделирование логических процессов

Архитектура и языки решателя задач

2007 г.

Введение

Прогресс в развитии вычислительной техники неоднократно приводил к активизации исследований по искусственному интеллекту. Наиболее важными рубежами здесь являются период первого появления электронных вычислительных машин и период распространения персональных компьютеров. Вычислительная техника стала широко доступной, а программирование приобрело массовый характер. Несмотря на это, искусственный интеллект так и не появился, а первоначальный оптимизм в отношении перспектив его создания стал постепенно вытесняться пессимизмом. Успехи и неудачи в программировании определяются на сегодняшний день границей, отделяющей те области, в которых имеются эффективные алгоритмы для решения задач, от областей, для которых эти алгоритмы не созданы. К числу последних относятся, к сожалению, не только творческая деятельность, но даже такие "простейшие" способности человека, как понимание естественного языка и изображений. Успехи здесь пока достаточно скромны.

Вместе с тем, отсутствие простого общего алгоритма решения задач в какой-либо из трудных для программирования областей, например, школьной геометрии, не является препятствием для решения таких задач человеком. Хотя общего алгоритма в учебниках и нет, но в каждом отдельном случае логика принятия решения вполне поддается анализу и объяснению. Обучаясь на сотнях и тысячах примеров, постоянно пополняя и корректируя свои умения, человек постепенно аккумулирует в себе необходимый ему алгоритм, нигде не изложенный в явном виде. Чтобы получить аналогичный алгоритм в компьютере, остается только следовать этому примеру. Видимо, это и есть реальный способ создавать программы "там, где нет алгоритмов". Если не овладеть таким искусством, то никакое, даже самое впечатляющее, развитие вычислительной техники сколь-нибудь существенным образом положения дел с искусственным интеллектом не улучшит.

Алгоритмы, извлекаемые человеком из процесса обучения на примерах, сильно отличаются от алгоритмов обычного программирования. Прежде всего, они почти лишены циклов - на каждом шаге для решения задачи привлекаются какие-то новые знания и приемы, зачастую из совершенно различных областей. Алгоритмы эти плохо декомпозируются на блоки. Скорее, они представляют собой что-то типа векторного поля приемов, в котором перемещается решаемая задача и которое создается путем длительной трудоемкой дрессировки. Срабатывание каждого приема продвигает задачу на один шаг вперед по ее траектории. До тех пор, пока не получено достаточно полной коллекции приемов, обеспечивающей с большой вероятностью "цепную реакцию" процесса решения задачи, эти алгоритмы практически бесполезны. А так как обычно критический минимум достаточно велик и насчитывает многие тысячи элементов, то возникает серьезный психологический барьер - у программиста после проработки первых сотен задач может возникнуть впечатление, что проект безнадежен и его следует бросить.

Еще одно отличие алгоритмов рассуждений от алгоритмов вычислений связано со способом кодирования информации. Вычислительная программа обычно использует технические структуры данных, семантика которых остается за кадром, так как непосредственно при вычислениях не нужна. Приемы интеллектуальной системы анализируют задачу независимо друг от друга, и чтобы они могли понимать текущий контекст, необходимо сохранить всю его семантику, используя для представления данных универсальный логический язык. Решение трудных для алгоритмизации задач - это почти всегда процесс преобразования логических структур данных или, иными словами, логический процесс.

История развития математики тесно связана с изучением процессов различных типов. Многие разделы классической "непрерывной" математики, и в первую очередь теория дифференциальных уравнений, возникли для изучения физических процессов. Изучение процессов обработки дискретной информации в электронике привело к появлению теории автоматов и других разделов математической кибернетики. К сожалению, логические процессы оказались настолько сложны, что попытки изучения их математическими методами так и не привели к созданию эффективных решателей в сколь-нибудь нетривиальных областях. Поэтому единственным подходом к созданию науки о логических процессах на сегодняшний день остается компьютерное моделирование, т.е. указанная выше дрессировка "логических векторных полей" на потоках обучающих примеров.

Создание больших решателей вручную чрезвычайно трудоемко, так как требует постоянной регулировки взаимодействий в многотысячной системе активных элементов. Такие решатели, как правило, изначально не очень качественные и требуют последующей, еще более трудоемкой оптимизации. Однако, они дают материал, позволяющий начать исследования по автоматизации синтеза приемов и саморазвитию "логических векторных полей", без которого невозможно было бы создание подлинно интеллектуальной системы. Объектом изучения здесь являются процессы синтеза алгоритмов по теоремам и процессы автоматического развития теорий, ориентированного на синтез алгоритмов. Это - еще одна разновидность логических процессов.

Данная монография описывает многолетний опыт компьютерного моделирования логических процессов, в результате которого возникла развитая технология обучения решателей. Обучение компьютерной системы предпринималось в различных областях математики и в элементарной физике. Моделировались также логические процессы, возникающие при анализе текстов естественного языка, анализе изображений и принятии решений в игровых ситуациях. Всего было проработано около 9 тысяч задач и создано около 25 тысяч приемов. Поведение системы в тех областях, где был накоплен необходимый критический минимум приемов, выглядит вполне целесообразным и доказывает правильность предлагаемого подхода. Ход рассуждений демонстрируется по шагам, причем во многих разделах уровень сложности решаемых задач достаточно серьезен.

Материал монографии разбит на три книги. Данная, первая книга посвящена общему описанию архитектуры компьютерной системы и языкам программирования, созданным для ускорения процесса обучения. Эти языки, ЛОС (Логический Описатель Ситуаций) и ГЕНОЛО́Г (ГЕНетический язык ЛОГического программирования), подняли обучение решателей практически до уровня формулировки математических теорем. Первый из них - чуть выше уровня ПРОЛОГ'а, второй - относится к первому примерно так же, как ПРОЛОГ к ассемблеру. Планируется выпуск второй и третьей книг. Вторая книга будет посвящена описанию собственно решателей; од-

новременно она может рассматриваться как курс программирования на ГЕНОЛОГе. Наконец, третья книга включит в себя материал, связанный с вопросами автоматического развития решателей и автоматизации теоретических исследований.

К книге прилагается диск, на котором находится текущая версия системы. Для установки системы на компьютере нужно создать новую папку, скопировать в нее содержимое папки LOGSYST с диска, и вынести ярлык "ЛСИ" на рабочий стол. Содержимое первой книги позволяет научиться пользоваться решателем в тех разделах, где он достаточно "укомплектован" приемами.

Однако, система никоим образом не претендует на роль коммерческой программы. Скорее, она представляет собой попытку создания среды программирования для обучения интеллектуальной системы, интегрирующей в себе знания из множества различных областей, способной к проведению самостоятельных исследований и саморазвитию. Архитектура системы допускает увеличение ее сегодняшних размеров (около 60 Мб) в тысячи раз без сколь-нибудь значимого замедления для уже освоенных разделов. Совместная работа над развитием универсального интеллектуального комплекса представляется единственно возможным способом объединения усилий многих исследователей в создании науки о логических процессах. Только таким образом можно обеспечить появление единой логической формализации понятий и добиться разумного взаимодействия приемов в рамках общего "логического векторного поля". Получение широкого спектра основанных на единой технологии решателей является необходимой предпосылкой для проведения исследований по автоматическому их синтезу.

Обучение логической системы требует постоянного развития программ интерфейсов и компилятора. Вторая половина книги (начиная с главы 13) содержит достаточно подробное описание этих программ. Она будет необходима тем, кто захочет проводить самостоятельные исследования по моделированию логических процессов. При чтении этой части книги нужно иметь перед собой соответствующий раздел программы, отображаемый на экране компьютера. Представленный здесь материал имеет чисто справочный характер, и для начала работы вполне достаточно извлечь из него лишь общее представление об архитектуре программ.

Проработав сравнительно полно ряд разделов элементарной математики, автор взялся за развитие решателей в целом ряде новых разделов, не собираясь своими силами доводить их до конца, а лишь с целью преодолеть стартовые трудности и предоставить тем, кто захотел бы продолжить исследование логических процессов, возможно более благоприятные исходные позиции.

Как и всякая большая программа, к тому же созданная единственным программистом, решатель, вероятно, имеет множество недостатков и ошибок. Автор заранее приносит за них свои извинения. Однако, будучи первой попыткой в развитии новых технологий, этот решатель, все же, заслуживает определенного снисхождения.

В заключение автор выражает искреннюю благодарность В.Б.Кудрявцеву, поддержка которого сделала возможным проведение данного исследования.

Оглавление

Глава 1

Общие замечания о логических процессах и их моделировании

Логические процессы играют исключительно важную роль в поведении любой интеллектуальной системы. Они необходимы для распознавания зрительных либо звуковых образов, для управления динамикой системы, для понимания языка, планирования действий, анализа потока событий, решения технических и теоретических задач. Хотя основной объем обработки информации во многих случаях берут на себя средства автоматики, использующие не логические, а "технические" структуры данных, логика играет роль диспетчера, организующего и контролирующего работу этой автоматики, а в необходимых случаях обеспечивает ее развитие. В тех случаях, когда для решения задачи имеется заранее выработанный план действий, обычно логический вывод не требуется - вместо него создается специализированный вычислительный алгоритм. Это обычный и хорошо известный путь развития программирования, на котором создаются различные системы численного моделирования, компиляции и оптимизации технических проектов, управления сложными системами и т.п. Потребность в логических процессах появляется лишь при отсутствии известного заранее плана действий. Тогда планирование действий происходит непосредственно во время решения задачи, причем для обеспечения возможности нахождения приемлемых вариантов система должна располагать достаточно обширным и разнообразным запасом знаний. Использование того или иного элемента этого запаса знаний заранее непредсказуемо - задача из одной предметной области может потребовать для своего решения весьма удаленных от этой области идей. По этой причине, вряд ли возможно создавать подлинно интеллектуальные системы для ограниченных классов задач - например, только для распознавания изображений, или только для понимания естественного языка, или только для написания программ, синтеза "чип"ов, и т.д. Чтобы достаточно эффективно действовать хотя бы в одной из таких областей, необходимо заложить в интеллектуальную систему универсальное "фундаментальное" образование, как это и происходит обычно при обучении человека. Уже один только количественный аспект этой не решенной на сегодняшний день задачи далеко выводит ее за рамки трудностей традиционного программирования. Заметим, что даже в случае обучения человека, с отлаженной и оптимизированной веками системой образования, процесс подготовки полноценного специалиста (с учетом весьма существенного дошкольного периода) занимает более 20 лет.

Попытки исследования логических процессов математическими методами привели к появлению и развитию такой науки, как математическая логика. В ее рамках удалось дать точные определения фундаментальным понятиям логического язы-

ка, аксиоматической теории, формального доказательства, алгоритма решения задачи, сложности вычислительной процедуры. Был накоплен значительный запас математических результатов, характеризующих общие свойства этих понятий. Возникли новые представления о математической строгости и формализации теорий, которые привели к переизложению многих разделов математики в виде формальных дедуктивных систем. Изучение общих свойств этих формальных теорий методами математической логики позволило решить ряд важных общематематических проблем: доказать независимость континуум - гипотезы; установить алгоритмическую неразрешимость диофантовых уравнений и др. Однако, основные продвижения математической логики оказались связаны не столько с логической "динамикой", сколько с логической "статикой". Несмотря на интенсивные исследования в области процедур автоматического доказательства теорем, не удалось не только предложить эффективно работающие общие процедуры решения задач, но даже найти общие принципы, на основе которых такие процедуры могли бы быть созданы. Основной проблемой, с которой здесь пришлось столкнуться, явилась проблема трудоемкости поиска решения задачи по деревьям прямого либо обратного логического вывода, экспоненциально возрастающей с ростом глубины поиска. Всевозможные результаты по отсечению ветвей рассматриваемого дерева для уменьшения объема перебора и различные эвристические общие принципы упорядочения перебора не изменили ситуации сколь-нибудь ощутимым образом.

В сложившейся ситуации, когда поиски математических методов, позволяющих создавать эффективные решатели задач, фактически зашли в тупик, единственной возможностью создания таких решателей становится компьютерное моделирование логики рассуждений человека. Компьютерная модель здесь должна сыграть роль "микроскопа" для изучения процессов поиска решения и самообучения, с помощью которого были бы выявлены фундаментальные принципы их организации. Возможно, что по мере развития такой "экспериментальной" теории логических процессов вновь окажутся востребованы и математические методы исследования.

Компьютерное моделирование логических процессов естественно начинать с моделирования автоматики, управляющей ходом рассуждений, оставляя пока в стороне значительно более сложные процессы саморазвития этой автоматики. Чтобы разрабатывать механизмы самообучения и самопроектирования, необходимо прежде всего получить определенный запас созданных вручную "эталонных" алгоритмических конструкций, моделирующих процессы решения задач. По-видимому, без таких "образцов для подражания" трудно определить необходимый для эффективного функционирования запас архитектурных стандартов, который могла бы использовать процедура самопрограммирования системы. Многократно предпринимавшиеся ранее попытки создания саморазвивающихся интеллектуальных систем, основанных на тех или иных простых общих принципах, без анализа реальной логической автоматики во всей ее сложности и разнообразии, оказывались вполне бесплодными.

В самых общих чертах, процесс расшифровки логической автоматики, обеспечивающей решение задач в некоторой предметной области, можно представить происходящим по следующей схеме. Рассматривается некоторая обучающая последовательность задач в этой области, решение которых известно эксперту, осуществляющему развитие компьютерной модели. Предпринимается анализ траектории решения очередной задачи из выбранной последовательности. Если эта задача не представляет собой простейший тест, для которого имеется заранее известный стандартный план действий, то процесс ее решения складывается из отдельных этапов, состоящих из локального планирования и реализации выработанного плана действий. Каждый такой

этап локального планирования анализируется, и для него предлагается некоторый гипотетический алгоритм, приводящий в рассматриваемой ситуации к тому же результату. Эти алгоритмы накапливаются в компьютерной модели, так что в новой задаче некоторые из них могут сработать и предложить свою версию развития решения. Каждое такое срабатывание анализируется на предмет целесообразности, и в решающие правила алгоритма локального планирования вводятся необходимые коррективы. Эта процедура повторяется для многих сотен задач предметной области - до тех пор, пока сложившаяся мозаика алгоритмов локального планирования действий не обнаружит достаточно устойчивого и эффективного поведения. В целом, процесс напоминает обучение нейросистемы методом "поощрений и наказаний"; отличие состоит лишь в более сложной процедуре коррекции, которая не сводится к изменению каких-либо числовых параметров, как у нейрокомпьютера, а требует привлечения логики для очередного перепроектирования алгоритмов планирования.

Многообразие алгоритмов локального планирования создает своего рода "логическое векторное поле", в котором перемещается решаемая задача - от исходной формулировки к ответу. Разумеется, это лишь упрощенная аналогия; отдельные шаги в таком перемещении могут представлять собой целые иерархии вложенных логических процессов, с различными циклами ограниченного поиска и отбора. Однако, она отражает главное обстоятельство: по завершении реализации каждого локального плана предпринимается независимое рассмотрение всей текущей ситуации заново, для выработки следующего плана. К сожалению, на сегодняшний день не известно другого способа создания эффективных "логических векторных полей" автоматического решения задач, кроме весьма трудоемкого процесса их дрессировки на тысячах обучающих примеров.

Так как алгоритмы локального планирования (для краткости, далее условимся называть такие алгоритмы приемами решения задач) берут всю инициативу на себя, не имея какого-либо стоящего над ними управляющего центра (ведь никто, кроме конкретного приема, и не располагает достаточной для принятия решения в области его компетенции информацией), то особо важную роль приобретает точный учет взаимодействий различных приемов в цепи решения задачи. Трудоемкость этого учета, в среднем, должна быть пропорциональна квадрату числа приемов, по крайней мере в той предметной области, где проводится обучение. Разумеется, адекватно большим должен быть и обучающий материал - иначе качество работы решателя будет весьма посредственным. По-видимому, единственной возможностью преодоления этого недостатка логической автоматики (как и вообще любой автоматики, ограниченной определенными разумными рамками) является развитие средств ее автоматического пополнения и оптимизациии, которые подключались бы при решении задачи по мере надобности. Анализ многих и многих математических задач так называемого "нестандартного характера" показывает, что выход за рамки накопленных запасов приемов (обычно связанный с попыткой скомбинировать несколько известных теорем и извлечь из них новую идею) является, в общем, обыденным явлением и даже по своей трудоемкости иногда вполне сопоставим с работой автоматики стандартных приемов. Впрочем, автоматика мышления в поведении любых интеллектуальных систем играет огромную роль, и лишь в ситуациях "аварийного" типа включаются дополняющие ее средства более высоких уровней.

Описанная выше общая схема моделирования логической автоматики была реализована на примере процессов решения математических задач. Причины, по которым на первом этапе была рассмотрена математика, вполне понятны: именно в ней при решении задач возникают наиболее сложные и разнообразные логические процессы,

причем имеются огромные запасы обучающего материала. В последнее время начато моделирование логических процессов и в ряде нематематических областей. Было проработано свыше 9 тысяч задач из таких разделов, как элементарные алгебра и геометрия, математический анализ, аналитическая геометрия, линейная алгебра, дифференциальные уравнения, теория вероятностей, комплексный анализ, элементарная физика, школьные текстовые задачи (с синтаксическим и семантическим анализом текста). На основе этого обучающего материала сформирована база приемов решателя задач, насчитывающая в настоящее время более 25 тысяч приемов. Фактически, возникла мощная система символьной компьютерной математики нового типа, позволяющая не только получать ответы, но и прослеживать ход решения по шагам, а при необходимости вмешиваться в процесс решения. В отличие от традиционных систем компьютерной математики, где накапливались тысячи функций для пользователя, который должен был вручную находить их в каталоге системы и применять к текущей задаче, новая система накапливает тысячи приемов, которые самостоятельно анализируют задачу и принимают решение о выполнении преобразований. Пользователь здесь может вообще не интересоваться содержимым базы приемов. Разумеется, это более высокая степень автоматизации, и как следствие нового подхода, системе становятся доступны значительно более сложные задачи. Преимущества нового подхода становятся особенно заметны в тех разделах, где велика роль логического вывода. В элементарной алгебре это решение уравнений и неравенств с параметрами; в математическом анализе - качественное исследование функций с помощью пределов и производных, исследование сходимости рядов, вычисление кратных интегралов и применение их для нахождения площадей и объемов. Разумеется, наиболее интересными с точки зрения моделирования логических процессов оказались элементарная и аналитическая геометрии. В распространенных системах компьютерной математики эти разделы представлены совсем слабо, причем элементарная геометрия по существу отсутствует, а аналитическая - связана лишь с выполнением стандартных операций явно одношагового характера.

Работа над моделированием процессов решения математических задач потребовала создания целого технологического комплекса, позволившего существенно приблизить язык для программирования приемов к языку для формулировки теорем предметной области. Фактически, прием задается как теорема, снабженная системой указателей для компилятора, генерирующего на основе этой теоремы программу приема. Задачей компилятора является синтез такой программы, которая была бы способна усматривать возможность применения теоремы даже в неявных, "замаскированных" ситуациях, оценивать целесообразность такого применения и осуществлять его. Программирование приемов имеет рекурсивный характер: в описании приема допускаются обращения к всевозможным вспомогательным процедурам и задачам, применяемым к фрагментам теоремы. Это означает, что при реализации приема может быть использована не только та теорема, на которой он основан, но и сотни других теорем, востребованных его вспомогательными операторами. Использование стандартной математической записи для изображения теоремы приема, а также сопровождение геометрических теорем чертежами позволили приблизить наглядность описания приема к наглядности текста решаемой задачи. Сочетание этой наглядности с развитием системы семантической трассировки процесса решения, выдающей на экран подробные объяснения выполняемых преобразований, значительно уменьшило трудоемкость процесса анализа решений и коррекции приемов. Вместе с тем, новый язык для записи приемов не просто создает определенные преимущества для процесса обучения решателя. В действительности, его приближенность к логиче-

скому языку предметной области ориентирована в первую очередь на последующие исследования в области автоматического синтеза приемов; на весь круг проблем, относящихся к пограничному слою между логикой и алгоритмами.

Фактически, развитие техники обучения сделало доступным продолжение обучения системы ее пользователям - так же, как и в случае обычных систем компьютерной математики, которые предоставляют пользователю средства для программирования новых функций. Отличие здесь заключается в том, что вместо мало приспособленных для логических процессов традиционных средств процедурного программирования развит мощный комплекс программирования продукционного. Чтобы сравнить уровень используемого языка с уровнем известного языка логического программирования "ПРОЛОГ", заметим лишь, что программы на промежуточном языке уровня "ПРОЛОГ"а возникают как результат компиляции описания приема; естественно, они значительно больше и, в сравнении с первоначальным описанием, совершенно "нечитабельны".

Собственно работе по моделированию логических процессов в математике предшествовало решение ряда важных технических проблем, определившее общую архитектуру компьютерной системы. Прежде всего, необходимо было выбрать логический язык для представления задач. Разумеется, для адекватного моделирования процессов решения такой язык должен быть возможно более приближен к "естественному" математическому языку. Языки математической логики, создававшиеся для теоретического исследования свойств аксиоматических теорий, обычно вводились как можно менее избыточным образом - для упрощения связанных с с ними формулировок теорем и доказательств этих теорем. Использование таких языков в решателе привело бы к неоправданному усложнению формулировок и повлекло бы существенные расхождения между моделью и моделируемым процессом. Фактически, оказалось, что несмотря на огромное количество работ в математической логике, посвященных формализации различных разделов математики, никакой практически пригодной формализации "сквозного" характера, сохраняющей все введенные для удобства логических вычислений в обычной математической практике условности, создано не было. Поэтому для решателя пришлось создавать свой собственный логический язык, пополняемый по мере проработки новых предметных областей. Одним из важных наблюдений, которые здесь были сделаны, оказался определенный конфликт между требованиями к языку, предъявляемыми к нему традиционными для формальной логики соображениями логической аккуратности с одной стороны, и соображениями вычислительной целесообразности - с другой. Аккуратная логическая запись иногда становится избыточно громоздкой, и в определенных контекстах ее можно заменить упрощенными суррогатами, достаточными для корректных вычислений. Возникло, таким образом, явление "контекстной семантики" языка, когда смысл того или иного утверждения задачи определяется лишь в контексте всей задачи в целом. Простейшим примером такого рода является обычное использование в математическом анализе утверждений вида $f(x) = O(g(x))$, для понимания смысла которых в контексте должно содержаться что-нибудь вроде $x \to 0$.

Учет контекста при обработке логических структур данных позволяет вводить в язык различного рода нечеткие понятия, что часто бывает необходимо в нематематических областях. Для уточнения их смысла в конкретной ситуации можно обращаться к сопровождающей информации - как логической, так и технической (например, анализировать изображения, протоколы и т.п.). Разумеется, какие-либо формальные аксиоматические теории для таких понятий будут отсутствовать, однако в процессе "дрессировки" системы на обучающем материале можно выработать достаточно

хорошие алгоритмические правила правдоподобных рассуждений, аппроксимирующие действия эксперта. Таким образом, созданный аппарат обучения баз приемов никоим образом не ограничен рамками хорошо формализованных предметных областей. Начатый при рассмотрении математических задач процесс создания некоторого всеобъемлющего логического языка, используемого решателем, предполагает продолжение и на нематематические области, включающие нечеткие понятия и нечеткие рассуждения. Для работы с нечеткой логикой, по-видимому, потребуются адаптивные алгоритмические конструкции, компенсирующие нечеткость понятий аккуратным учетом контекста и накапливающие "четкие" аппроксимации нечетких понятий в виде сложных логических описаний образов, стоящих за такими понятиями. Заметим, наконец, что работа над указанной выше широкой логической формализацией, с развитием многообразия сопутствующих приемов, совершенно необходима для создания системы, способной адекватно понимать естественные языки.

Чтобы при обучении решателя легко распознавались отклонения его реального поведения от желаемого, необходима хорошая визуализация процесса решения. В частности, при отображении на экране математических утверждений, сформулированных на внутреннем логическом языке решателя, необходим перевод этих утверждений на язык общепринятой математической символики. Обратный перевод также является весьма существенным, так как ускоряет ввод нового обучающего материала и новых приемов. В системе реализован редактор математических формул, обеспечивающий быстрый ввод математических утверждений и выражений, а также осуществляющий перевод во внутреннее логическое представление и обратно. Размеры элементов вводимой формулы определяются и корректируются в процессе ввода автоматически; широко используется ввод длинного термина с помощью нескольких нажатий клавиш (в большинстве случаев - первые две-три буквы термина). Это позволяет набирать даже весьма громоздкие на вид выражения с помощью сравнительно малого числа нажатий клавиатуры. Так как для подавляющего большинства рассматриваемых в неэлементарной математике понятий не предусмотрена специальная символика, пришлось вводить в формульном редакторе словесное их представление (как правило, такое понятие F используется в скобочной записи вида $F(t_1, \cdots, t_n)$). Логика действий формульного редактора достаточно сложна, и хотя он реализован в духе обычного процедурного программирования, но фактически сам является небольшим решателем. По-видимому, для увеличения динамики обмена информацией с системой было бы полезно более явным образом ввести логические процессы в процедуры визуализации и перевода формульно-графических данных на внутренний язык, обучая соответствующие редакторы примерно так же, как обучаются решатели задач.

Следующий вопрос после выбора логического языка - формализация понятия задачи, достаточная для работы с обучающим материалом. На начальном этапе исследований по искусственному интеллекту предпринимались многочисленные попытки дать математически строгие определения понятия задачи, решения задачи и ответа на задачу. В самом общем виде, задача представлялась как логическое условие $P(x)$ на элементы x некоторого универсального множества объектов U; ответ на задачу - как утверждение логического языка, определяющее в некоторых "явных"терминах конкретный элемент x множества U, для которого $P(x)$ истинно; решение задачи - как цепочка преобразований исходного условия, преобразующая его к виду ответа. Вводя в универсум U классы объектов, в таком виде легко представить не только задачу поиска единственного элемента некоторого множества, удовлетворяющего заданному условию, но и задачу описания всех таких элементов. Однако, это

простое и, казалось бы, весьма общее представление о задаче сразу же сталкивается с трудностями при сопоставлении его даже с простейшими реальными задачами по элементарной алгебре. Так, при решении задачи на упрощение алгебраического выражения вовсе не накладывается какого - либо строгого ограничения $P(x)$ на предъявляемый школьником ответ x - например, не требуется предъявления доказательства того, что этот ответ минимален по числу символов или по какому-либо другому функционалу сложности выражения. Все, что в таких случаях требуется - это применять определенный запас стандартных приемов упрощения до тех пор, пока дальнейшие попытки не будут давать каких-либо улучшений, и выдать полученное выражение в качестве ответа. Этот и другие примеры приводят к выводу, что более адекватным является представление о задаче, как о сочетании некоторого строгого условия на допустимость ответа с рядом целевых установок на его оптимизацию, учитываемых при решении по мере возможности. Тогда понятие ответа становится "субъективным", зависящим от уровня обученности системы. Впрочем, при аккуратной формализации целевых функционалов появляется возможность некоторой "объективизации", заключающейся в последовательном обучении или самообучении для повышения качества ответов.

Даже в хорошо формализованных областях, реально используемое понятие задачи является в значительной степени нечетким. Целевая установка на проведение исследования в математике, или задание на разработку проекта в технике, как правило, требуют существенного доопределения, опирающегося на всю сумму знаний из рассматриваемой области. Значительную часть таких знаний составляют различного рода условности и эвристические приемы, необходимые для адекватной расшифровки и уточнения задания уже в процессе его реализации. Поэтому при разработке архитектуры решателя следует в первую очередь четко определить лишь структуры данных, которые будут определять "текущее состояние" задачи в процессе ее решения, а допустимые шаги преобразования задачи и способ усмотрения ответа на нее, достигаемого в конце цепочки таких преобразований, уточнять по мере проработки конкретного обучающего материала.

Применительно к логическим процессам, понятие задачи должно быть определенным образом ограничено. Если речь идет о разработке технической системы, написании программы, составлении плана действий и т.п., то синтезируемый объект обычно задается не на логическом, а на специализированном техническом языке. Такое задание часто имеет иерархический характер, причем на каждом уровне структуры данных приспособлены для быстрых вычислений, необходимых при компиляции и оптимизации проекта. Собственно логические процессы возникают тогда, когда заранее созданных средств вычислительной автоматики оказывается недостаточно. В этом случае из всего контекста технической информации извлекаются данные, необходимые для "локального" логического рассмотрения , формулируется задача, и после ее решения вносятся необходимые коррективы в технические структуры данных. Роль решателя задач, реализующего логические процессы, здесь можно уподобить роли головки машины Тьюринга, выполняющей вычисления на ленте - она сугубо локальна. С этой точки зрения, процессы проектирования или исследования представляют собой макрозадачи - обращения к решению задач (в указанном выше узком смысле применения логического процесса для получения ответа) в них являются лишь отдельными атомарными шагами и тесно переплетаются с нелогическими процессами. Вообще, взаимодействие логических и нелогических процессов в интеллектуальных системах происходит практически постоянно. Для объектов, упоминаемых в логических структурах данных, создаются модели (мысленные или иные). Наблюдение за

структурой и функционированием моделей подсказывает гипотезы, направляющие ход рассуждений; при проектировании - позволяет поставить новые задачи, после решения которых предпринимается коррекция модели, и т.п. Важной проблемой, связанной с использованием нелогических моделей объектов и процессов, оказывается проблема перевода результатов наблюдения за моделью на логический язык. Частными случаями этой проблемы являются распознавание изображений и анализ динамических образов.

Предварительная классификация "логических типов" задач, возникшая при анализе математических предметных областей, привела к рассмотрению 4 основных типов задач: на доказательство, на преобразование, на описание, и на исследование. Любая такая задача имеет, прежде всего, некоторый (возможно, пустой) список утверждений относительно встречающихся в ней "известных" объектов, истинность которых считается априори данной. Эти утверждения будем называть посылками задачи. Типичные примеры списков посылок - перечисление условий на известные параметры в системе уравнений; логическое описание чертежа в геометрической задаче на вычисление; список "данных" утверждений в задаче на доказательство. В зависимости от типа задачи, список посылок сопровождается рядом дополнительных элементов.

В случае задачи на доказательство добавляется то утверждение, относительно которого требуется установить, что оно является следствием посылок; будем называть такое утверждение условием задачи на доказательство.

В случае задачи на преобразование добавляется то выражение (называемое ее условием), которое должно быть преобразовано к некоторому специальному виду в предположении истинности посылок. В отличие от задачи на доказательство, здесь возникает необходимость сформулировать целевую установку, уточняющую желаемые вид и оптимизацию ответа (упростить; разложить на множители; проинтегрировать, и т.п.). Эта формулировка уже не относится к тому логическому языку "предметного уровня", на котором задаются посылки и условие. Хотя ее можно было бы задавать на некотором логическом языке, предназначенном для записи утверждений о логических структурах данных, на практике оказалось более удобным представлять целевую установку задачи как список специальных технических "пометок", называемых далее целями задачи. Это объясняется тем, что учетом целевых "пометок" занимается автоматика, управляющая логическим процессом и, как всякая автоматика, использующая свои узкоспециализированные нелогические структуры данных.

Задача на описание - это обобщение таких типов задач, как решение системы уравнений или неравенств. У нее, кроме списка посылок, имеется также некоторый список утверждений с неизвестными - они называются далее условиями задачи. Требуется дать описание всех или части значений неизвестных, при которых выполнены условия. Как и в случае задачи на преобразование, списки посылок и условий задачи на описание приходится сопровождать целевой установкой. Она уточняет вид искомого описания; определяет, нужно ли получить полное описание или достаточно лишь частичного (например, единственного примера значений неизвестных). Ответ задачи на описание обычно достигается в процессе последовательных преобразований ее списка условий. Однако, во многих случаях для получения ответа бывает необходимо накапливать некоторое многообразие следствий объединенного списка ее условий и посылок. В таком режиме решаются системы уравнений: извлекая следствия из исходных уравнений, иногда удается получить равенства, указывающие значения неизвестных. Этот же режим определения значений неизвестных путем вывода след-

ствий типичен для геометрических задач на вычисление. Занесение следствий непосредственно в список условий задачи нежелательно, так как впоследствии пришлось бы расчищать этот список, исключая все избыточные его элементы, что потребовало бы дополнительных вычислительных затрат на проверку избыточности. Поэтому в структуре данных задачи на описание предусмотрен специальный накопитель следствий условий и посылок. Роль такого накопителя играет вспомогательная задача на исследование (см.ниже), вводимая в процессе решения задачи на описание. Изначально она имеет своими посылками все посылки и условия задачи на описание.

Задача на исследование, в дополнение к списку посылок, имеет лишь целевую установку, уточняющую направленность логического вывода в этом списке. При решении ее исходное логическое описание некоторой ситуации в том или ином смысле "упрощается" и пополняется утверждениями, представляющими интерес в контексте целевой установки. Этот процесс обрывается либо по исчерпании возможностей добавления к имеющейся "картине" каких-либо ценных новых фактов, либо при получении следствий, требующих немедленного возвращения к внешней задаче, для которой предпринимается исследование. Заметим, что в решателе задачи на исследование используются только как вспомогательные - например, в роли накопителя следствий условий и посылок задачи на описание; в роли накопителя следствий при доказательстве от противного некоторого утверждения, и т.д. Различные математические задачи "на исследование" - такие, как исследование поведения функции вещественного переменного с помощью пределов и производных; исследование вида кривой или поверхности, заданной своим уравнением, и т.п., - оказалось целесообразно представлять в виде задачи на описание, сводя ее решение практически целиком к выводу следствий в связанной с ней задаче на исследование, и отбирая затем в качестве результата лишь некоторые из полученных фактов. Кроме того, отметим, что проведение исследований согласно заданной целевой установке для получения новых теоретических утверждений вообще не является задачей в рассматриваемом здесь узком смысле и никак не связано с применением введенных выше задач на исследование. Различные теоретические факты в математике являются замкнутыми логическими конструкциями, не имеющими общих варьируемых параметров, в то время как в рамках одной задачи обычно объединяются лишь такие утверждения, которые относятся к общим варьируемым переменным. Поэтому попытки организации исследований в теории как процесса логического вывода в списке посылок некоторой задачи не выявили никаких преимуществ и лишь привели к неоправданному замедлению вычислений. Процесс исследований в теории представляет собой "макрозадачу", для которой постановка обычной задачи и ее решение являются лишь атомарным шагами.

Кроме логических структур данных, задача содержит также некоторые вспомогательные технические структуры данных, вводимые в процессе работы самим решателем и используеыме для сохранения информации, направляющей его действия. Прежде всего, это пометки, указывающие на различные ранее предпринимавшиеся неудачные попытки, блокирующие их повторение, а также сообщения управляющего характера, которыми обмениваются между собой приемы. Для ускорения поиска в задаче объектов, связанных определенным образом с заданным объектом, создаются древовидные адресные конструкции. Работа приемов сопровождается потоком многократных обращений к различным вспомогательным процедурам, осуществляющим быстрые логические вычисления. Результаты таких обращений сохраняются в специальных буферах, что существенно ускоряет работу системы и упрощает организацию взаимодействия между приемами. Фактически эти буферы создают что-то

типа самоорганизующейся сетевой структуры данных, позволяющей быстро определять свойства объектов и связи между ними. Их применение оказалось особенно эффективным в геометрических задачах, насчитывающих сотни посылок.

Преобразования задачи, а также ввод и рассмотрение различных вспомогательных задач в процессе решения обеспечиваются приемами - процедурами локального анализа ситуации, способными усматривать целесообразность определенных действий и выполнять эти действия. Таких приемов в процессе обучения накапливаются многие тысячи, и для обеспечения быстрого поиска нужного приема необходима специальная организация базы приемов. В принципе, здесь возможны два подхода. Первый из них состоит в использовании древовидной поисковой системы, охватывающей все многообразие приемов. Вершины такого древовидного каталога определяют некоторые проверки, выполняемые для текущего состояния задачи, и в зависимости от результатов этих проверок осуществляется отсечение отдельных ветвей каталога, заведомо не содержащих нужного приема. Если после отсечения осталось более одной ветви рассматриваемой вершины, то далее поиск предпринимается независимо в каждой из оставшихся ветвей. После того, как нужный прием найден, он реализуется, и далее цикл поиска повторяется заново. Недостатком этого подхода оказалось в первую очередь то, что в реальном многообразии приемов не удалось найти сколь-нибудь хороших принципов отсечения, ввиду слабой зависимости между условиями применимости различных приемов. Кроме того, по мере обучения и увеличения числа приемов время поиска очередного приема будет при данном подходе увеличиваться. Это, безусловно, создаст (а при плохом отсечении - достаточно скоро) сильные ограничения на объем базы приемов, при котором возможна эффективная работа решателя. Первый подход представляет собой, по существу, цикл просмотра базы приемов на каждом шаге работы системы. В противоположность этому, второй подход основан на цикле просмотра задачи. Большинство приемов, используемых при решении задач, допускает явное указание такого понятия, что для возможности применения приема необходимо появление данного понятия в задаче. Это позволяет организовать базу приемов по принципу энциклопедии: за каждым понятием логического языка закрепляется сравнительно небольшая группа приемов этого понятия. Как показывает опыт, исключение составляет лишь крайне незначительная группа приемов, для которых не выделяется какого-либо естественного "инициализирующего" понятия. Очередной шаг решения задачи при втором подходе состоит в последовательном, понятие за понятием, просмотре всего текста задачи, с обращением для каждого текущего понятия к поиску внутри группы его приемов. Эта группа невелика, и во многих случаях ее можно организовать по древовидному принципу, получая таким образом дополнительное ускорение поиска. Преимуществом данного подхода является то, что при обучении системы новым разделам мы практически не уменьшаем скорости ее работы на ранее проработанных разделах. Это происходит потому, что "новые" понятия не встречаются в "старых" задачах, и наличие даже очень большого числа "новых" приемов никак не сказывается на времени поиска "старых", связанных с другими понятиями. Фактически, здесь снимаются сколь-нибудь сильные ограничения на рост числа приемов и появляется принципиальная возможность создавать гигантские логические системы, имеющие фундаментальное разностороннее образование. Заметим, что нынешняя версия решателя, охватывающая многие разделы элементарной и высшей математики и насчитывающая более 25000 приемов, занимает всего лишь порядка 60 мегабайт, в то время как используемая организация базы приемов и размеры жесткого диска компьютера позволяют довести ее размеры до гигабайт без существенного ухудшения быстродействия.

Таким образом, рабочий цикл решателя состоит в сканировании текста задачи - последовательном просмотре встречающихся в задаче понятий и обращении для текущего понятия к его программе, объединяющей в себе все закрепленные за понятием приемы. Эта процедура представляет собой что-то типа внутреннего "логического зрения", обеспечивающего контроль за развитием событий. Как и человек, в процессе решения система предпринимает мысленное "рассмотрение" объектов задачи и связей между ними для выработки плана ближайших действий. Чтобы выбрать из всех возможных действий наилучшее, необходимо каким-то образом сравнивать между собой результаты применения различных приемов в текущем контексте. Для этого естественно ввести числовые оценки приоритетности, выбирая каждый раз, например, прием с наименьшей такой оценкой. Вообще говоря, нерационально при сканировании задачи находить все возможные варианты применения приемов и лишь по окончании сканирования отбирать лучший - это может потребовать рассмотрения слишком большого числа вариантов, причем по своей априорной ценности анализируемые варианты тоже будут сильно различаться. К меньшим вычислительным затратам можно прийти, если разбить все приемы на группы или уровни, объединяя более-менее равноценные, и поиск нужного приема вести с последовательным увеличением номера уровня, обрывая его, как только найден применимый в текущей ситуации прием. Фактически, здесь оценкой приоритетности приема становится номер уровня. Так как обработка каждого уровня приемов требует своего цикла сканирования задачи, то число уровней целесообразно сделать не очень большим; в решателе это число равно 16, причем как правило срабатывание приема происходит на первых 5-6 уровнях. В действительности, один и тот же прием может в различных ситуациях относиться к различным уровням - номер уровня определяется, в зависимости от контекста, самой программой приема. Если этот уровень не соответствует тому, для которого выполняется текущее сканирование задачи, то дальнейшие действия по рассмотрению приема обрываются, и система переходит к другим приемам.

Важную роль при сканировании задачи играет управление переключением внимания. Вообще говоря, различные элементы описания задачи неравноценны при поиске очередного приема. Одни из них находятся в задаче давно, другие - только что занесены либо изменены. Вероятность обнаружить при рассмотрении первых ситуацию, требующую немедленных действий, обычно значительно ниже, чем для вторых. Чтобы учесть такую статистическую неоднородность элементов задачи и уменьшить среднее время поиска приема, элементы задачи (посылки и условия) снабжаются весами - целыми неотрицательными числами, указывающими номер того уровня сканирования, до которого ранее доходило рассмотрение элемента. Как только элемент изменяется, его вес уменьшается до 0; веса новых элементов также полагаются равными 0. При сканировании, связанном с приемами i - го уровня, игнорируются все посылки и условия, веса которых больше i - они образуют "теневую" часть задачи. По мере прохождения i - го уровня веса тех элементов задачи, для которых не произошло срабатывание приема, автоматически увеличиваются до $i+1$. Как только срабатывает какой-либо прием и возникают элементы задачи с весом 0, сканирование задачи возобновляется начиная с нулевого уровня. При этом в "зону внимания" попадут только те элементы задачи, которые имеют вес 0, т.е. только что были изменены либо введены. Эта простая автоматика уже обеспечивает в большинстве случаев разумную стратегию переключения внимания и значительно увеличивает быстродействие системы. В особенности это ощутимо для задач геометрического типа, имеющих сотни посылок и условий. Конечно, необходим ряд дополнительных средств, обеспечивающих переключение внимания в специальных случаях. Некоторые из этих средств

включены в "общую автоматику" решателя; другие - реализованы непосредственно в приемах, которые могут избирательно изменять веса различных элементов задачи и таким образом управлять зоной внимания.

Следующий важный вопрос, возникающий после выбора механизма сканирования задачи как диспетчера, организующего работу базы приемов, - это вопрос о языке для записи приемов. В приеме выделяются такие основные части, как описание ситуации, в которой логически допустимы реализуемые им действия; описание ситуации, в которой эти действия целесообразны, и описание процедуры, собственно реализующей действия приема. Первые две части предполагают использование некоторого логического языка, на котором формулируются соответствующие условия. Эти условия относятся не к объектам предметной области, рассматриваемым в задаче (числам, точкам, векторам и т.п.), а к тем структурам данных, с помощью которых задача представлена в системе: к термам логического языка, вхождениям в эти термы, к спискам термов, к различного рода техническим пометкам и конструкциям, вводимым решателем для управления процессом. Чтобы формулировать такие условия, необходимо обеспечить достаточно богатый набор понятий логического языка, ориентированных на используемые в решателе структуры данных, и прежде всего - на задачи и их элементы. Именно этот набор и составил основную часть нового языка ЛОС (Логический Описатель Ситуаций) для записи приемов; чтобы задавать те действия, которые должны быть выполнены приемом в уже "опознанной" им ситуации, оказалось достаточно присоединить к логической части языка сравнительно небольшое количество кодирующих типовые преобразования конструкций.

После того, как на логическом языке сформулирована ситуация, в которой применение приема возможно и целесообразно, необходимо обеспечить некоторый алгоритм, реализующий в задаче поиск этой ситуации. В описании ситуации должны фигурировать объекты, идентифицированные еще до начала поиска - сама задача; выделенное в ней при сканировании вхождение понятия; текущий уровень сканирования, и т.п. Описание представляет собой последовательность утверждений $P_1, \ldots, P_n$. Некоторые P_i определяют условия на ранее идентифицированные объекты. Так как все эти объекты относятся к текущим структурам данных решателя, истинность условий проверяется непосредственно и не требует какого-либо логического вывода. Для проверки этой истинности в интерпретаторе языка имеются специальные подпрограммы. Другие P_i вводят в рассмотрение новые объекты, связанные заданным образом с уже введенными до этого объектами. Если такие объекты определяются неоднозначно, то при поиске ситуации необходимо перечислять все возможные варианты. При программировании на обычном алгоритмическом языке "процедурного" типа здесь нужно было бы использовать операторы цикла, причем вложенность циклов была бы равна числу неоднозначно определенных переходов к "новым" объектам в цепочке $P_1, \ldots, P_n$. Чтобы избежать таких громоздких конструкций, в языке выделен ряд специальных отношений между объектами, для которых перечисление реализуется автоматически. При обработке "перечисляющего" утверждения P_i сначала рассматривается первая версия выбора новых объектов, и предпринимается попытка для выбранных объектов продвинуться вправо по цепочке $P_1, \ldots, P_n$. Если в некоторый момент дальнейшее продвижение, ввиду ложности условия, становится невозможным, то происходит откат к последнему "перечисляющему" утверждению и определение очередной версии выбора объектов. Этот принцип "перечисления по умолчанию" позволяет в качестве исполняемого текста программы ЛОСа использовать само логическое описание искомой ситуации и таким образом существенно повысить степень "читабельности" программы.

Заметим, что из совсем других соображений принцип реализации операторов с перечислением значений их выходных переменных возник в известном алгоритмическом языке ПРОЛОГ. В отличие от ЛОСа, ПРОЛОГ изначально ориентирован на работу с объектами предметного, а не структурного уровня. Поэтому центральное место в ПРОЛОГе занимает логический вывод, основанный на процедуре унификации. Этот логический вывод становится совершенно излишним при работе с элементами структуры данных, допускающими непосредственную проверку условий, и какого-либо аналога его в ЛОСе не предусмотрено. С другой строны, в ПРОЛОГе отсутствует целый ряд важных возможностей, предусмотренных в ЛОСе для эффективной работы со сложными логическими описаниями образов "структурного" уровня и для использования режима сканирования задачи. ЛОС ориентирован на создание автоматики, управляющей логическими процессами, в то время как при разработке ПРОЛОГа эта автоматика осталась почти "за кадром".

Впрочем, оба языка - и ЛОС, и ПРОЛОГ, в общем, относятся примерно к одному уровню, и оба обладают настолько существенными недостатками, что работу над созданием языка для записи приемов пришлось продолжить уже после того, как на ЛОСе был создан весьма эффективный решатель задач по элементатной алгебре. Как правило, значительную часть программы приема на ЛОСе составляет описание вида тех утверждений или выражений, которые могут быть идентифицированы с фрагментами заданной теоремы предметной области. На практике редко бывает так, чтобы идентифицируемые логические конструкции в точности совпадали с теми, которые имеются в теореме. Например, при идентификации квадратного трехчлена ax^2+bx+c коэффициент a может оказаться равным единице, и тогда в задаче вместо произведения ax^2 будет записана степень; сама эта степень может не иметь в точности вида квадрата, а лишь иметь четный коэффициент показателя степени, и т.д. Чтобы, несмотря на эти различия, идентифицировать теорему, приходится фактически в программе приема описывать не ее вид, а вид некоторого класса теорем, получающихся из нее различными вариациями. Текст программы оказывается, во-первых, весьма громоздким, и, во-вторых, достаточно удаленным от исходной записи теоремы. Еще большую громоздкось этому тексту придают различные вставки, связанные с решающими правилами приема. Поэтому, несмотря на все предоставляемые ЛОСом и интерфейсом его редактора возможности для быстрого чтения и понимания логических описаний ситуаций, все же в сколь-нибудь невырожденных случаях описания приемов оказываются трудночитаемыми. Заметим, что хотя ПРОЛОГ, казалось бы, и ориентирован на запись программы в виде совокупности теорем предметной области, однако он тоже не позволяет преодолеть указанной трудности. Если теоремы идентифицируются без учета возможных вариаций, то программа оказывается заведомо слабой; если начинается их учет, то и на ПРОЛОГе, для задания общего вида некоторого класса теорем, придется от предметного уровня перейти к структурному (к которому ЛОС более приспособлен), и таким образом полностью утратить наглядность исходного текста теоремы.

Чтобы восстановить утерянную при погоне за эффективностью работы приема наглядность и компактность записи, был создан новый язык, уровень которого существенно выше, чем у ЛОСа или ПРОЛОГа. Прием на этом языке задается как теорема предметной области, снабженная некоторой "алгоритмизирующей" разметкой. На основе разметки компилятор осуществляет необходимое обобщение вида теоремы, формулирует описание на ЛОСе ситуаций, в которых она должна быть применена, и добавляет к нему операторы ЛОСа, реализующие применение. Фактически этот язык имеет два независимых логических уровня: предметный уровень, на котором

представлена теорема, и структурный уровень, на котором формулируются условия целесообразности применения теоремы. Оба эти уровня предполагают использование полноценного логического языка, однако логические записи структурного уровня, в отличие от записей уровня предметного, не участвуют в каком-либо логическом выводе - они используются только для проверки условий при принятии решения. Роль "алгоритмизирующей" разметки приема весьма разнообразна - это и указание способа применения теоремы, и определение необходимых правил ее обобщения, и указание условий целесообразности применения, и уточнение алгоритмических средств, используемых для обработки посылок теоремы, и указание на "технические" сообщения, передаваемые при срабатывании данного приема другим приемам, и указатели переключения внимания, и многое другое. Эта разметка представляет собой что-то вроде генотипа приема, элементы которого допускают независимое варьирование в достаточно широких пределах и по которому компилятор создает фактически реализуемую программу приема на ЛОСе. Такая аналогия позволила назвать новый язык ГЕНОЛО́Гом (ГЕНетический язык ЛОГического программирования).

Описание приема на ГЕНОЛО́Ге оказалось значительно более компактным и понятным, чем на ЛОСе. На экране изображается в стандартной математической записи теорема приема, под которой размещается в нескольких окнах сопровождающая теорему информация. Во многих случаях эта информация занимает всего несколько строчек, в то время как текст создаваемой компилятором программы на ЛОСе составляет несколько полных экранов. Трудоемкость чтения приема и его коррекции, по сравнению с ЛОСом, уменьшилась на порядок.

В отличие от обычных языков программирования, ГЕНОЛО́Г не представляет собой сколь-нибудь завершенного явления. По существу, он является коллекцией типовых алгоритмических конструкций, используемых при переходе от теоремы к программе. Эта коллекция, достаточно богатая на текущий момент, продолжает (хотя все реже и реже) пополняться при рассмотрении новых предметных областей. Соответственно, при переходе к новой предметной области обычно приходится затрачивать некоторые усилия на развитие компилятора ГЕНОЛО́Га. Как показывает опыт, эти затраты несопоставимо малы в сравнении с той последующей экономией, которую дает применение ГЕНОЛО́Га при обучении решателя в данной области.

Главным доводом в пользу применения и развития ГЕНОЛО́Га является, впрочем, даже не наглядность представления приема и проистекающие из нее преимущества для ручной "дрессировки" решателя на тысячах задач из различных разделов, а возможность приблизиться вплотную к тому источнику, из которого приемы возникают - к логическому языку предметной области. Накопленное в решателе многообразие приемов можно рассматривать как задачник на извлечение приемов из теорем. Оно допускает некоторую предварительную классификацию приемов по типам, определяющим в общих чертах, какая именно цель достигается при использовании теоремы. Ряд элементов описания приема на ГЕНОЛО́Ге удается автоматически синтезировать по теореме и типу приема. Это используется при ручном обучении решателя, как средство пополнения текста приема различными стандартными элементами, и существенно облегчает процесс ввода приема. Однако, значительная часть элементов описания приема не может быть непосредственно извлечена из теоремы и общей установки на синтез приема. Эти элементы возникают для регулировки поведения приема в контексте всей базы приемов, и для их создания нужно развивать аппарат анализа взаимодействий между приемами, а также аппарат анализа решений, осуществляющий тестирование приема на серии задач.

При переходе от теоремы к приему, прежде всего, требуется определить класс

возможных типов приемов, создание которых по данной теореме имеет смысл. Во многих случаях такая целевая характеризация теоремы может быть выполнена "из общих соображений" и реализована сравнительно несложной процедурой. Однако, чаще всего сама теорема возникает для вполне определенной целевой нагрузки, и истоки приема расположены гораздо глубже - уже в самом процессе логического вывода, пополняющего запасы теорем.

Математическая логика не позволила сколь-нибудь существенно продвинуться в понимании динамики реальных математических теорий. Теория в ней обычно представляется как множество всех вообще утверждений, которые можно получить из аксиом с помощью правил вывода, или как множество всех утверждений, истинных в допустимых интерпретациях. За исключением вырожденных случаев, такое множество является бесконечным и раз навсегда заданным. С другой стороны, реальная математическая теория не просто конечная - заносимые в нее истинные утверждения подвергаются тщательному отбору. По-видимому, объяснение принципов этого отбора должно было бы основываться на целевой ориентации развития теории - то есть на анализе того алгоритмического аппарата решения задач, который она предлагает. Попытки обучения решателей показывают, что этот аппарат чрезвычайно сложен и эвристичен по своей сути. По существу, его можно уподобить "дрессированному хаосу". Включение такого аппарата в рамки теоретических построений математической логики весьма проблематично, а замена его простыми суррогатами не решает проблемы. С другой стороны, наличие эффективного решателя позволяет начать хотя бы экспериментальные исследования динамики теорий.

Новые утверждения в математической теории возникают в процессе логического вывода. Этот процесс основан на применении некоторых правил вывода к ранее полученным утверждениям либо к аксиомам. Простейший принцип отбора в состав теории получаемых таким образом утверждений состоит в удалении тех утверждений, применение которых при решении задач (после синтеза их приемов) сводились бы к последовательному применению уже имеющихся приемов. Эту проверку "на избыточность" легко организовать, если использовать решатель для упрощения новых утверждений - сохраняться будут только те, которые после упрощения не выродились в логическую константу "истина". Разумеется, целевая установка используемой здесь задачи на упрощение должна быть весьма аккуратно согласована с целевой установкой процесса поиска новых теорем. Такого рода проверка новых фактов на нетривиальность - в общем, обычное явление в практике математических исследований. Заметим, что последовательное применение правила вывода и решения задачи на упрощение приводит нас к более сложной процедуре вывода, использующей всю силу аппарата решателя - к своего рода "приему вывода". В общем, применение процедуры упрощения на выходе обычных правил вывода уже позволяет существенно уменьшить поток получаемых следствий, сохраняя только наиболее ценные из них. Следует заметить, что эксперименты с процессами логического вывода, использующими некоторую несложную процедуру упрощения ("редуцирования"), и попытки теоретического изучения этих процессов предпринимаются в математической логике уже достаточно давно (см. работы по "rewriting theory"). Однако, логический вывод с упрощением не устраняет экспоненциального роста числа получаемых следствий с ростом глубины вывода и годится лишь для получения серии первых, простейших следствий из исходных аксиом теории.

В нормальной практике математических исследований получению новых результатов обычно предшествует постановка задачи либо формулировка гипотезы. Лишь в простейших случаях новые утверждения получаются путем непосредственного выво-

да из ранее известных. Поэтому естественно предположить, что главным механизмом для развития теории служат не правила вывода, а более сложные процедуры, включающие в себя средства для постановки задач и выработки гипотез на основе анализа имеющихся теорем, и использующие решатель для работы с поставленными ими задачами. Такие процедуры будем далее называть приемами вывода теорем. Приведенная выше конструкция с последовательным применением правила вывода и задачи на упрощение является лишь частным примером такого рода приемов. Чтобы получать другие типы приемов вывода теорем, можно использовать в качестве обучающего материала различные уже известные утверждения из предметной области, предлагая возможно более простые объяснения логики "открытия" этих утверждений путем постановки вспомогательных задач и формулировки гипотез. Эта работа была проделана для некоторых простейших разделов (алгебра логики, алгебра множеств, свойства арифметических операций), в результате чего возникло около сотни различных приемов вывода. Даже для такой сравнительно сложной теоремы, как формула Кардано, удалось найти весьма простую и, по существу, общелогическую процедуру, извлекающую ее (разумеется, с помощью решателя), непосредственно из тождества для куба суммы. Продолжение этой работы могло бы привести к созданию значительной по своим размерам базе приемов вывода теорем, необходимой для процедур автоматического развития теории и синтеза приемов. Способность извлекать из исходных утверждений неожиданные и сильные следствия - за счет длинных цепочек логических вычислений, реализуемых решателем - позволила бы взять такой базе приемов на себя роль "генератора идей".

В последующих разделах книги мы дадим более подробное описание того комплекса, который был разработан для моделирования логических процессов. Он включает в себя технические средства для программирования на языках ЛОС и ГЕНОЛО́Г, созданный с помощью этих языков решатель задач, а также ряд экспериментальных средств для автоматизации синтеза приемов и автоматического развития базы теорем. Так как этот комплекс явно выходит за рамки понятия "решатель задач", и вместе с тем, безусловно, охватывает лишь небольшую часть тех возможностей, которые могли бы составить понятие "интеллектуальная система" (на сегодняшний день вообще представленное лишь естественным интеллектом), для названия его более уместно использовать термин "логическая система". К книге прилагается диск, на котором записана текущая версия этой системы, включающая в себя и те компоненты, работа над которыми лишь начата. В данной книге речь пойдет в основном о языках для программирования логических процессов; фактическому описанию решателей и средств автоматизации их развития будут посвящены вторая и третья книги. Тем не менее, из имеющихся в системе справочников и программ можно извлечь достаточно подробную информацию о текущем устройстве всех разделов системы и предполагаемых направлениях ее развития.

Излагаемые в книге принципы компьютерного моделирования логических процессов являются новыми и имеют своим источником анализ примеров. По этой причине, какие-либо библиографические ссылки в последующих разделах отсутствуют. Тем не менее, в конце приведен список литературы, ознакомление с которой позволит начинающему читателю получить необходимые представления об элементах математической логики и истории развития исследований по искусственному интеллекту. Чтение данной монографии предполагает наличие таких представлений. Классическими учебниками по математической логике являются книги [1], [2]. Хорошее изложение основ математической логики содержится также в [5]. Для знакомства с языком Пролог можно рекомендовать книгу [7], содержащую большое число простых примеров.

Монографий по искусственному интеллекту в списке литературы приведено всего три, хотя фактическое их число очень велико. Однако, описание логических подходов в таких монографиях обычно сводится лишь к изложению принципов автоматического доказательства теорем в логике предикатов и применению этих принципов в языке Пролог. Иногда добавляются разделы, посвященные немонотонным и модальным логикам. Прогресс в этой области по сравнению с первыми учебниками по искусственному интеллекту не очень велик, в чем нетрудно убедиться, сравнив хотя бы [9] и [11]. Рассматриваемые в данных книгах примеры моделирования процессов решения задач обычно крайне примитивны и создают впечатление явной недостаточности предлагаемой техники для систематического моделирования логических процессов в развитых предметных областях. Особняком в приводимом списке литературы стоят книги Д.Пойа [12], [13]. Они представляют собой пионерские работы по анализу логических процессов в математике и дают великолепную иллюстрацию того, как нужно разбивать процесс решения на отдельные шаги локального планирования действий. Впрочем, компьютерный решатель играет роль микроскопа для логических процессов, позволяя гораздо точнее оценить реальные возможности приемов, и проработка тех же примеров с его помощью вносит определенные коррективы в объяснения, полученные методом "самонаблюдения".

Кроме учебной литературы, приведен список основных задачников, использованных при обучении решателя. Конечно, в большинстве случаев эти задачники проработаны лишь частично. Задачи теоретического или нестандартного характера, которые требовали применения "одноразовых" приемов, малополезных в прочих ситуациях, при обучении обычно пропускались. В действительности такие задачи представляют собой хороший обучающий материал для развития системы автоматического синтеза приемов по теоремам и будут с этой целью использованы впоследствии.

Глава 2

Логический язык решателя задач

В системе используется логический язык открытого типа, пополняемый в процессе обучения. Для каждого нового понятия вводятся свои правила образования с его помощью корректных синтаксических конструкций и свои соглашения о их смысловой интерпретации, учитываемые при создании приемов, использующих данное понятие. При этом соблюдаются некоторые простейшие общие требования, которые будут изложены в данном разделе. Мы дадим достаточно подробное (хотя и не полное) перечисление используемых в языке способов записи понятий из различных разделов математики, так как представление о них необходимо для правильной постановки задач решателю. Кроме того, оно понадобится далее при разборе примеров. Напомним, что ниже речь идет лишь о внутреннем представлении утверждений в решателе; для диалога с пользователем применяется внешняя запись, приближенная к стандартной математической, которая автоматически транслируется во внутреннее представление. Впрочем, для получения необходимой логической однозначности иногда приходится вводить коррективы и в эту стандартную запись. Хотя в целом материал данной главы имеет справочный характер - перечисляются понятия из различных областей, отобранные для использования в решателе, - в отдельных случаях выбор формализации все же требует определенных пояснений.

Алфавит языка состоит из элементов двух типов - логических символов и символов переменных. Логические символы обозначают конкретные понятия (отношения между объектами, операции над ними, имена объектов, логические связки, кванторы и т.д.); переменные служат для обозначаения варьируемых объектов. Символы каждого типа пронумерованы последовательными натуральными числами. В компьютерной реализации для этих номеров зарезервированы диапазоны от 1 до $2^{16}-1$. Логические символы, по существу, отождествлены со своими номерами, и в этом смысле все они изначально имеются в алфавите логического языка. Однако, используемыми на текущий момент считаются только те из них, для которых введено название - слово либо словосочетание, имеющее не более 24 букв либо цифр. Такие названия хранятся в специальном файле и легко могут быть изменены. Внутренние алгоритмы системы этих названий не используют - они нужны только для отображения логических символов на экране в различных диалогах.

В языке рассматриваются конструкции двух типов - утверждения и выражения. Выражения используются для обозначения объектов; утверждения - для формулировки свойств объектов и отношений между ними. Как выражения, так и утверждения представляют собой записи со скобками (термы), определяемые следующим индуктивным образом:

1) Каждое однобуквенное слово, состоящее из логического символа либо символа

переменной, является термом.

2) Если $t_1, \ldots, t_n$ - термы; $n \geq 1$; f - логический символ, то слово $f(t_1 \ldots t_n)$ есть терм.

Помимо этого общего правила образования новых термов, с каждым используемым логическим символом f будут связываться свои дополнительные ограничения на элементы $t_1, \ldots, t_n$, при которых новый терм представляет собой осмысленное утверждение или выражение.

Еще одно предварительное соглашение общего характера относится к понятиям свободной и связанной переменных терма. Среди логических символов выделены несколько особых символов, называемых связывающими (к ним относятся кванторы общности и существования, а также символы для определения классов и функций). Вхождение переменной x в терм θ называется связанным, если оно расположено внутри подтерма θ' терма θ, имеющего вид $f(x_1 \ldots x_k x t_1 \ldots t_p)$, где f - связывающий символ; $x_1, \ldots, x_k$ - переменные, $t_1, \ldots, t_p$ - термы; $k \geq 0$; $p \geq 0$. Вхождения переменных в терм, не являющиеся связанными, называются свободными. Переменная, имеющая хотя бы одно свободное вхождение в терм, называется его свободной переменной, или параметром.

2.1 Общелогические понятия

Начнем с перечисления наиболее общих логических символов языка, сохраняя для них те же названия, которые используются в системе. Все эти названия будем выделять в тексте кавычками.

Прежде всего, выделим логические константы "истина", "ложь", а также логические связки "и", "или", "не", "эквивалентно", позволяющие строить новые утверждения "и($A_1 \ldots A_n$)"; "или($A_1 \ldots A_n$)"; "не(A_1)"; "эквивалентно($A_1 A_2$)"из ранее построенных утверждений $A_1, \ldots, A_n$.

Логическая связка "если-то"используется только в сочетании с квантором общности: "длялюбого($x_1 \ldots x_n$ если $A_1 \ldots A_m$ то A_0)", где $x_1, \ldots, x_n$ - попарно различные переменные, $A_0, A_1, \ldots, A_m$ - некоторые утверждения. Это обусловлено тем, что в бескванторных ситуациях предпочтительнее использовать связки "или", "и", "не", к которым она легко сводится. В случае $m = 0$ запись приобретает вид "длялюбого($x_1 \ldots x_n A_0$)". Аналогичным образом записывается утверждение существования значений $x_1, \ldots, x_n$, при которых истинно A_0: "существует($x_1 \ldots x_n A_0$)". Чтобы обеспечить возможность компактной записи утверждения о существовании и единственности этих значений, введена модификация квантора существования "Существует($x_1 \ldots x_n$ A_0)" (логический символ "Существует" - с большой буквы). Символы "длялюбого", "существует"и "Существует" - связывающие; переменные $x_1, \ldots, x_n$ образуют в приведенных выше утверждениях связывающую приставку.

Простейшими выражениями языка являются однобуквенные слова, образованные переменными либо логическими символами. Во втором случае считается, что выражение имеет своим значением тот символ, из которого оно состоит. Такое соглашение упрощает использование логического языка для описания вида его собственных конструкций. Хотя из него и вытекает, что константы "истина"и "ложь"одновременно должны считаться и утверждениями, и выражениями, особых неудобств в дальнейшем это не создает.

Для задания значения путем разбора случаев используется условное выражение "вариант(At_1t_2)". Если утверждение A является истинным, то значение этого

выражения равно значению выражения t_1, иначе оно равно значению выражения t_2. Аналогичным образом, рассматриваются условные утверждения "альтернатива(AB_1 B_2)". Если утверждение A истинно, то истинность условного утверждения совпадает с истинностью утверждения B_1, иначе - с истинностью утверждения B_2. Условные утверждения оказались практически неиспользуемыми в формулировках и решениях задач, однако они весьма часто применяются в логических описаниях структурного уровня - внутри программ приемов решателя.

Используются всего две конструкции, вводящие связанные переменные при построении новых выражений. Первая из них - "класс($x_1 \dots x_n$ A)" - определяет класс всех наборов ($x_1 \dots x_n$), на которых утверждение A является истинным. Вторая - "отображение($x_1 \dots x_n$ A t)" - задает функцию, определенную на множестве всех таких наборов ($x_1 \dots x_n$), для которых утверждение A является истинным, и принимающую на этом множестве значения, определяемые выражением t. Для задания интеграла, производной, предела, суммы по множеству значений параметра, и т.п., используются обычные операции над функциями, не вносящие сами по себе каких-либо связанных переменных.

К списку используемых в языке общелогических конструкций добавим также равенство "равно(t_1 t_2)" и выражение "значение(f t)", определяющее значение функции f в точке t (в обычной записи - $f(t)$).

2.2 Алгебра множеств

Утверждения "множество(A)" ; "принадлежит(t A)" ; "содержится(AB)"означают, соответственно, что A есть множество; t - элемент этого множества; множество A является подмножеством множества B. Для условия непересечения двух множеств A,B введено обозначение "непересек(A B)".

Пустое множество обозначается посредством логического символа "пусто". Выражения "объединение($A_1 \dots A_n$)", "пересечение($A_1 \dots A_n$)", "разность(A B)", "прямое-произведение($A_1 \dots A_n$)"используются для обозначения простейших операций над множествами.

Иногда приходится ссылаться на некоторый произвольный элемент непустого множества, не конкретизируя этот элемент. Для обозначения такого элемента введено выражение "элемент(A)". Аналогично, выражение "внешнийэлемент(A)"обозначает некоторый абстрактный элемент, не принадлежащий множеству A.

Для задания конечной последовательности элементов $A_1, \dots, A_n$ используется выражение "набор($A_1 \dots A_n$)". Всюду далее, говоря о наборах, мы отождествляем это понятие с понятием конечной последовательности. В тех случаях, когда приходится вводить операции либо отношения с переменным числом операндов (за исключением двуместных ассоциативных операций), обычно они представляются как одноместные операция либо отношение над набором. В частности, конечное множество, состоящее из элементов $A_1, \dots, A_n$, обозначается посредством выражения "перечень(набор(A_1 $\dots$ A_n))". Для пустого набора (последовательности длины 0) введено обозначение "пустоеслово".

Для результата последовательной записи наборов $A_1, \dots, A_n$ используется выражение "конкатенация($A_1 \dots A_n$)". Результаты добавления к началу либо концу набора A элемента t обозначаются, соответственно, "префикс(t A)"и "суффикс(A t)". Чтобы выделить из набора ($A_1 \dots A_n$) поднабор ($A_i \dots A_j$), используется обозначение "поднабор(A i j)". Для обозначения результата вставки в указанный набор элемента

B перед i-м элементом служит выражение "Вставка(A i B)".

Утверждение "семействомножеств(A)"означает, что A есть функция, принимающая в качестве своих значений множеств. Выражения "объединениевсех(A)", "пересечениевсех(A)" обозначают объединение и пересечение множеств для всевозможных значений аргумента функции A, принадлежащих ее области определения.

Обычным образом используются выражение "мощность(A)", утверждение "конечное(A)"и константы "счетное", "континуум".

Перечислим обозначения, введенные для работы с числовыми множествами. Числовой промежуток с концами a, b и указателями c, d отнесения этих концов к промежутку (0 - конец не включается в промежуток, 1 - включается) обозначается "промежуток(a b c d)". В частности, таким образом могут задаваться и бесконечные промежутки; для обозначения их концов (и в других аналогичных случаях) используются логические символы "минусбеск"и "плюсбеск". Напомним, что в данном разделе описывается только внутреннее логическое представление, используемое решателем; при отображении на экране процесса решения задач числовые промежутки изображаются традиционным образом, с использованием круглой скобки для указания невключения конца в промежуток и квадратной - если конец относится к промежутку. Множество вещественных чисел во внутреннем представлении обозначается посредством "промежуток(минусбеск плюсбеск 0 0)", хотя на экране оно прорисовывается как R. Утверждение "числовойотрезок(A)"означает, что A есть отрезок на числовой прямой (включая концы).

Множества целых, целых неотрицательных, натуральных и рациональных чисел обозначаются, соответственно, "целые" , "целыенеотрицательные", "натуральные"и "рациональные" .

Множество целых чисел, состоящее из элементов $i, i{+}1, \ldots, j$, обозначается "номера(i j)" . Набор тех же самых элементов обозначается "наборномеров(i j)" . Множество всех членов арифметической прогрессии с первым элементом a и знаменателем p обозначается "арифмпрогрессия(a p)".

Утверждения "нижняягрань(x A)" , "Нижняягрань(x A)" , "верхняягрань(x A)" , "Верхняягрань(x A)"означают, соответственно, что x есть нижняя либо верхняя грань числового множества A, причем большая заглавная буква указывает на строгую верхнюю грань, а ее отсутствие - на нестрогую. Утверждения "огрснизу(A)"и "огрсверху(A)"означают, что множество ограничено снизу либо, соответственно, сверху. Точная нижняя и точная верхняя грани обозначаются "инф(A)"и "суп(A)". Утверждения "наибольший(x A)"и "наименьший(x A)"указывают наибольший либо наименьший элемент x множества A.

Обычным образом используются выражения "граница(A)" , "внутренность(A)" , "замыкание(A)".

Для работы с функциями используются обозначения "функция(f)"(утверждение о том, что f есть функция); "область(f)"(область определения функции f); "значения(f)"(множество значений функции f); "Отображение(f A B)"(утверждение о том, что f есть функция, определенная на множестве A и принимающая значения в множестве B); "взаимнооднозначно(f)"(утверждение о том, что функция в различных точках своей области определения принимает различные значения).

Образ множества M для функции f обозначается "образ(f M)" ; прообраз множества M - "прообраз(f M)" ; прообраз элемента t - "слой(f t)".

Для задания конкретных функций используются следующие выражения.
"конст(M b)"обозначает константную функцию, принимающую во всех точках своей области определения M значение b. "См(a b)"обозначает функцию, определенную на

одноэлементном множестве $\{a\}$ и принимающую на нем значение b. "тождфунк(a)" обозначает тождественную функцию, определенную на множестве a. "таблица($f_1 \ldots f_n$)"обозначает результат "объединения" функций $f_1, \ldots, f_n$, принимающих на пересечении своих областей определения одинаковые значения. "доопределение(f M b)" обозначает результат доопределения функции f в тех точках множества M, где она еще не определена, значением b. "сужение(f M)" обозначает сужение функции f на множество M. Утверждение "перестановка(f A)"означает, что функция f представляет собой взаимно-однозначное отображение начального отрезка натурального ряда на конечное множество A.

Число переменных многоместной функции f обозначается "числопеременных(f)" ; функция, получающаяся из f подстановкой констант набора b вместо переменных, номера которых (начиная с 1) перечислены в наборе a, обозначается "подфункция(f a b)" . Утверждение "существперем(i f)" означает, что i - я переменная функции f является существенной.

2.3 Элементарная алгебра

Все рассматриваемые в этом разделе операции и отношения являются вещественнозначными; для работы с комплексными числами предусмотрены альтернативные обозначения.

Для представления десятичного числа $a_1 \ldots a_n$ ($a_1, \ldots, a_n$ - цифры) в виде терма используется запись "величина($a_1 \ldots a_n$)" . Если число дробное, то одно из a_i в этой записи является символом запятой (запятая введена в число логических символов только для данной цели; при изображении на экране термов в стандартной математической записи вместо запятой используется точка). Заметим, что в случае $n = 1$ символ "величина"не используется, а число представляется цифрой. Для простейших операций применяются обозначения "плюс($a_1 \ldots a_n$)" (здесь и далее $n \geq 2$); "минус(a)" ; "умножение($a_1 \ldots a_n$)" ; "дробь(a_1 a_2)" ; "степень(a_1 a_2)" ; "максимум($a_1 \ldots a_n$)" ; "минимум($a_1 \ldots a_n$)" . Операция вычитания не введена, так как все равно при решении задач ее приходится выражать через "плюс"и "минус", а прорисовка вычитания в стандартной математической записи неотличима от выражения с плюсом и минусом.

Сумма и произведение конечного семейства значений обозначаются "суммавсех (F)"и "произведениевсех(F)". Здесь F - функция, определенная на некотором конечном множестве и принимающая в качестве своих значений суммируемые либо перемножаемые величины. При прорисовке в стандартной математической записи эти выражения изображаются обычным образом, с помощью знаков $\sum$ и $\prod$.

Неравенства записываются в виде "меньше(a b)" ; "больше(a b)" ; "меньшеилиравно(a b)" ; "большеилиравно(a b)". Утверждение "число(a)"означает, что a есть вещественное число. Логические символы "е"и "пи"обозначают соответствующие числовые константы.

Логарифм числа b по основанию a обозначается "логарифм(a b)" . Для прочих элементарных функций используются обозначения: "синус(a)" ; "косинус(a)" ; "тангенс(a)" ; "котангенс(a)" ; "секанс(a)" ; "косеканс(a)" ; "арксинус(a)" ; "арккосинус(a)" ; "арктангенс(a)" ; "арккотангенс(a)" ; "модуль(a)" . Выражение "сигнум(a)" считается равным минус единице для отрицательных чисел, не определенным в нуле, и равным единице для положительных чисел. "Сигнум(a)" доопределяется в нуле единицей. Для гиперболических функций используются обозначения "гипсинус(a)"

; "гипкосинус(a)" ; "гиптангенс(a)" ; "гипкотангенс(a)" .

Утверждения "целое(a)" , "натуральное(a)" , "четное(a)" , "рациональное(a)" , "простое(a)" уточняют тип вещественного числа a. Выражения "числитель(a)" и "знаменатель(a)" определяют, соответственно, целочисленный числитель и натуральный знаменатель рационального числа a (после сокращения соответствующей дроби). Утверждение "делит(m n)" означает, что целое число m делит целое число n. Выражения "нод($n_1 \dots n_k$)" и "нок($n_1 \dots n_k$)" означают, соответственно, наибольший общий делитель и наименьшее общее кратное целых чисел $n_1, \dots, n_k$. Утверждение "взаимнопросты(m n)" означает взаимную простоту целых чисел m и n. Выражение "целаячасть(a)" обозначает целую часть числа a. Для обозначения вычета целого числа m по натуральному модулю n служит выражение "вычет(m n)" .

Выражение "числосочетаний(m n)" обозначает число сочетаний из m по n; выражение "факториал(n)" - факториал целого неотрицательного числа n.

Формальные многочлены требуют в логическом языке специального представления - они, разумеется, не являются функциями, но не являются также и формулами. Последние суть слова в конечном алфавите и множество их всего лишь счетно, в то время как множество формальных многочленов над полем вещественных чисел континуально. Поэтому многочлены рассматриваются как абстрактные объекты, задаваемые тройкой (X, F, K). Здесь $X = (x_1, \dots, x_n)$ - упорядоченный набор символов переменных; F - поле; K - функция, определенная на некотором конечном подмножестве M множества n-ок целых неотрицательных чисел и принимающая значения из поля F. Эта функция сопоставляет набору степеней переменных $x_1, \dots, x_n$ коэффициент соответствующего одночлена, входящего в многочлен. Для обозначения многочлена используется выражение "многочлен(X F K)". Разумеется, для отображения формального многочлена на экране предусмотрено более удобное обозначение:
$\mu_{x_1 \dots x_n}(S)$, где S - обычным образом записанная сумма одночленов. Переход от одного представления многочлена к другому выполняется автоматически. Утверждения "Многочлен(A)" и "веществмногочлен(A)" используются для указания на то, что A есть многочлен (в первом случае - общего вида, во втором - над полем вещественных чисел).

2.4 Математический анализ

Предел функции вещественного переменного f в точке a обозначается "предел(f p a)" ; здесь p - указатель типа предела: $p = 0$ означает двусторонний предел; $p = 1$ - левый предел и $p = 2$ - правый предел. Предел числовой последовательности обозначается таким же образом, но в этом случае функция f определена на множестве натуральных чисел. Аналогичным образом используются обозначения "верхнийпредел(f p a)"и "нижнийпредел(f p a)" . Используемые в асимптотических оценках обозначения $f(x) = O(g(x))$ и $f(x) = o(g(x))$ представляют собой хороший пример контекстной семантики. Они имеют смысл только в сочетании с указанными в контексте допущениями о "стремлении" x к некоторому значению. Разумеется, можно было бы исключить из языка такую контекстную зависимость и сгруппировать указанное выше равенство с указателем на предельное поведение аргумента в одну синтаксическую конструкцию, однако это привело бы к неоправданно громоздким записям, так как обычно одни и те же указатели предельного поведения обслуживают множество различных равенств с O и o. Поэтому в языке решателя сохранено обычное

применение таких равенств. Небольшое изменение здесь затрагивает лишь *o*: вместо указанного выше равенства, во "внутреннем"представлении используется запись "$o(f(x), g(x))$" .

Значение производной функции f одного вещественного переменного в точке a обозначается "производная(f a)". Значение в точке a частной производной функции f, зависящей от n переменных, обозначается "частнпроизв(f K a)". Здесь $K = (k_1 \dots k_n)$ - набор кратностей дифференцирования по отдельным переменным; если $k_i = 0$, то по i - му аргументу дифференцирование не выполняется. Значение повторной производной функции f одного вещественного переменного также обозначается посредством "частнпроизв(f K a)" ,однако здесь уже K - не набор, а величина кратности дифференцирования. Утверждение "Производная(f g)"означает, что значения функции g в области ее определения суть значения производной функции f. Аналогично, утверждение "Частнпроизв(f i g)"означает, что значения функции g в области ее определения суть значения частной производной функции нескольких переменных f по ее i - му аргументу. Значение полного дифференциала порядка n функции f нескольких переменных в точке x при наборе a приращений значений ее аргументов обозначается "дифференциал(f n x a)".

Для обозначения неопределенного интеграла функции f одного вещественного переменного используется чисто техническая запись "Интеграл(f)". Предполагается, что ее значением является какая - то одна из множества возможных первообразных функций для f. В процессе решения задачи эта первообразная находится, и тогда указанная запись устраняется. Разумеется, для использования в базе теорем следовало бы взять не одноместную операцию над f типа указанной выше, а двуместное отношение между функцией и ее первообразной, либо (как это обычно и делается) в качестве значения для неопределенного интеграла брать все множество первообразных. Однако, первый способ приводит при формальном интегрировании к обозначениям, сильно отличающимся от общепринятых, а второй заставляет неоправданно усложнять язык, вводя в него операции над множествами функций. Поэтому для вычислений была выбрана указанная выше запись, как технически наиболее удобная, хотя и требующая достаточно жестких ограничений на контекст, при которых ее использование не приводит к противоречиям. Определенный интеграл функции f одной вещественной переменной по ее области определения обозначается "интеграл(f)". При прорисовке на экране этого интеграла в обычной записи происходит "извлечение"из описания области определения f пределов интегрирования и явное их указание. Для двойного и тройного интегралов введены обозначения "двойнойинтеграл(f)"и "тройнойинтеграл(f)", где f - числовая функция, зависящая, соответственно, от двух либо от трех переменных. Интегралы берутся по всей области определения f.

Утверждение "сходится(f)"означает сходимость последовательности (функции натурального аргумента) f. Числовой ряд рассматривается как функция натурального аргумента. Утверждение о сходимости ряда с общим членом A_i формулируется в терминах последовательности его частичных сумм как "сходится(отображение(n натуральное(n) суммавсех(i целое(i) & $1 \le i$ & $i \le n$ A_i)))". Сумма ряда с общим членом A_i обозначается как "суммавсех(отображение(n целое(n) & $k \le n$ A_n))".

Вспомогательное утверждение "стремится(x a p)"означает то же самое, что $x \to a$, причем p - указатель двусторонней ($p = 0$) либо односторонней ($p = 1$ - левая, $p = 2$ - правая) окрестности. Оно не имеет самостоятельной семантики, а служит для определения семантики других утверждений, встречающихся с ним в общем контексте (например, $f(x) = O(g(x))$). Для указания локальных свойств числовой функции

f введены обозначения "переменазнака(f a)" (функция меняет знак в окрестности точки a); "убываетвточке(f a)"и "возрастаетвточке(f a)". Убывание и возрастание здесь понимаются в строгом смысле. Утверждение "огрвточке(f a)"означает ограниченность функции f в окрестности точки a.

Для указания на то, что некоторое выражение A ограничено в том контексте, в котором оно встречается (модуль его не превосходит некоторой константы, фиксированной для всей области допустимых в контексте значений параметров), используется вспомогательное утверждение "ограничено(A)". По существу, здесь подразумевается функция, значения которой определяются выражением A, а область определения задается утверждениями из текущего контекста, однако ради технических удобств явно эта функция не вводится. Разумеется, это накладывает существенные ограничения на использование данного обозначения - оно встречается лишь во вспомогательной процедуре проверки ограниченности выражения в заданном контексте и в тех приемах, которые к ней обращаются.

Утверждение "Максимум(f A B c)" означает, что B есть множество всех точек максимума функции f на множестве A, причем c - значение ее в этих точках. В случае минимума используется запись "Минимум(f A B c)" . Утверждение "экстремум(f a b c)" означает, что функция f имеет экстремум в точке a; b - значение ее в этой точке, c - тип экстремума - логический символ "минимум"либо "максимум". Множество всех точек из внутренности области определения многоместной числовой функции f, в которых все ее частные производные первого порядка определены и равны 0, обозначается "стационарныеточки(f)". Множество всех точек из этой внутренности, в которых хотя бы одна частная производная первого порядка не определена, обозначается "особыеточки(f)".

Утверждения "убывает(f A)" , "возрастает(f A)" , "неубывает(f A)" , "невозрастает(f A)"обозначают строгую и нестрогую монотонность функции f на множестве A. Для указания выпуклости либо вогнутости служат обозначения "Выпуклавверх (f A)"и "Выпуклавниз(f A)".

Для обозначения непрерывности и равномерной непрерывности числовой функции f на множестве A используются записи "непрерывна(f A)" ,"равномернонепрерывна(f A)". Утверждение "периодична(f A)" означает, что функция f периодична на всей числой оси и число A является ее периодом (не обязательно наименьшим). Утверждения "четнаяфункция(f)"и "нечетнаяфункция(f)"указывают четную либо нечетную функцию.

Для переупорядочения элементов набора A по убыванию либо возрастанию значений числовой функции f на элементах этого набора служат обозначения "упорядубыв(A f)"и "упорядвозр(A f)".

Для операций сложения, умножения, и т.п. , применяемых к числовым функциям, а не к числам, в логическом языке должны быть введены независимые обозначения - двойники аналогичных обозначений для числовых операций. Так, для сложения числовых функций с одинаковой областью определения введено обозначение "плюсфунк" ; для умножения - "умножфунк" ; для изменения знака - "минусфунк" . Как и в случае чисел, первые две операции могут применяться к произвольному (большему единицы) числу операндов. На экране операции над функциями прорисовываются так же, как и соответствующие числовые операции. Ввод их формульным редактором тоже ничем не отличается от случая числовых операций; после такого ввода предпринимается анализ контекста, и при необходимости числовые операции автоматически заменяются на своих функциональных двойников. В особых случаях такую коррекцию можно выполнять вручную.

2.5 Элементарная геометрия

Почти все вводимые в элементарной геометрии понятия относятся одновременно и к двумерному, и к трехмерному случаям. Использование в задаче понятия, имеющего смысл только для двумерного случая, автоматически означает, что эта задача целиком планиметрическая. Чтобы ускорить работу решателя при рассмотрении планиметрических задач, в список посылок таких задач обычно вводится (как правило, автоматически самим решателем) техническая пометка "планиметрия". Эту пометку можно рассматривать как вспомогательное фиктивное утверждение, указывающее на существование некоторой общей плоскости, к которой относятся все рассматриваемые в задаче точки.

Утверждение "точка(A)"означает, что A есть точка трехмерного пространства (в планиметрическом случае эта точка относится к некоторой не уточняемой плоскости данного пространства). Выражение "прямая(A B)"обозначает прямую, проходящую через точки A, B. В случае совпадения этих точек уславливаемся считать, что данное выражение обозначает какую-то произвольную прямую, проходящую через рассматриваемую точку. Аналогично, выражение "плоскость(A B C)"обозначает плоскость, проходящую через точки A, B, C. Обучение решателя выполнялось таким образом, что он может работать только с прямыми и плоскостями, заданными явным образом через пару (тройку) точек с помощью указанных выражений. Поэтому в условиях задач не рекомендуется обозначать прямые и плоскости вспомогательными переменными. Тем не менее, в языке предусмотрены используемые в некоторых специальных случаях утверждения "Прямая(A)"и "Плоскость(A)", указывающие, что A является, соответственно, прямой либо плоскостью.

Выражения "отрезок(A B)"и "интервал(A B)"обозначают, соответственно, отрезок и интервал с концами в точках A, B. Выражения "луч(A B)"и "обратныйлуч (A B)"обозначают, соответственно, луч с началом в точке A, проходящий через точку B и луч, обратный данному лучу.

Расстояние межу точками A и B обозначается "расстояние(A B)". Расстояние от точки A до прямой либо плоскости B обозначается, соответственно, "расстдопрямой (A B)"и "расстдоплоскости(A B)". Расстояние между двумя замкнутыми множествами точек A и B обозначается "расстмежду(A B)".

Величина угла с вершиной в точке B, стороны которого проходят через точки A и C, обозначается "угол(A B C)". Эта величина измеряется от 0 до π. Угол как фигура в этом случае обозначается "Угол(A B C)". Утверждение о том, что луч AD является биссектрисой угла ABC, записывается в виде "биссектриса(A B C D)". Прямая, на которой лежит данная биссектриса, обозначается "Биссектриса(A B C)".

Утверждения "параллельны(A B)"и "перпендикулярно(A B)"используются для указания на параллельность либо перпендикулярность прямых либо плоскостей A, B (допускаются любые сочетания: прямая - прямая; прямая - плоскость; плоскость - плоскость).

Для указания на то, что две точки A и B лежат по одну сторону от прямой либо плоскости C, используется утверждение "однасторона(A B C)". Утверждение "разныестороны(A B C)"указывает, что они лежат по разные стороны. Если C - прямая, то в обоих случаях утверждение указывает также и на принадлежность точек A, B общей плоскости с этой прямой. Если одна из точек попадает на прямую (плоскость), то считается, что точки одновременно лежат и по одну, и по разные стороны от прямой (плоскости). Утверждение "симметричны(A B C)"означает, что точки A и B расположены симметрично относительно точки, прямой либо плоскости C.

Утверждение "осьсимметрии(A B)"означае, что прямая A является осью симметрии плоской фигуры либо тела B. Утверждение "плоскостьсимметрии(A B)"означает, что плоскость A является плоскостью симметрии тела B.

Выражение "полоса(A B)"обозначает полосу между двумя параллельными прямыми A и B. Выражения "полуплоскость(A B)"и "обрполуплоскость(A B)"обозначают полуплоскости, лежащие по одну сторону от прямой A; первая из них содержит точку B, а вторая - не содержит этой точки (хотя плоскость этой полуплоскости содержит B).

Утверждение "треугольник(ABC)"означает, что точки A, B, C суть вершины треугольника (различны и не лежат на одной прямой). Аналогично, утверждения "параллелограмм($ABCD$)" , "ромб($ABCD$)" , "прямоугольник($ABCD$)" , "квадрат($ABCD$)", "трапеция($ABCD$)"означают, что четверка точек A, B, C, D составляет набор вершин четырехугольника соответствующего типа. В первых четырех случаях допускаются произвольные циклические перестановки; в последнем случае вершины A, D должны лежать на большем (нижнем) основании трапеции. Заметим, что в решателе предусмотрены два обозначения для трапеции: указанное выше предполагает, что оба угла при нижнем основании не больше 90 градусов; в обозначении "Трапеция($ABCD$)" это предположение отсутствует. Утверждение "четырехугольник($ABCD$)"означает, что четверка точек A, B, C, D образует вершины выпуклого четырехугольника. Если a - набор точек, то утверждения "многоугольник(a)" ; "правмногоугольник(a)" означают, что данные точки образуют последовательно проходимые вершины некоторого многоугольника (во втором случае - правильного). Выражение "фигура(a)" обозначает множество всех точек многоугольника, ограниченного ломаной, проходящей через точки набора a. Утверждение "центр(P a)"означает, что точка P является центром области a.

Утверждения "Медиана($ABCD$)" и "Высота($ABCD$)" означают, что точка D является основанием, соответственно, медианы либо высоты треугольника ABC, проведенной из вершины A. Утверждение "Биссектреуг($ABCD$)"означает, что точка D является основанием биссектрисы этого же треугольника, проведенной из вершины B.

Выражения "площадь(a)"и "периметр(a)"обозначают площадь и периметр области a (во втором случае - только имеющей вид многоугольника). Выражение "длина(a)" обозначает длину линии a.

В двумерном случае окружность с центром в точке A обозначается либо как "окружность(AB)" , где B - некоторая точка на этой окружности, либо как "окр(A r)", где r - радиус. Аналогичные обозначения "круг(AB)"и "круградиуса(A r)"используются для круга. В трехмерном случае задание окружности либо круга дополняется указанием третьей точки, лежащей в ее (его) плоскости. Здесь используются обозначения "Окружность(ABC)"и "Круг(ABC)"; C - дополнительная точка, фиксирующая плоскость.

Дуга окружности с центром A, имеющая концевые точки B, C, обозначается либо "дуга(ABC)" (меньшая из двух дуг), либо "большаядуга(ABC)" (большая из двух дуг). В случае диаметрально противоположных точек B, C большая и меньшая (условно) дуги выбираются некоторым неуточняемым образом, причем различны. Выражение "дугаугла(ABC)"обозначает дугу, на которую опирается вписанный угол ABC.

Чтобы задавать область со сложной границей, образованной множеством фрагментов стандартного вида (отрезки прямых, дуги окружностей и т.п.), используется запись "Фигура(перечень(набор($A_1 \dots A_n$)))" , где $A_1, \dots, A_n$ - обозначения данных

фрагментов.

Выражения "сектор(A B C)"и "сегмент(A B C)"обозначают, соответственно, сектор и сегмент круга, имеющего центр в точке A и определяемые крайними точками B, C, лежащими на окружности. Выбирается меньшая из дуг между данными точками (в случае равенства дуг берется неуточняемым образом одна из них). Выражение "кольцо(A B C)"обозначает кольцо с центром в точке A, внутренняя граница которого проходит через точку B, а внешняя - через точку C.

Утверждение "касательная(A B)"означает, что прямая A является касательной к окружности B. Утверждения "внешкасательная(A B C)"и "внутркасательная(A B C)"означают, что прямая A является, соответственно, внешней либо внутренней общей касательной окружностей B и C. Утверждения "внешкасаются(A B)"и "внуткасаются(A B)"означают, что окружности A и B касаются друг друга, соответственно, внешним либо внутренним образом.

Утверждение "вписана(A B)"означает, что окружность A вписана в многоугольник B (последний обычно задается выражением вида "фигура(...)"). Аналогично, утверждение "описана(A B)"означает, что окружность A описана около многоугольника B.

Утверждение "подобны(A B)"означает, что многоугольники, наборы вершин которых суть A и B, подобны. Заметим, что здесь вместо обозначающего многоуольник выражения "фигура(A)"используется выражение A, определяющее набор его вершин. Утверждение "подобны(A B)"дополнительно фиксирует соответствие между вершинами, при котором имеет место подобие: вершине A_i соответствует вершина B_i.

Утверждение "выпукло(A)"указывает на выпуклость множества A.

В отличие от планиметрии, в стереометрии не введены обозначения тел и поверхностей через их "опорные"точки. В задачах они должны обозначаться переменными. Утверждения "куб(a)" ; "призма(a)" ; "параллелепипед(a)" ; "пирамида(a)" ; "усеч-пирамида(a)" ; "конус(a)" ; "шар(a)" ; "сфера(a)" ; "цилиндр(a)" указывают тип тела либо поверхности a. Дополнительно к этим утверждениям, можно применять также утверждения "прямой(a)" (случаи прямой призмы и прямого параллелепипеда), "правильный(a)" (случаи правильных призмы, пирамиды либо усеченной пирамиды) и "прямоугольный(a)" (применительно к параллелепипеду).

Выражения "объем(a)" , "площповерхности(a)" , "боковаяповерхность(a)"обозначают, соответственно, объем, полную площадь поверхности и площадь боковой поверхности тела a. Для обозначения площади поверхности a, как и в плоском случае, используется выражение "площадь(a)". Высота тела a (пирамида, конус, цилиндр и т.п.) обозначается "высота(a)" . Выражение "радиус(a)"обозначает радиус шара либо сферы a.

Утверждение "грань(a b)" означает, что многоугольник a является гранью многогранника b. Утверждение "основание(a b)"означает, что плоская фигура a служит основанием тела b. Утверждения "вершина(a b)" ; "ребро(a b)" означают, что точка либо, соответственно, отрезок a являются вершиной либо ребром многогранника b. Для указания на то, что точка a является вершиной пирамиды либо конуса b, служит запись "Вершина(a b)" . Запись "диагональ(a b)" означает, что отрезок a является главной диагональю параллелепипеда b. Утверждение "центр(a b)" указывает точку a - центр тела b. Осевое сечение a цилиндра либо конуса b вводится при помощи утверждения "осевоесечение(a b)" . Выражение "плоскостьфигуры(a)"обозначает плоскость двумерной фигуры a, расположенной в пространстве.

Выражения "полупространство(a b)"и "обрполупространство(a b)"обозначают полупространство, ограниченное плоскостью a и, соответственно, содержащее либо не содержащее точки b. Выражение "Полоса(a b)"обозначает полосу между двумя параллельными плоскостями a и b.

Для обозначения двугранного угла используются две различных записи. Первая из них имеет вид "двугрУгол(A B C d)", где A, B - плоскости, ограничивающие угол; C - точка, не лежащая на этих плоскостях; d - указатель отнесения этой точки. Если $d = 0$, то точка лежит внутри двугранного угла; если $d = 1$, то она находится в смежном с ним относительно плоскости A двугранном угле; если $d = 2$, то она находится в смежном с ним относительно плоскости B двугранном угле; если $d = 3$, то она находится в двугранном угле, вертикальном с данным. Вторая запись имеет вид "двугранУгол(A B C)" . Здесь B - прямая, лежащая в вершине двугранного угла; B и C - точки, лежащие на полуплоскостях, являющихся его сторонами. В первом случае величина двугранного угла обозначается "двугругол(A B C d)" ; во втором - "двугранугол(A B C)".

Трехгранный угол с вершиной в точке A и направляющими векторами ребер AB, AC, AD обозначается "трехгранугол(A B C D)".

Величина острого угла между прямыми либо плоскостями a и b обозначается "уголмежду(a b)". Величина того угла между двумя прямыми a, b, в котором проходит прямая c (все три прямые проходят через вершину угла), обозначается "Уголмежду(a b c)".

Условие принадлежности точки D биссекторной плоскости двугранного угла, в вершине которого расположена прямая B, а точки A, C лежат на полуплоскостях - сторонах этого угла, записывается в виде "биссектрплоск(A B C D)".

В дополнение к указанным выше, введен ряд специальных планиметрических обозначений для операций, позволяющих при решении задач на построение выражать новые элементы чертежа через уже известные. Выражение "тчк(A B r)"обозначает точку, отложенную на ориентированной прямой AB от точки A на величину r (при неотрицательном значении - на луче AB, иначе - на продолжении этого луча). Выражение "доляотрезка(A B a)"обозначает точку C ориентированной прямой AB, для которой отношение ориентированных длин отрезков AC и AB равно a.

Выражение "перпендикуляр(A B)"обозначает прямую, проведенную через точку B перпендикулярно прямой A, а выражение "параллелпрямая(A B)" - прямую, проведенную через указанную точку параллельно прямой A. Выражение "параллель(A b B)"обозначает прямую, параллельную прямой A, находящуюся от нее на расстоянии b и расположенную в той же полуплоскости, что и точка B. Наконец, выражение "поворотпрямой(A B C a)"обозначает прямую, получаемую при поворотое луча AB на угол a в направлении полуплоскости, содержащей точку C.

2.6 Аналитическая геометрия

Аффинная система координат на плоскости либо в пространстве отождествляется, соответственно, с тройкой (A, B, C) либо с четверкой (A, B, C, D) точек. Здесь A - начало координат; B, C, D - концы координатных векторов, начинающихся в A. Какого-либо специального символа одноместного отношения, выделяющего наборы, являющиеся аффинными системами координат, не предусмотрено. Для выделения прямоугольной системы координат K используется утверждение "прямкоорд(K)",

означающее, как и обычно, что координатные векторы попарно ортогональны и имеют длину 1.

Выражение "коорд(A K)"используется для обозначения координат точки, вектора либо множества точек A в аффинной системе координат K. В первых двух случаях его значением служит координатный набор (пара либо тройка); во втором случае - множество координатных наборов. Выражение "тчкоорд(K A)"обозначает точку, имеющую координатный набор A в аффинной системе координат K; выражение "точки(A K)"обозначает множество точек, координатные наборы которых в аффинной системе координат K образуют множество A.

Выражение "полкоорд(A K)"обозначает координаты точки A в полярной системе координат K. Сама полярная система координат при этом отождествляется с тройкой (P, Q, R), где точка P - начало системы координат, причем полярный угол отсчитывается от луча PQ в направлении луча PR. Аналогично, выражения "сферкоорд(A
K)" и "цилкоорд(A K)"обозначают координаты точки A в сферической и цилиндрической системах координат K. Под сферической системой координат понимается четверка (P, Q, R, S), где P - начало координат; широта отсчитывается от луча PQ в направлении луча PR; долгота - от луча PS в направлении плоскости PQR. Лучи PQ,PR,PS предполагаются взаимно перпендикулярными. Цилиндрическая система координат отождествляется с аналогичной четверкой точек, причем PS - ось ординат.

Вектор с началом в точке A и концом в точке B обозначается "вектор(AB)". Утверждение "Вектор(a)"означает, что a есть вектор (в плоскости либо пространстве). Нулевой вектор обозначается "вектор0". Операции сложения векторов, изменения знака вектора и умножения вектора на число обозначаются, соответственно, "плюсвект", "минусвект"и "умножвект". Первая из них применяется к произвольному, большему или равному 2, числу слагаемых; последняя - к двум операндам, первый из которых является числовым множителем, а второй есть вектор. На экране эти операции прорисовываются точно так же, как аналогичные операции над числами. Формульным редактором они вводятся как числовые операции, причем выполняется автоматическая их коррекция, основанная на анализе контекста. Длина вектора a обозначается "длина(a)"; угол между векторами a, b - "уголмежду(a b)"(этот угол измеряется от 0 до π).

Ортогональность, коллинеарность и компланарность векторов обозначаются, соответственно, "перпендикулярно(a b)", "коллинеарны(a b)"и "компланарны(a b c)". Конец вектора b, отложенного от точки a, обозначается "конецвектора(a b)". Скалярное и векторное произведения обозначаются "скалумнож(a b)"и "вектумнож(a b)". Выражение "проекция(a b)"обозначает ортогональную проекцию точки, вектора либо прямой a на прямую либо плоскость b. Утверждение "делениеотрезка(A B C a b)"означает, что точки A, B, C лежат на одной прямой, причем отношение вектора AB к вектору BC равно отношению числовых параметров a и b.

Указатель ориентации упорядоченного набора векторов B в системе координат A, равный 1, если ориентация положительная, и равный -1 в случае отрицательной ориентации, обозначается "ориентация(A B)". Ориентированные площадь и объем обозначаются "орплощадь(A B C)"и "оробъем(A B C D)". Здесь A - система координат, определяющая ориентацию плоскости либо пространства; в первом случае рассматривается площадь параллелограмма, порожденного векторами B, C, а во втором случае - объем параллелепипеда, порожденного векторами B, C, D. Ориентированный угол между векторами A, B при ориентации плоскости, задаваемой системой

координат C, обозначается "оруголмежду(A B C)". Утверждение "направление(A B C D)"означает, что вектор D трехмерного пространства направлен по отношению к плоскости, определяемой векторами A, B, в ту же сторону, что и вектор C.

Утверждение "Прямая(A)"означает, что A есть прямая на плоскости либо в пространстве. Выражение "углкоэффициент(A K)"обозначает угловой коэффициент прямой A в двумерной системе координат K.

Уравнение линии либо поверхности используется только внутри выражения "класс(...)", определяющего множество координатных наборов точек этой линии либо поверхности относительно некоторой системы координат. Так, координаты прямой на плоскости задаются выражением "класс(xy $ax + by + c = 0$ & число(x) & число(y))". Координаты плоскости в пространстве аналогичным образом выражаются в виде "класс(xyz $ax + by + cz + d = 0$ & число(x) & число(y) & число(z))". Для задания координат прямой в пространстве обычно используется выражение "класс(xyz пропорцнаборы($(x - a, y - b, z - c), (d, e, f)$) & число($x$) & число($y$) & число($z$))". Здесь используется запись "пропорцнаборы(A B)", означающая линейную зависимость двух числовых наборов A, B одинаковой длины. (a, b, c) - координаты некоторой точки прямой; (d, e, f) - координаты ее направляющего вектора. Реже используется задание уравнения прямой как пересечения двух плоскостей - "класс(xyz $ax + by + cz + d = 0$ & $px + qy + rz + s = 0$ & число(x) & число(y) & число(z))".

Утверждение "направлпрямой(A B C)"означает, что C есть координатный набор направляющего вектора прямой A в системе координат B. Утверждение "плоскбазис (A B a b)"означает, что a, b суть два координатных набора для ортогональных единичных векторов, лежащих на плоскости A, относительно прямоугольной системы координат B трехмерного пространства.

Утверждения "линвторпорядка(a)" , "эллипс(a)" , "гипербола(a)" , "парабола(a)" означают, стответственно, что a есть невырожденная линия второго порядка на плоскости, эллипс, гипербола либо парабола. Чтобы указать на равностороннюю гиперболу, используется запись "равнгипербола(a)" . Утверждение "направлпараболы(a b)"означает, что b есть вектор, определяющий направление оси параболы a. Утверждения "действось(a b)"и "мнимаяось(a b)"означают, что прямая a является, соответственно, действительной либо мнимой осью гиперболы b (кроме того, для указания произвольной оси симметрии a кривой b можно использовать уже введенное выше обозначение "осьсимметрии(a b)"). Утверждение "фокус(a b)"означает, что точка a является фокусом кривой второго порядка b; утверждение "директриса(a b c)" - что прямая a является директрисой кривой второго порядка c, соответствующей фокусу b. Утверждение "диаметр(a b)"означает, что прямая a является диаметром кривой второго порядка b; утверждение "сопряжнаправления(a b c)" - что направления прямых a и b являются сопряженными относительно кривой второго порядка c. Утверждение "асимптота(a b)"означает, что прямая a является асимптотой для кривой b.

Выражение "ветвькривой(a b)"обозначает ветвь кривой b, содержащую точку a. Фокальный параметр кривой a обозначается "фокпараметр(a)" ; длина хорды, проходящей через фокус кривой a и перпендикулярной фокальной оси, обозначается "фокхорда(a)" (удвоенный фокальный параметр). Длина полуоси эллипса b по направлению его оси a обозначается "полуось(a b)" ; длины большой и малой осей эллипса - "большаяось(a)"и "малаяось(a)" . Длины действительной и мнимой полуосей гиперболы a обозначаются "действполуось(a)"и "мнимаяполуось(a)" . Выражение "эксцентриситет(a)"обозначает эксцентриситет кривой a.

Утверждение "каноничкоорд(a b)"означает, что a есть каноническая система координат для кривой либо поверхности второго порядка b. Утверждение "фоксистема(a b)"означает, что a есть прямоугольная система координат с центром в фокусе кривой второго порядка b, положительное направление оси абсцисс которой в случае параболы совпадает с направлением оси параболы, в случае эллипса - с направлением на другой фокус, а в случае гиперболы - противоположно этому направлению.

Утверждения "поввторпорядка(a)" ; "Цилиндр(a)" ; "Конус(a)" ; "эллипсоид(a)" ; "гиперболоид(a)" ; "параболоид(a)"означают, соответственно, что a есть невырожденная поверхность второго порядка; цилиндрическая поверхность; коническая поверхность; эллипсоид; гиперболоид и параболоид. Для уточнения типа поверхности используются (в сочетании с указанными выше) также утверждения "круглый(a)"(цилиндр либо конус); "эллиптический(a)" ; "гиперболический(a)" ; "параболический (a)" ; "однополостный(a)"и "двуполостный(a)".

Утверждения "осьцилиндра(a b)"; "образующая(a b)"означают, что прямая a является, соответственно, осью либо образующей цилиндра b. Утверждение "направляющая(a b)"означает, что кривая a служит направляющей цилиндра b. Утверждение "оськонуса(a b)"означает, что прямая a является осью конуса b; утверждение "осьвращения(a b) что прямая a есть ось вращения поверхности b. Утверждение "направлпараболоида(a b c)"означает, что координатный набор c определяет направление оси параболоида a в системе координат b.

В случае эллипсоида вращения a длина той оси, относительно которой происходит вращение эллипса, обозначается "высота(a)"; длина другой полуоси эллипса - "радиус(a)". В общем случае, длина полуоси эллипсоида по направлению его оси a (т.е. a - соответствующая прямая) обозначается "полуось(a b)".

Утверждение "диаметрплоскость(a b)"означает, что a есть диаметральная плоскость поверхности второго порядка b. Утверждение "сопряжнаправления(a b c)"означает, что направления плоскости a и прямой b являются сопряженными относительно поверхности второго порядка c.

Утверждение "собствкоорд(a b)"означает, что a есть прямоугольная система координат, единичные векторы которой задают направления главных осей поверхности второго порядка b.

2.7 Элементы линейной алгебры

Матрицы рассматриваются как функции от двух натуральных переменных, пробегающих номера строк и столбцов. Если размерность матрицы нефиксирована, то она задается явно через описатель "отображение": "отображение(ij и(принадлежит(i номера($1 \ldots n$))принадлежит(j номера($1 \ldots m$))) $A(i,j)$)", где $A(i,j)$ - выражение, определяющее общий вид элемента. Для матрицы фиксированной размерности можно применять также задание ее по строкам: "строки(набор($a_{11} \ldots a_{1n}$) ... набор($a_{m1} \ldots a_{mn}$))"либо по столбцам: "столбцы(набор($a_{11} \ldots a_{m1}$) ... набор($a_{1n} \ldots a_{mn}$))".

Определитель и ранг матрицы a обозначаются "определитель(a)"и "ранг(a)".

Для сложения и изменения знака матриц, как и произвольных числовых функций, используются выражения "плюсфунк($a_1 \ldots a_k$)"и "минусфунк(a)". Результат умножения матрицы a на число k обозначается "матркоэфф(k a)"; результат перемножения матриц - "умножматр($a_1 \ldots a_k$)". Результат возведения квадратной матрицы a в целую степень k обозначается "степеньматр(a k)". Выражение "линкомбинации(a)" обозначает множество линейных комбинаций матриц одинаковой размерности, при-

надлежащих множеству a.

Утверждение "собствзначение(a b k)"означает, что b есть собственное значение матрицы a, имеющее кратность k. Утверждение "собствектор(a b c)"означает, что c есть собственный вектор-столбец матрицы a, отвечающий собственному значению b.

Утверждение "каноничматрица(a x b)"означает, что матрица b является канонической формой полиномиальной матрицы a, элементы которой рассматриваются как многочлены от переменной x. Здесь снова используется контекстная семантика - элементы матриц a, b записываются не в виде функций или формальных многочленов, а непосредственно в виде выражений, определяющих значения этих функций или многочленов.

Утверждение "жордформа(a b c)"означает, что матрица b является жордановой нормальной формой матрицы a, причем c - матрица, трансформирующая a в b.

2.8 Комплексные числа

Для работы с комплексными числами пришлось продублировать обозначения, используемые в вещественном случае, сохранив способ их прорисовки на экране. Применение одних и тех же обозначений оказалось технически нецелесообразным хотя бы из-за того, что многие приемы вещественного случая в комплексном случае выглядят совершенно иначе. Параллелизм в обозначениях привел к необходимости создания параллельных приемов. Однако, имеющиеся в системе средства программирования позволили легко решить эту проблему. Проработка комплексного анализа в решателе продолжается, и приводимый ниже список обозначений будет расширяться.

Утверждение "комплексное(a)"означает, что a есть комплексное число. Мнимая единица обозначается "мнимаяединица" ; вещественная и мнимая части комплексного числа a - "вещественнаячасть(a)"и "мнимаячасть(a)" . Главное значение аргумента комплексного числа обозначается "аргумент(a)"; множество всех значений аргумента - "Аргумент(a)". Посредством "сопряженное(a)"обозначено число, комплексно сопряженное с a. Для арифметических операций и элементарных функций над комплексными числами введены обозначения "Плюс(a_1 ... a_n)" ; "Минус(a)" ; "Умножение($a_1 \ldots a_n$)" ; "Дробь(a b)" ; Степень(a b)" ; "Логарифм(a b)" ; "Синус(a)" ; "Косинус(a)" ; "Тангенс(a)"; "Котангенс(a)"; "Гипсинус(a)"; "Гипкосинус(a)"; "Гиптангенс(a)"; "Гипкотангенс(a)"; "Модуль(a)"; "Арксинус(a)"; "Арккосинус(a)"; "Арктангенс(a)"; "Арккотангенс(a)"; "Аркгипсинус(a)", "Аркгипкосинус(a)"; "Аркгиптангенс(a)"; "Аркгипкотангенс(a)", дублирующие аналогичные обозначения для вещественных чисел. При использовании стандартной математической записи эти обозначения вводятся и прорисовываются так же, как и в вещественном случае, однако при переводе во внутреннее представление они автоматически корректируются на основе контекста. Для степени и логарифма указанным выше образом обозначены главные значения соответствующих многозначных функций. Выражение "Корень(a n k)" обозначает k - й корень n - й степени из комплексного числа a; здесь k изменяется от 1 до n.

Чтобы можно было работать не только с главными значениями, но с произвольными ветвями элементарных функций, добавлены обозначения "ветвьстепени(a b n)" (n-я ветвь для a^b; при $n = 0$ имеем главное значение); "ветвьарксинуса(a n)"; "ветвьарккосинуса(a n)"; "ветвьарктангенса(a n)"; "ветвьарккотангенса(a n)"; "ветвьаркгипсинуса(a n)"; "ветвьаркгипкосинуса(a n)"; "ветвьаргкгиптангенса(a n)"; "ветвьаркгипкотангенса(a n)". Здесь n - номер ветви для соответствующего значе-

ния. Четные и нечетные номера в случаях арксинуса и арккосинуса, обычного либо гиперболического, соответствуют знакам "плюс" либо "минус" для радикала под логарифмом. n-я ветвь натурального логарифма обозначается "Натурлог(a n)".

Для перехода от рассмотрения комплексных чисел к изображающим их точкам плоскости и обратно используются выражения "комплформа(A)", "вещформа(A)". Первое из них обозначает множество комплексных чисел $a + bi$, у которых пары (a,b) принадлежат множеству A. Оно же может обозначать комплексное число $a + bi$, определяемое парой $A = (a, b)$. Второе обеспечивает обратный переход - дает множество пар (a,b) вещественных чисел, для которых комплексное число $a + bi$ принадлежит множеству A, либо единственную пару (a,b), определяемую комплексным числом $A = a + bi$.

Множество всех комплексных чисел обозначается символом "комплексные", который прорисовывается на экране стандартным образом, в виде несколько видоизмененной буквы C. Для удобства работы с аргументами комплексных чисел введен вспомогательный оператор "нормарг(a)", выполняющий приведение вещественного числа a по модулю 2π к промежутку от $-\pi$ (не включая это значение) до π.

Выражения "комплотрезок(a b)" и "комплинтервал(a b)" обозначают, соответственно, комплексный отрезок и комплексный интервал с концами в точках a,b. Ориентированный комплексный отрезок с началом в точке a и концом в точке b обозначен "Комплотрезок(a b)". Односвязность подмножества A точек комплексной плоскости обозначается "комплодносвязно(A)". Граничные точки множества A обозначаются выражением "комплграница(A)".

Для суммирования и перемножения конечных семейств A комплексных чисел введены обозначения "Суммавсех(A)", "Произведениевсех(A)", которые прорисовываются на экране стандартным образом, как $\sum$ и $\prod$.

Выражение "Предел(f t a)" обозначает предел функции комплексного переменного f в точке a. t - тип предела; если он равен 0, то берется полная окрестность точки a. Прорисовка комплекснозначного предела на экране такая же, как для вещественнозначного. Иначе t - подмножество окрестности, по которому находится предел. Утверждение "Бескмалая(f g a t)" означает, что отношение функций комплексного переменного f и g стремится к 0 при стремлении аргумента к точке a. t - указатель типа предела. Сходимость последовательности p комплексных чисел обозначается "Сходится(p)". Непрерывность функции комплексного переменного f на множестве M обозначается "Непрерывно(f M)". Утверждение "локНепрерывно(f a)" означает, что функция f непрерывна в какой-либо окрестности точки a. Чтобы указать, что a есть точка ветвления функции f, используется запись "точкаветвления(a)".

Производная функции комплексного переменного f в точке a обозначается "комплпроизводная(f a)". Прорисовывается она так же, как обычная производная. Если производная не определена, то указанное выражение имеет своим значением символ "неопред". Для обозначения производных высших порядков служит запись "компл-Производная(f a n)", в которой n указывает порядок. Прорисовка на экране этого выражения также стандартная. Условие дифференцируемости функции f в точке a записывается как "Дифференцируема(f a)".

Если функция $f(x + iy)$ комплексного переменного записана в виде $u(x, y) + iv(x, y)$, то возникают две вещественнозначные функции u, v. Утверждения "вещчастьфунк(f u)" и "мнимчастьфунк(f v)" связывают эти функции с функцией f. Если аргумент функции комплексного переменного f задан в полярных координатах (r, φ), и эта функция записана в виде $U(r, \varphi) + iV(r, \varphi)$, то аналогичным образом используются утверждения "вещчастьфункпол(f,U)" и "мнимчастьфункпол(f,V)".

Аналитичность функции f на множестве M записывается в виде "аналитическая(f M)". Утверждение "дроблинфунк(f)" выделяет дробно-линейные функции комплексного переменного.

Интеграл функции комплексного переменного f вдоль ориентированной комплексной кривой K обозначается "комплинтеграл(f K)". Вычет функции f в точке a записывается как "вычет(f a)".

Радиус сходимости степенного ряда с комплексными коэффициентами, образующими последовательность p, обозначен "радиусссходимости(p)".

Для указания типа особой точки a функции f служат утверждения "устранимособая(f a)", "существособая(f a)", "Полюс(f a n)", где n - порядок полюса.

2.9 Теория вероятности

Приведем список логических конструкций, возникших при обучении решателя теории вероятностей. Как и в предыдущем случае, он неполон из-за того, что обучение пока далеко не завершено. Утверждение "верпространство(A)" означает, что A есть вероятностное пространство. Выражение "равновер(A)" обозначает вероятностное пространство, определяемое на конечном множестве A соглашением о равновероятности его элементов, либо вероятностное пространство, определяемое на множестве A бесконечных последовательностей элементов конечного множества соглашением о равновероятности для каждого натурального n событий, получающихся при фиксации первых n элементов последовательности. Утверждение "событие(S A)" означает, что S есть событие в вероятностном пространстве A. Множество элементарных событий пространства A обозначается "элемсобытия(A)".

Вероятность события S в вероятностном пространстве A обозначается "вероятность(S A)". Условная вероятность события S в вероятностном пространстве A при наличии события D обозначается "услвероятн(S D A)".

Независимость событий семейства S в вероятностном пространстве A обозначается "незавсобытия(S A)". Утверждение "услнезавис(S D A)" означает, что каждое из событий семейства S является подсобытием события D, причем события этого семейства в сужении вероятностного пространства A на событие D являются независимыми. Утверждение "незавсерия(S A)" означает, что S есть семейство семейств $S(i)$ событий вероятностного пространства A, имеющих одну и ту же область определения D. Для каждого $d \in D$ это семейство определяет группу событий $S(i)(d)$, причем независимы любые события Q_j, построенные каждое при помощи теоретико-множественных операций из событий одной и той же группы и соответствующие различным группам. Аналогично, утверждение "услнезавсерия(S D A)" выражает условную независимость указанных событий Q_j относительно события D. Утверждение "незавгруппы(S A)" означает, что S есть семейство семейств событий вероятностного пространства A, причем события, полученные из событий различных семейств теоретико-множественными операциями, независимы. Еще одно условие независимости, связанное с семействами событий - "относитнезавис(S P A)". Оно означает, что S, P суть два семейства событий вероятностного пространства A, причем любые два события семейства S независимы в сужении вероятностного пространства A на любую теоретико-множественную комбинацию событий семейства P.

Утверждение "несовместны(S A)" означает, что любые два различных события семейства S являются несовместными в пространстве A.

Прямое произведение вероятностных пространств $A_1, \ldots, A_n$ обозначается "вер-

произв($A_1 \dots A_n$)".

В ряде задач оказалось полезным выражение "версочетания(d p n)", обозначающее вероятность одновременного выполнения ровно n независимых событий из списка, длина которого равна длине набора целых неотрицательных чисел d, а вероятность события, соответствующего числу m данного списка - вероятности объединения m независимых событий, каждое из которых происходит с вероятностью p.

Утверждение "случвеличина(f A)" означает, что f есть числовая функция, определенная на множестве элементарных событий вероятностного пространства A. Утверждения "непрвеличина(f A)", "дискрвеличина(f A)" означают, соответственно, что f есть непрерывная либо дискретная случайная величина. Функция распределения случайной величины f, заданной на пространстве A, обозначается "функраспред(f A)". Плотность распределения непрерывной случайной величины f обозначается "плотнраспред(f A)". Ряд распределения дискретной случайной величины f (функция, сопоставляющая каждму допустимому значению его вероятность) обозначен "рядраспред(f A)". Независимость семейства f случайных величин над вероятностным пространством A обозначается "незавслучвел(f A)".

Выражения "матожидание(f A)", "дисперсия(f A)", "средквадроткл(f A)" обозначают, соответственно, математическое ожидание, дисперсию и среднее квадратичное отклонение случайной величины f. Начальный и центральный моменты этой случайной величины, имеющие порядок n, обозначаются "начмомент(f n A)", "центрмомент(f n A)". Результат центрирования случайной величины f обозначен "центрирование(f A)". Мода случайной величины f обозначается "мода(f A)".

Условие распределенности случайной величины f по закону Пуассона с параметром p обозначается "Пуассон(f A p)". Условие распределенности случайной величины f по показательному закону с параметром p записывается в виде "показатзакон(f A p)". Утверждение "нормраспред(f A m s)" означает, что случайная величина f распределена по нормальному закону с математическом ожиданием m и средним квадратическим отклонением s. Значение функции распределения нормальной случайной величины с математическим ожиданием 0 и средним квадратическим отклонением 1 в точке x обозначается "Нормраспред(x)". Значение в точке x функции, обратной к данной, обозначено "обрнормраспред(x)".

Утверждение "случпоток(f A)" означает, что f есть случайный поток. Он рассматривается как семейство событий в вероятностном пространстве A, состоящих в том, что в заданный момент наступает событие потока (временная интерпретация), либо как семейство событий, состоящих в том, что в заданном месте числовой прямой расположена точка некоторого случайного множества точек (пространственная интерпретация). Случайная величина, равная числу событий случайного потока f, произошедших на множестве моментов времени M, обозначается "числособытий(f M)". Простейший случайный поток f, характеризуемый параметром p, задается утверждением "простейшийпоток(f p)".

Утверждение "случполе(f n A)" означает, что f есть случайное поле размерности n. Оно рассматривается как семейство событий в вероятностном пространстве A, состоящих в том, что в заданной геометрической точке располагается точка поля. Случайная величина, равная числу точек случайного поля f, попадающих в подмножество M, обозначается "числоточек(f M)". Посредством "равномерноеполе(f p)" обозначается равномерное случайное поле с плотностью p. Утверждение "незавсобытие(f S)" выражает независимость события S от любого события, состоящего в том, что число точек случайного потока либо поля f, попадающих на какой-то промежу-

ток, равно данной величине. Случайная величина, равная длине паузы между двумя последовательными событиями в случайном потоке A либо расстоянию от точки до ближайшей к ней в случайном поле обозначается "длинапаузы(A)".

2.10 Элементарная физика

Как и в других разделах, логический язык решателя для физики пополняется по мере надобности при рассмотрении последовательности обучающих задач. Физические задачи, даже относящиеся к одному и тому же разделу, часто бывают связаны с совершенно различными сценариями, и пополнение списка понятий, необходимых для описания этих сценариев, происходит до некоторой степени бессистемно. Каждый раз вводятся минимально необходимые для задачи средства. Видимо, впоследствии понадобится повторная итерация для упорядочения системы физических понятий. Они воспринимаются в контексте ряда умолчаний и не обладают той степенью самодостаточности, как математические понятия. Впрочем, это никак не сказывается на возможности развития решателей, работающих с такими понятиями. Просто в приемах больший акцент делается на фильтры, контролирующие корректность контекста. Приведем ту систему понятий, которая сложилась к текущему моменту проработки задачника.

Общие свойства процессов

Длительность процесса A; его начальный и последний моменты обозначаются, соответственно, "длительность(A)", "исхмомент(A)", "послмомент(A)". Временной отрезок, на протяжении которого протекает этот процесс, обозначается "Период(A)". Условие "внутрмомент(A t)" означает, что момент t относится к периоду процесса A. Если A, B - два процесса, то для указания на то, что временной промежуток первого расположен внутри временного промежутка второго, используется запись "внешпроцесс(A B)". В частном случае, когда A есть просто ограничение процесса B на меньший временной промежуток, применяется обозначение "подпроцесс(A B)". Утверждение "циклпроцессов(A B)" означает, что процесс A образован последовательно происходящими процессами набора B.

Общие свойства веществ

Плотность вещества A (масса на единицу объема) обозначается "плотность(A)". Утверждение "вещество(P A)" означает, что тело P состоит из вещества A. Для указания на то, что A есть химическое вещество, используется запись "Вещество(A)". Утверждение "типвещества(A)" означает, что A есть тип химического вещества. В задачах встречались такие типы, как "руда", "сталь", и т.п. Процентное содержание вещества A в веществе B (по массе) обозначается "содержание(A B)". То же самое, но по объему обозначается как "Содержание(A B)". Утверждение "извлечение(P A R)"означает, что R есть результат извлечения из тела P вещества A. Наконец, утверждение "твердоетело(P)" означает, что P есть твердое тело.

Общие свойства объектов

Масса объекта A обозначается "масса(A)". Для указания множества B составных частей объекта A используется утверждение "состоит(A B)". Утверждение "глад-

кповерхн(A)" означает, что твердое тело A имеет гладкую поверхность. Выделены несколько простейших геометрических типов тел. Утверждение "стержень(A B)" означает, что A есть стержень, концы которого (материальные точки) составляют множество B. Утверждение "брусок(A B C)" означает, что A есть твердое тело, имеющее форму прямоугольного параллелепипеда. B - одна из вершин этого параллелепипеда (материальная точка); C - тройка (множество) других его вершин, соединенных с B ребрами. Утверждение "клин(A B C D)" означает, что A есть твердое тело, имеющее форму прямоугольной призмы, в основании которой лежит треугольник. B - одна из вершин призмы; C - пара (множество) вершин, лежащих в одном основании с B; D - вершина, смежная с B и лежащая в противоположном основании.

Единицы измерения

Формально единицы измерения не являются числовыми величинами. Однако, они встречаются в формулах, при работе с которыми решатель иногда уточняет тип значений подвыражений, например, сомножителей. Если такой сомножитель представляет собой единицу измерения и решатель не распознает в ней числовое выражение, то могут оказаться заблокированы те или иные необходимые приемы. Поэтому, чтобы избежать перерегулировки уже имеющихся математических приемов, в решателе принято считать единицы измерения числами. Это всего лишь технический трюк. Никакой дополнительной информации о том, как соотносятся между собой значения разнородных физических величин, решатель не имеет, и нежелательных коллизий указанное соглашение не вызывает. Можно было бы работать с единицами измерения отдельно от числовых выражений, однако такой способ более сложен и не соответствует общепринятой практике.

Для обозначения единиц измерения применяются одноименные логические символы: "км", "м", "дм", "см", "сек", "мин", "час", "г", "кг". "т", "кГ", "Н", "Дж", "Вт", "кВт". В некоторых задачх встречалась единица измерения "день", обозначенная посредством символа "дн.".

Имена и физические константы

Для ускорения свободного падения и гравитационной постоянной используются логические символы "же" (прорисовывается как несколько модифицированная буква g) и "грав". Единственное встречавшееся в задачах имя собственное - "Земля" (как планета) обозначается одноименным логическим символом.

Траектории

Траектории перемещения материальных точек суть произвольные ориентированные кривые. Кроме физики, ориентированные кривые рассматривались также при обучении решателя математическому анализу, в подразделе, посвященном криволинейным интегралам. Поэтому приводимые ниже обозначения обслуживают сразу оба раздела. Чтобы получить ориентированную кривую, возникающую при последовательном прохождении ориентированных кривых набора K, используется выражение "путь(K)". Ориентированная кривая, обратная к ориентированной кривой A, обозначается "обратныйпуть(A)". Начальная и конечная точки ориентированной кривой A обозначаются, соответственно, "началопути(A)" и "конецпути(A)". Утверждение

"отрезокпути(A B)" означает, что ориентированная кривая A является частью ориентированной кривой B. Точка ориентированной кривой A, отстоящая от начала на расстоянии p, обозначается "точкапути(A p)". Множество точек ориентированной кривой A обозначается "точкипути(A)". Утверждение "простойпуть(A)" означает, что A есть ориентированная кривая без самопересечений.

Утверждение "закругление(A B C D)" означает, что траектория D имеет в точке A закругление, причем центр кривизны расположен в точке B, а точка C доопределяет соприкасающуюся плоскость.

Ориентированная кривая, соответствующая прохождению отрезка от точки A до точки B, обозначается "Отрезок(A B)". Ориентированная кривая, соответствующая движению по дуге окружности с центром в точке A, начинающемуся с точки B и продолжающемуся на величину ориентированного угла p (положительным считается движение против часовой стрелки), обозначается "Дуга(A B p)". Утверждение "общнаправл(A B)" означает, что ориентированные кривые A и B составлены из отрезков прямых, параллельных заданному направлению. Утверждение "прямлинпуть(A)" означает, что A есть ориентированная кривая, соответствующая прохождению некоторого отрезка от одного его конца к другому. Направляющий вектор ориентированной кривой A, представляющей собой отрезок, обозначается "напрпути(A)".

Чтобы получать ориентированную кривую из обычной кривой A, служит запись "ориенткривой(A r)", где r - указатель ориентации. Рассматриваются следующие типы указателей ориентации:

1. "убывает(K i)" - в системе координат K положение точки кривой однозначно определяется ее i-й координатой, и кривая проходится в направлении убывания значения этой координаты;

2. "возрастает(K i)" - аналогично предыдущему, но кривая проходится в направлении возрастания значения;

3. "убывает(K 0)" - в системе координат K кривая проходится по часовой стрелке;

4. "возрастает(K 0)" - в системе координат K кривая проходится против часовой стрелки.

Для параметрического задания ориентированной кривой с помощью числовой вектор-функции f от числового аргумента (параметра) в системе координат K служит запись "оркривая(f K)". Выражение "отрезоклинии(P A B)", где P - неориентированная кривая; A, B - ее точки, - задает ориентированную кривую, проходимую вдоль P от A к B.

Координаты

В физических выкладках часто встречаются проекции векторных величин на оси координат. Они обычно обозначаются с помощью индекса, указывающего нужную ось: A_x,F_y,v_z, и т.п. В решателе для аналогичной цели использована запись "крд(A K i)". Здесь A - рассматриваемый трехмерный вектор; K - обозначение системы координат; i - номер выделяемой координаты (1 - OX, 2 - OY, 3 - OZ). Заметим, что в аналитической геометрии было введено обозначение "коорд(A K)", дающее весь координатный набор. Обозначение для отдельного разряда такого набора оказалось востребовано лишь в задачах по физике.

Утверждение "вверх(A K)" означает, что вектор A в трехмерной системе координат K направлен вертикально вверх. Используется также аналогичное утверждение "вниз(A K)".

В физических задачах часто предполагается по умолчанию, что процессы происходят вблизи поверхности земли или какой-либо иной планеты. Чтобы явно обозначить это обстоятельство в посылках задачи, применяется утверждение "поверхнземли(K)". Одновременно оно фиксирует какую-то прямоугольную систему координат K, плоскость OXY которой совпадает с плоскостью поверхности планеты. Можно было бы сказать точнее - касается этой поверхности, однако обычно в указанных задачах локально поверхность допустимо считать плоской. Ось OZ системы K ведет вертикально вверх, т.е. от центра планеты. Если отсутствует дополнительная посылка "поверхнпланеты(K P)", уточняющая, какая планета P имеется в виду, то решатель будет считать, что рассматривается Земля, и брать соответствующее значение ускорения свободного падения.

Чтобы выделить в траектории движения верхнюю точку, служит запись "Верхняяточка(a K t)". Она означает, что в момент t тело, участвующее в процессе движения a, оказывается в точке с наибольшей высотой относительно системы координат K, причем этот момент - строго внутренний для процесса a. Если последнее требование отбрасывается, то используется запись "верхняяточка(A K t)". Аналогичным образом применяется запись "нижняяточка(A K t)".

Введены обозначения для характеризации направленности траекторий. Утверждение "вертикаль(A K)" означает, что траектория (т.е. ориентированная кривая) A состоит из отрезков, расположенных вертикально в прямоугольной системе координат K. Утверждение "одномерное(A K)" означает, что A есть процесс движения, траектория которого расположена на оси абсцисс прямоугольной трехмерной системы координат K. Утверждение "вертплоск(A K)" означает, что траектория процесса движения A состоит из частей, расположенных в плоскости OXZ системы K. Утверждение "горизонталь(A K)" означает, что траектория A состоит из отрезков, расположенных в плоскости OXY. Утверждение "горизплоск(A K)" относится к процессу движения A, траектория которого удовлетворяет аналогичному условию.

Чтобы работать не с материальной точкой, а с телом, имеющим невырожденные размеры, используется выражение "Крд(A K i)". Его значением служит множество i - х координат в системе K геометрических точек либо векторов, принадлежащих множеству A. Утверждение "вертикположение(A t K)" означает, что тело, совершающее процесс движения A, занимает в момент времени t вертикальное положение относительно системы координат K. Аналогичным образом вводится утверждение "горизположение(A t K)". Смысл каждого из приведенных утверждений уточняется согласно контексту задачи. Утверждение "горизповерхн(A K)" означает, что A есть процесс движения тела, верхняя сторона которого в системе координат K представляет собой горизонтальную поверхность. Высота в момент t горизонтальной поверхности тела, совершающего процесс движения A, относительно системы координат K, обозначается "высотаповерхн(A t K)". Утверждение "уклонвверх(P K t)" относится к процессу P движения клина $ABCD$. В момент t этот клин расположен так, что его треугольное основание ABC лежит в плоскости XOZ, причем вектор AB однонаправлен с координатным вектором OX, вектор AC образует с ним острый угол, а точка C лежит выше точки B.

Кинематика

Перейдем к логическим обозначениям, относящимся к кинематической составляющей описания сценария. Напомним, что приводимые далее понятия не дают сколь-нибудь полной и систематической картины, а просто перечисляют те предварительные версии формализации, которые были извлечены из проработанных задач.

Утверждение "движение(A B C)" означает, что A есть процесс движения материальной точки B по траектории C. Для указания фиктивного процесса движения A неподвижной в рассматриваемой системе отсчета материальной точки либо неподвижного тела B служит запись "неподв(A B)". Утверждение "движется(A B)" означает, что A,B суть процессы движения материальных точек, причем относительная скорость их не равна нулю в моменты, отличные от исходного и последнего. Утверждение "мточка(A)" означает, что A есть материальная точка.

Для обозначения скалярной скорости используется запись "скорость(A D)". Здесь A - процесс движения; D - либо момент времени, либо участок траектории процесса A (обычно вся траектория). В первом случае берется величина мгновенной скорости тела; во втором - средняя скалярная скорость на участке D. Заметим, что скалярная скорость всегда неотрицательна. Обычно она используется в задачах, где по умолчанию фиксирован неподвижный фон, относительно которого и определяется величина скорости. Можно считать, что роль такого фона играет неподвижная траектория процесса движения. Векторная скорость обозначается "Скорость(A B D)". Здесь A - процесс движения; B - другой процесс движения либо система координат, относительно которых рассматривается вектор скорости первого процесса. Если D - момент времени, то берется мгновенная скорость; если D - временной промежуток, то берется средняя скорость на данном промежутке. Для случая скалярных скоростей также предусмотрено понятие относительной скорости. Именно, посредством "относитскорость(D A B)" обозначается скорость объекта, находящегося в процессе движения A, относительно объекта, находящегося в процессе движения B. При этом предполагается, что траектории процессов A,B совпадают либо противоположны. Относительная скорость отсчитывается вдоль траектории и может быть любого знака. При этом положительное направление отсчета соответствует движению вдоль траектории B. D - либо момент времени, либо траектория процесса движения B. В последнем случае имеется в виду средняя скорость.

Расстояние, пройденноое в процессе движения A к моменту времени t, обозначается "Путь(A t)". Так же обозначается расстояние, пройденное за временной отрезок t. Утверждение "встреча(A B t P)" означает, что A, B суть два процесса движения, для которых в момент t в точке P происходит встреча. Утверждение "прохождение(A B C)" означает, что A есть процесс прохождения одного длинномерного объекта вдоль другого при движении обоих объектов по общей траектории C. B - неупорядоченная пара данных объектов. Утверждение "отрезокдвижения(A B C)" означает, что A есть подпроцесс процесса движения B, начинающийся с того момента, когда пройден начальный участок C траектории процесса B. Подпроцесс A может заканчиваться раньше, чем процесс B. Условие "невстр(A B)" означает, что объекты, участвующие в процессах движения A, B, не встречаются между собой.

Геометрическая точка, в которой располагается в момент времени t материальная точка, совершающая процесс движения A, обозначается "Место(A t)". Множество точек пространства, в которых находятся в момент времени t материальные точки тела, совершающего процесс движения A, обозначается "место(A t)".

Утверждение "равномерное(A)" означает, что A есть процесс движения, скаляр-

ная скорость которого неизменна. В частности, к этой категории относятся процессы равномерного вращения. Если движение происходит с постоянной векторной скоростью, то используется условие "Равномерное(A)".

Скалярное ускорение, т.е. производная по времени скалярной скорости, обозначается "ускорение(A t)". Здесь A - процесс движения; t - либо момент времени, к которому относится ускорение, либо участок траектории, на котором берется среднее значение ускорения. Векторное ускорение обозначается "Ускорение(A B t)". Здесь A - процесс движения, для которого определяется ускорение; B - другой процесс движения либо система координат, относительно которых рассматривается A. t - момент времени, либо временной промежуток, на котором выполняется осреднение ускорения. Утверждение "равноускоренное(A)" означает, что процесс движения A имеет постоянное скалярное ускорение; утверждение "Равноускоренное(A K)" - что он имеет постоянное векторное ускорение в системе координат K.

Утверждение "бросок(A)" означает, что A есть движение тела, на которое действует единственная сила - сила тяжести. Иногда используется утверждение "около(A K)", означающее, что процесс движения A происходит в непосредственной близости от начала системы координат K. Чтобы обозначить ускорение свободного падения у поверхности планеты A, явно указанной в задаче, используется запись "ускорсвобпад(A)".

Утверждение "плавныйпереход(A B)" означает, что A, B суть два последовательных процесса движения некоторого тела, у которых скорости в момент перехода одинаковы. Утверждение "отражение(A B C)" означает, что процесс движения B продолжает процесс A после упругого отражения относительно поверхности (в двумерной модели - линии) C.

Утверждение "напрдвижение(A B)" означает, что векторная скорость процесса движения A постоянно направлена вдоль прямой, соединяющей текущие точки процессов A, B. Утверждение "подвеска(A B)" означает, что A есть процесс движения тела, подвешенного на нерастяжимой нити к телу, находящемуся в процессе движения B. Утверждение "гибкаясвязь(A B C)" означает, что тела, находящиеся в процессах движения A и B, связаны тонкой нерастяжимой нитью C. Аналогичное утверждение "твердаясвязь(A B C)" относится к тонкому твердому стержню C. В утверждении "упругсвязь(A B C)", означающем наличие упругой связи (например, пружины) между телами, совершающими процессы движения A, B, в качестве C берется уже не само упругое соединение, а процесс его движения. Это вызвано необходимостью учитывать изменение внутренней энергии упругой связи. Утверждение "упругподвеска(A B C)" отличается от предыдущего лишь дополнительной информацией о том, что A подвешено к B на упругом соединении. Утверждение "блок(A B C)" означает, что A - процесс движения блока; B,C - процессы движения двух тел, соединенных перекинутой через блок тонкой нерастяжимой нитью. Утверждение "Подвеска(A B)" используется для указания на то, что A - процесс движения материальной точки, подвешенной с помощью системы нерастяжимых нитей к системе материальных точек, процессы движения которых образуют множество B.

Для движения тела по наклонной плоскости введена запись "движнаклплоск(A B K p)". Здесь A - процесс движения тела; B - процесс движения наклонной плоскости, по которой движется тело. K - прямоугольная система координат в пространстве, размещенная так, что наклонная плоскость параллельна оси OY и наклонена под углом p к направлению оси абсцисс. Этот угол может изменяться от 0 до $\pi/2$. Данная запись может показаться неоправданно громоздкой - вместо нее можно было бы использовать более простые утверждения. Одно из них определяло бы

параметры наклонной плоскости как геометрического тела; другое - указывало бы, что тело движется по поверхности наклонной плоскости. Однако, тогда было бы необходимо еще и третье утверждение, уточняющее, что тело перемещается по линии наибольшего уклона. Все это вместе взятое, все же, оказалось бы менее удобным, чем указанная выше запись.

Чтобы можно было рассматривать движение тела по наклонной плоскости, наклон которой изменяется с течением времени, введена запись "наклон(A B K p t)". Она аналогична записи "движнаклплоск(...)", однако дополнительно указывается момент времени t, для которого угол наклона равен p. Этот угол может изменяться от 0 до двух пи.

Утверждение "движениетела(A B)" означает, что A есть процесс движения тела B. Для обозначения фиктивного процесса A движения неподвижного в рассматриваемой системе отсчета тела B служит запись "неподвтело(A B)". Утверждение "лежитна(A B C)" означает, что A есть процесс движения материальной точки, лежащей на материальной точке, совершающей процесс движения B и относящейся к телу, совершающему процесс движения C. Утверждение "Лежитна(A B t)" означает, что в момент t тело, совершающее процесс движения A, лежит на верхней стороне тела, совершающего процесс движения B. Утверждение "движениепо(A B)" означает, что A есть процесс движения материальной точки по поверхности тела, совершающего процесс движения B. Утверждения "внутрповерхн(A B)", "внешповерхн(A B)" означают, что A есть процесс движения материальной точки, соответственно, по внутренней либо внешней поверхности тела, совершающего процесс движения B. Смысл понятий "внутренняя" и "внешняя" уточняется контекстом. Утверждение "моментотрыва(A B t)" означает, что t есть момент отрыва тела, совершающего процесс движения A по поверхности тела, совершающего процесс движения B, от последнего.

Утверждение "попадание(A B)" означает, что A, B суть такие два процесса движения тел, имеющие общий период, что в конце этого периода тело A попадает на тело B, а скорости тел становятся равны. Утверждение "распадение(A B)" означает, что B есть множество процессов движения составных частей тела, находившегося до этого в процессе движения A. Начальные моменты всех процессов из B совпадают с конечным моментом процесса A. Множество геометрических точек, являющихся в момент t точками соприкосновения тел, совершающих процессы движения A и B, обозначается "общиеточки(A B t)".

Для указания на то, что A есть процесс движения материальной точки, относящейся к телу, совершающему процесс движения B, служит утверждение "движениеточки(A B)".

Чтобы указать на то, что процесс A движения тела B является поступательным, используется запись "поступатдвижение(A B)". Утверждение "наклплоск(A K p)" означает, что A есть процесс поступательного движения поверхности, образующей в каждый момент наклонную плоскость, расположенную относительно прямоугольной системы координат K так, что она параллельна оси OY и наклонена под углом p к направлению оси абсцисс. Параметр p изменяется от 0 до $\pi/2$.

Утверждение "вращение(A B)" означает, что A есть процесс вращательного движения тела B. Расстояние, пройденное в процессе вращения A материальной точкой B вращающегося тела, либо расстояние, пройденное точкой на внешней окружности вращающегося круглого тела B, обозначается "путьвращения(A B)". Угол, на который поворачивается тело, находящееся в процессе вращения A, к моменту t, обозначается "уголвращения(A t)". Утверждение "центрвращения(A B)" означает, что

A есть процесс движения точки - центра вращательного движения B. Утверждение "точкавращения(A B)" означает, что A, B суть процессы движения двух материальных точек, относящихся к различным телам, скрепленным в этих точках между собой так, что они могут свободно вращаться относительно точки соединения. Расстояние от центра процесса вращения A до точки B вращающегося тела обозначается "радиусвращения(A B)".

Утверждение "полныйоборот(A)" означает, что процесс движения A представляет собой однократное прохождение материальной точкой некоторой окружности.

Условная внешняя окружность круглого тела A (колесо и т.п.), рассматриваемая как множество материальных точек этого тела, обозначается "внешокружность(A)".

Утверждение "качение(A B C)" означает, что A есть процесс качения тела B; C - траектория, описываемая точкой качения. Геометрическая точка, в которой тело, совершающее процесс качения A, соприкасается в момент t с поверхностью качения, обозначается "точкакачения(A t)".

Среднее число оборотов в единицу времени для процесса A вращательного движения тела B, либо мгновенная скорость вращения (обороты в единицу времени) для процесса A в момент B обозначается "скоростьвращения(A B)". Выражение "углскорость(A B)" обозначает либо угловую скорость в момент B тела, находящегося в процессе движения A, либо среднюю угловую скорость на участке траектории B указанного тела, либо угловую скорость тела B, находящегося в процессе вращения A. Выражение "Углскорость(A B T)" обозначает угловую скорость вращения вектора, направленного от точки, совершающей процесс движения A, к точке, совершающей процесс движения B. T - момент либо период, в который измеряется скорость. Число оборотов, сделанных телом в течение процесса вращения A, обозначается "числооборотов(A)". Выражение "периодобращения(A)" обозначает период обращения процесса вращательного движения A.

Утверждение "ременнаяпередача(A B)" означает, что тела, находящиеся в процессах A и B вращательного движения, соединены ременной передачей. Утверждение "вращблока(A B C D)" означает, что A есть процесс вращения шкива, обеспечивающего связь через тонкую нерастяжимую нить процессов движения B,C,D. Центр шкива отождествляется с материальной точкой B.

Приведем несколько понятий, встречающихся в задачах на вращение спутников. Утверждение "планета(A)" означает, что тело A является планетой. Утверждение "спутник(A B h)" означает, что A есть процесс движения некоторого тела относительно другого шарообразного тела, находящегося в процессе движения B, по круговой орбите под действием только силы взаимного притяжения. h - высота орбиты относительно поверхности второго тела. Утверждение "полюс(A B)" означает, что материальная точка A является полюсом планеты B. Утверждение "экватор(A B)" означает, что материальная точка A лежит на экваторе планеты B.

Утверждение "течение(A B)" означает, что A есть процесс протекания некоторого потока по руслу B. Предполагается, что русло имеет направляющую - линию, в каждой точке которой задано поперечное сечение русла. Как и траектория, русло ориентировано - выделены его начало и конец. Начальная и конечная точки направляющей линии русла A обозначаются "исток(A)". "сток(A)". Поперечное сечение русла A в точке B его направляющей линии обозначается "сечениерусла(A B)". Утверждение "простоерусло(A)" означает, что русло A имеет постоянную площадь сечения. Для указания на то, что траектория A движения материальной точки направлена вдоль русла B, начинаясь в начале этого русла и заканчиваясь в его конце, используется запись "вдольрусла(A B)". Утверждение "стекает(A B)" означает, что

A есть процесс течения вдоль поверхности твердого тела, совершающего процесс движения B.

Содержимое русла процесса течения A в момент t обозначается "струя(A t)". Плотность вещества процесса течения A в точке B направляющей его русла в момент t обозначается "плотностьтечения(A B t)". Если B - не точка, а ссылка на все русло, то берется средняя плотность вещества, находящегося в струе в момент t. Если t - не момент времени, а ссылка на русло, то берется средняя плотность вещества (в точке B либо по всему руслу - в зависимости от B) на протяжении всего процесса A.

Утверждение "истечение(A B)" означает, что A есть процесс истечения из источника B либо вытекания через сток B некоторой субстанции. Величина потока (масса в единицу времени) для процесса истечения A в момент времени t обозначается "поток(A t)". Если t - не момент, а источник (сток) процесса A, то берется средняя величина потока за период процесса A. Для обозначения величины потока как объема в единицу времени аналогичным образом используется запись "Поток(A t)". Выражение "вкладисточника(A t)" обозначает суммарную массу вещества, добавленную в процессе источения A на момент времени t. Если источник является стоком, то эта величина отрицательна. Вместо момента времени можно брать в качестве t источник (сток); тогда получается обозначение для суммарной массы вещества за весь период процесса A. Выражение "Вкладисточника(A t)" аналогичным образом используется для указания суммарного объема вещества.

Динамика

Для указания процесса A силового воздействия объекта, совершающего процесс движения C, на объект, находящийся в процессе движения B, используется запись "воздействие(A B C)". Вектор силы, характеризующий процесс силового воздействия A в момент времени t, обозначается "сила(A t)". Если t - временной промежуток, то берется среднее значение вектора силы. Вектор равнодействующей всех сил, действующих в момент либо промежуток времени t на материальную точку, совершающую процесс движения A, обозначается "Сила(A t)". Утверждение "Силы(A B)" означает, что B есть множество всех процессов силового воздействия на материальную точку, находящуюся в процессе движения A.

Утверждение "констсила(A)" означает, что процесс силового воздействия A характеризуется не изменяющимся в течение его периода вектором силы. Аналогично, утверждение "констсилы(A)" выделяет множество A процессов силового воздействия с не изменяющимся вектором силы.

Если на тело могут действовать не указанные в задаче явно внешние силы, то добавляется посылка "внештело(A)", где A - процесс движения данного тела.

Утверждение "притяжение(A)" означает, что процесс A силового воздействия есть процесс притяжения. Утверждение "торможение(A)" означает, что A есть процесс движения, для которого вектор силы постоянно направлен противоположно вектору скорости. Утверждение "тянет(A B C)" означает, что A есть процесс силового воздействия объекта, совершающего процесс движения B, на объект, совершающий процесс движения C, при котором вектор силы направлен вдоль траектории движения C по направлению к объекту B.

Вектор силы трения, характеризующий процесс A силового воздействия в момент времени t либо на промежутке t, обозначается "силатрения(A t)". Вектор силы нормальной реакции при этом обозначается "нормреакция(A t)". Коэффициент трения (в тех же предположениях) обозначается "коэффтрения(A t)". Если A,B суть процес-

сы движения тел, первое из которых перемещается по поверхности второго, причем коэффициент трения равен 0, то используется запись "гладкое(A B)". Утверждение "граньскольжения(A B t)" означает, что A и B суть процессы движения двух тел, одно из которых в момент либо период t лежит на поверхности другого и находится на грани скольжения. Утверждение "безпроскальзывания(A B t)" означает, что A есть процесс движения материальной точки по поверхности тела, совершающего процесс движения B, причем в момент t это движение происходит без проскальзывания относительно кривой, получающейся при пересечении поверхности тела B с нормальной плоскостью к траектории A. Иными словами, проекция на касательную к этой кривой суммарного вектора сил, действующих на точку A, рассматриваемого в системе координат, связанной с телом B, равна 0. Утверждение "граньпроскальз(A B)" означает, что A,B суть процессы движения двух тел, первое из которых перемещается по поверхности второго, находясь на грани проскальзывания в направлении, обратном направлению к центру кривизны.

Утверждение "воздействия(A t B)" означает, что B есть множество всех силовых воздействий на точки совершающего процесс движения A тела в момент либо промежуток времени t. Утверждение "центртяжести(A B)" означает, что условная материальная точка A является центром тяжести тела B. Чтобы выделить центр A притяжения планеты, рассматриваемой в задаче, используется также утверждение "центрземли(A)". Величина первой космической скорости для планеты A обозначена "первкосмскор(A)".

Коэффициент жесткости упругого соединения A обозначается "жесткость(A)". Величина удлинения либо сжатия в момент либо период t упругого соединения, совершающего процесс движения A обозначается "удлинсвязи(A t)". Длина упругой связи A в отсутствии внешних сил обозначена "исхдлина(A)".

Утверждение "давление(A B C)" означает, что A есть процесс силового воздействия объекта, находящегося в процессе движения B, на субстанцию, находящуюся в процессе течения C. Предполагается, что воздействие имеет место в начальной точке русла процесса C.

Закон сохранения импульса

Вектор импульса тела, находящегося в процессе движения A, который берется относительно системы координат K в момент либо временной промежуток t, обозначен "импульс(A K t)".

Утверждение "отскок(A B C D)" означает, что A есть процесс отскока тела, совершающего процесс движения C, от тела, совершающего процесс движения B. D - процесс движения тела C после отскока. Считаем, что при отскоке движение B не изменяется. Утверждение "соударение(A B C D E)" означает, что A есть процесс соударения тел, совершающих процессы движения B,C. D,E - процессы движения этих же тел после соударения. Утверждения "упругий(A)". "неупругий(A)" означают, что процесс A отскока или соударения является абсолютно упругим либо абсолютно неупругим. Утверждение "центральнудар(A)" означает, что процесс соударения A является центральным, т.е. скорости направлены вдоль линии, соединяющей центры тяжести соударяющихся тел. Утверждение "гладкийудар(A)" означает, что в процессе соударения A участвуют тела с гладкими поверхностями, т.е. скорости этих тел в направлении, перпендикулярном оси удара, сохраняются.

Утверждение "падение(A B C D)" означает, что A есть процесс падения тела, совершающего процесс движения C, на тело, совершающее процесс движения B.

D - процесс движения тела C после падения. Движение B после падения остается неизменным.

Утверждение "толчок(A B)" означает, что A есть процесс движения тела, происходящий при силовом воздействии B и отсутствии других силовых воздействий.

Утверждение "выброс(A B)" означает, что A есть процесс движения тела, от которого в течение периода A отделяются тела, множество процессов движения которых есть B, причем силовыми воздействиями на тела B, отличными от воздействия на них тела A, можно пренебречь. Считается, что A и любой из процессов B относятся к непересекающимся телам, которые на начальный момент их общего периода составляли одно тело.

Утверждение "наезд(A B C D E F G)" означает, что A есть процесс наезда тела, совершающего процесс движения B, на поверхность тела, совершающего процесс движения C. Оба тела до наезда движутся по поверхности тела, совершающего процесс движения F. D,E,G - процессы движения тел B,C,F после наезда. Предполагается, что скорость процесса G после наезда совпадает со скоростью процесса F до наезда. Утверждение "плавныйнаезд(A)" означает, что процесс наезда A протекает без удара.

Утверждение "ударструи(A B C D)" означает, что A есть процесс удара струи, процесс течения которой есть C, о твердое тело, процесс движения которого есть B. D - процесс течения струи после удара.

Утверждение "замкнсистема(A M K)" означает, что A есть множество процессов движения тел, имеющих общий период, причем в течение этого периода сумма i-х координат сил, действующих на тела системы, для каждого $i = 1, 2, 3$, кроме, быть может, значений i, принадлежащих множеству M, равна 0. K - система координат. Утверждение "замкнапр(A B)" означает, что A есть множество процессов движения тел, имеющих общий период, причем в течение этого периода сумма проекций на направление вектора B сил, действующих на тела системы, равна 0.

Работа и энергия

Работа, совершаемая силовым воздействием A за период времени t, обозначается "работасилы(A t)". Мощность, потребляемая процессом A в момент t, либо средняя мощность, потребляемая им за период t, обозначается "потреблмощность(A t)". Мощность, затрачиваемая процессом A на процесс B в момент либо период t, обозначается "полезнмощность(A B t)". Коэффициент полезного действия для мощности, затрачиваемой процессом A на процесс B в момент либо период t, обозначается "кпд(A B t)".

Кинетическая энергия тела, совершающего процесс движения A, в момент t, обозначается "кинетичэнергия(A t)". Потенциальная энергия, при тех же предположениях, обозначается "потенцэнергия(A t)". Потенциальная энергия тяготения тела, совершающего процесс движения A, обозначается "потенцэнергтягот(A t)"; потенциальная энергия деформации того же тела (пружины, и т.п.) обозначается "потенцэнергдеформ(A t)". Полная механическая энергия обозначается "мехэнергия(A t)". Энергия, выделившаяся в виде тепла в течение процесса A, обозначается "теплэнергия(A)".

Чтобы указать на то, что B есть процесс движения, непосредственно продолжающий процесс движения A, причем механическая энергия тела на стыке процессов не изменяется, используется утверждение "продолжбезудара(A B)".

Глава 3

Представление задач в решателе

3.1 Структуры данных, используемые для представления задачи

Как уже говорилось в главе 1, в решателе рассматриваются задачи четырех типов - на доказательство, на описание, на преобразование и на исследование. Начнем с описания используемых для их представления структур данных. Задача Z любого типа представлена набором $(p_1, \ldots, p_n)$, первый элемент p_1 которого есть логический символ - название типа задачи. Эти названия суть логические символы "доказать", "описать", "преобразовать"и "исследовать". p_2 есть набор $(f_1, \ldots, f_m)$ утверждений, называемых посылками задачи. Посылки перечисляют априори известные ограничения на фигурирующие в задаче объекты - то, что считается в задаче "данным". Список посылок обязательно непуст; если никаких исходных допущений не делается, то он предполагается состоящим из логической константы "истина". p_3 есть набор $(a_1, \ldots, a_m)$ целых неотрицательных чисел, называемых весами посылок (a_i - вес посылки f_i) и используемых для переключения внимания при поиске очередного приема; в исходной ситуации веса посылок равны 0. p_4 - набор $(K_1, \ldots, K_m, K_0)$, элементы K_i которого при $i \neq 0$ суть наборы комментариев к i -й посылке, а K_0 есть набор комментариев ко всему списку посылок. Комментарии суть технические пометки, делаемые решателем в процессе работы для сохранения различной информации, направляющей ход решения. В исходной ситуации все K_i пусты. Компоненты p_1 - p_4 являются общими для задач всех перечисленных выше типов. Прочие компоненты определяются в зависимости от типа задачи следующим образом:

1. Задачи на доказательство. В этом случае $n = 7$. Элемент p_5 представляет собой утверждение, называемое условием задачи. Его истинность и требуется доказать, предполагая истинность посылок. p_6 - целое неотрицательное число, называемое весом условия задачи; как и веса посылок, оно используется для переключения внимания в процессе решения задачи. p_7 - последний элемент задачи на доказательство - набор комментариев к задаче в целом. Они, как и комментарии к посылкам, представляют собой технические пометки, вводимые решателем в процессе работы. Решая задачу на доказательство, решатель может выдать в качестве ответа либо логический символ "истина если он убедится, что условие действительно является логическим следствием посылок, либо логический символ "отказ если его попытки останутся безуспешными. Заметим, что решатель при рассмотрении задачи на доказательство не выдает сообщений о ложности условия либо о том, что оно не является следствием посылок (хотя для ускоренной выдачи отказа он и может по-

пытаться установить эти обстоятельства). Задача на доказательство служит только для "односторонней"попытки установления истинности утверждения - именно в такого рода вспомогательных попытках обычно и возникает надобность при решении других задач. Если же нужно провести "двустороннюю"проверку - выяснить, является ли некоторое утверждение истинным либо ложным, то для этого используется задача на описание с пустым списком неизвестных (подробнее об этом см. ниже).

2. Задачи на описание. Здесь $n = 9$. В виде задачи на описание оформляются различные задачи, у которых требуется преобразовать заданным образом некоторый список утверждений - "условий" задачи. Преобразования этого списка условий осуществляются в предположении истинности списка посылок. Как правило, условия содержат неизвестные, и в задаче требуется получить явное описание (полное либо неполное - в зависимости от целевой установки задачи) допустимых значений таких неизвестных. Элемент p_5 задачи на описание представляет собой список $(g_1, \ldots, g_k)$ ее условий. Элемент p_6 - набор $(b_1 \ldots, b_k)$ весов условий; элемент p_7 - набор $(Q_1, \ldots, Q_k, Q_0)$ списков комментариев. Здесь Q_i $(i = 1, \ldots, k)$ представляет собой список комментариев к условию g_i; Q_0 - список комментариев ко всей задаче в целом. Элементы p_5, p_6, p_7 совершенно аналогичны элементам p_2, p_3, p_4, характеризующим посылки задачи. Элемент p_8 представляет собой набор технических пометок, называемых целями задачи. Ответ, получаемый в процессе решения задачи на описание, постепенно складывается в списке ее условий, так что при успешном завершении преобразований в конце решения выдается группа соединенных логической связкой "и" заключительных условий. При неудаче в качестве ответа выдается логический символ "отказ". Класс утверждений, являющихся ответами на задачу, определяется в зависимости от набора ее целей. При этом некоторые из целей могут не накладывать каких-либо явных ограничений на ответ, а лишь указывать те или иные установки на оптимизацию, влияющие на принятие решений в процессе преобразований. Фактически, список целей задачи представляет собой установку на управление базой приемов решателя, осуществляющей рассмотрение задачи. За исключением особых случаев, принимается ряд соглашений о связи ответа задачи на описание с ее исходными условиями и посылками. Обычно задача на описание имеет цель, указывающую множество ее неизвестных, причем вхождение этих неизвестных в посылки задачи не допускается. Условия задачи должны являться следствиями ответа и посылок задачи; при этом ответ можно рассматривать как частичное либо полное описание множества наборов значений неизвестных, удовлетворяющих условиям задачи.

Достаточно часто при решении задачи на описание может оказаться полезным и даже необходимым рассмотрение совместных следствий условий и посылок. Так как ответ задачи извлекается из списка ее условий, то регистрация таких вспомогательных следствий в списке условий нежелательна - она приведет к чрезмерному его увеличению и потребует сложной процедуры "расчистки" условий после определения значений неизвестных. Поэтому в структуре задачи на описание пришлось ввести специальный накопитель совместных следствий условий и посылок. Роль этого накопителя (элемент p_9) играет вспомогательная задача на исследование, называемая блоком анализа задачи на описание. Изначально блок анализа отсутствует и вводится решателем в процессе рассмотрения задачи. В исходной ситуации элемент p_9 может быть равен либо логическому символу "пустоеслово"(общий случай), либо, в особых случаях, логическому символу "0" - последнее означает запрет на ввод блока анализа при решении.

3. Задачи на преобразование. В этом случае $n = 8$. p_5 - выражение, называемое условием задачи. Это выражение нужно преобразовать к некоторому виду, определя-

емому целевой установкой задачи. Как правило, требуется, чтобы исходное и результирующее выражения были равны, в предположении истинности списка посылок. Иногда это требование может нарушаться; самый простой пример такого рода - использование задачи на преобразование для получения асимптотической оценки, когда результирующее выражение лишь асимптотически равно исходному. Более того, из-за технических причин в некоторых случаях удобно применять вспомогательные задачи на преобразование, условиями которых являются не выражения, а утверждения. p_6 - целое неотрицательное число, называемое весом условия; p_7 - набор комментариев к задаче. p_8 - набор целей задачи. Некоторые из целей накладывают ограничения на вид ответа; другие - определяют установки на оптимизацию ответа. Ответом задачи на преобразование служит либо результирующее ее условие - если оно удовлетворяет определяемым целями ограничениям, либо логический символ "отказ".

4. Задачи на исследование. В этом случае $n = 5$. p_5 - набор целей задачи. Эти цели управляют процессом вывода представляющих интерес следствий из посылок задачи, упрощения посылок и удаления избыточных посылок. В результате таких действий исходная задача на исследование Z преобразуется в некоторую новую задачу Z', и при исчерпании средств, отведенных для решения задачи Z, в качестве ответа на нее выдается задача Z'. По существу, такая выдача ответа является фикцией - в действительности внешняя процедура, обратившаяся к решению задачи на исследование, обычно имеет непосредственную ссылку на нее, и по этой ссылке способна получать всю необходимую информацию о произошедших изменениях.

В исходной ситуации веса посылок и условий задачи, предъявляемой решателю, равны 0; блок анализа и комментарии к задаче (ее посылкам, условиям) отсутствуют. Вспомогательные задачи, вводимые решателем, могут иметь ненулевые исходные веса посылок и непустые списки комментариев. Это обеспечивает необходимую преемственность - сохраняет ранее накопленную информацию, которая может понадобиться при решении вспомогательной задачи. Изменение весов, формирование комментариев и блока анализа осуществляются системой в процессе решения задачи.

3.2 Целевые установки задач

Целевая установка задачи определяется ее списком целей. Она имеется у всех типов задач, за исключением задач на доказательство. С технической точки зрения, целевая установка обеспечивает управление поведением многих тысяч независимо функционирующих приемов, решающих задачу. Так как каждый прием рассматривает в своих решающих правилах лишь сравнительно небольшой фрагмент целевой установки, и достижение ответа происходит в результате интегрального действия многих различных приемов, то варьирование списка целей задачи предоставляет достаточно большие степени свободы для управления коллективным поведением приемов. Пополнение списка возможных типов целей задач осуществляется, как правило, в связи с рассмотрением конкретной предметной области и не приводит к какой-либо модификации приемов, относящихся к другим предметным областям.

Перечислим некоторые наиболее часто встречающиеся типы целей, а в следующем подразделе данной главы на большом количестве примеров проиллюстрируем способы формального представления задач в конкретных областях.

Заметим, что цель задачи обычно представляется либо логическим символом, либо набором $(\phi, A_1, \ldots, A_m)$, где ϕ - логический символ, называемый заголовком

цели; $A_1, \ldots, A_m$ - некоторые объекты. В последнем случае указанный набор будем записывать далее как "$\phi A_1 \ldots A_m$" .

Пусть сначала Z - задача на описание. Если Z имеет цель "неизвестные $x_1 \ldots x_k$" , где $x_1, \ldots, x_k$ - попарно различные переменные, то $x_1, \ldots, x_k$ называются неизвестными задачи Z. Прочие свободные переменные посылок и условий называются параметрами задачи. Задача на описание может и не иметь неизвестных, причем в зависимости от прочих целей в этом случае необходимо либо установить эквивалентность условий одной из логических констант "истина" , "ложь" , либо осуществить определенную переформулировку условий. Истинность или ложность посылок задачи Z должна однозначно определяться при указании значений всех их свободных переменных, отличных от неизвестных (как правило, неизвестные задачи Z вообще не встречаются в посылках, хотя допускается фиктивное в указанном смысле появление их в качестве свободных переменных посылок). Если Z имеет цель "параметры $y_1 \ldots y_k$" , где $y_1, \ldots, y_k$ - некоторые из неизвестных задачи, то переменные $y_1, \ldots, y_k$ называются несущественными неизвестными задачи Z. Несущественные неизвестные задачи представляют собой те из неизвестных, для которых требуется установить лишь сам факт существования удовлетворяющих условиям их значений, не находя эти значения явным образом. Более точно, произвольный ответ на задачу Z представляет собой такое утверждение f, что для любых значений свободных переменных посылок задачи Z, при которых эти посылки истинны, из истинности f вытекает существование таких значений не входящих в f несущественных неизвестных задачи, что все условия задачи Z истинны. Несущественные неизвестные задачи являются "исключаемыми"неизвестными и обычно не входят в ответ. Если задача имеет цель "прямойответ", то ответ f не может содержать вспомогательных переменных, вводимых для обозначения объектов, существование которых вытекает из истинности посылок задачи. В этом случае описание допустимых значений неизвестных осуществляется только в терминах объектов, упоминавшихся в посылках и условиях задачи.

Если задача Z имеет цель "пример", то ответ f представляет собой утверждение, построенное при помощи логических символов "и" , "или" из утверждений вида "равно(x t)" , где x - неизвестная задачи Z; t - выражение, не зависящее от неизвестных, а также из некоторых утверждений, не зависящих от неизвестных задачи Z. При преобразовании f к виду дизъюнктивной нормальной формы ни в одну из элементарных конъюнкций не входят два различных утверждения вида "равно(x t)" ,соответствующих одной и той же неизвестной x. Данная цель используется в тех случаях, когда требуется указать конкретный набор значений неизвестных задачи, при которых истинны ее условия, быть может, с дополнительными ограничениями на объекты, упоминаемые в посылках задачи. Если такие ограничения недопустимы, то вводится дополнительная цель "полный". Более подробно, если задача Z имеет цели "полный" , "пример" , то при замене всех входящих в ответ f утверждений вида "равно(x t)" , где x - неизвестная задачи, на логический символ "истина", должно получаться такое утверждение f', являющееся следствием посылок задачи. Если же задача Z, имеющая цель "полный" , не имеет цели "пример" , то из истинности посылок этой задачи вытекает эквивалентность истинности ответа f существованию значений несущественных неизвестных, не являющихся свободными переменными f, при которых истинны все условия задачи Z.

Если задача Z имеет цель "явное" , то ее ответ f представляет собой утверждение, относящееся к классу так называемых явных описаний значений неизвестных этой задачи. Для определения класса явных описаний в каждой из рассматриваемых предметных областей выделяются семейства конечных множеств $\{f_1, \ldots, f_s\}$ утвер-

ждений, называемых атомарными описаниями переменной x; эта переменная является параметром каждого из утверждений $f_1, \ldots, f_s$. Так, атомарными описаниями переменной x считаются, например, множества $\{x = t_1\}$, $\{x \in t_1\}$, $\{t_1 < x, x < t_2\}$, $\{\exists k(k - \text{целое } \& \ x = \pi k)\}$, $\{x - \text{целое}\}$ и т.п. Здесь t_1, t_2 - произвольные выражения, не имеющие параметра x. Явными описаниями значений неизвестных задачи Z считаются утверждения, построенные при помощи логических символов "и", "или"из элементов атомарных описаний неизвестных этой задачи и не зависящих от неизвестных утверждений, причем такие, что в результате преобразования к виду дизъюнктивной нормальной формы каждая их элементарная конъюнкция, при подходящей группировке членов, приобретает вид "и($A_1(x_1)A_2(x_2)\ldots A_s(x_s)B$)", где для любого $i = 1, \ldots, s$ $A_i(x_i)$ - конъюнкция утверждений атомарного описания неизвестной x_i задачи Z; x_i не является параметром утверждений $A_{i+1}(x_{i+1}), \ldots, A_s(x_s)$; B - утверждение, не зависящее от неизвестных.

В некоторых случаях бывает полезной цель "и" , при наличии которой ответ f на задачу Z имеет вид "и(равно$(x_{j_1} t_1)$ ... равно$(x_{j_m}\ t_m)$ $\ g_1 \ldots\ g_s$)" , где $m + s - 1 \geq 1$; $x_{j_1}, \ldots, x_{j_m}$ - различные неизвестные задачи Z; $t_1, \ldots, t_m, g_1, \ldots, g_s$ не зависят от неизвестных задачи. Указанный вид ответа определяет подстановку выражений $t_1, \ldots, t_m$ вместо переменных $x_{j_1}, \ldots, x_{j_m}$; такая подстановка может использоваться, например, при решении системы уравнений путем выражения части ее неизвестных $x_{j_1}, \ldots, x_{j_m}$ через остальные неизвестные.

Для ограничения зависимости ответа от использованных при формулировке задачи переменных применяется цель "известно $y_1 \ldots y_k$" , где $y_1, \ldots, y_k$ - переменные, не являющиеся неизвестными задачи; $k \geq 0$. При наличии такой цели каждая свободная переменная ответа f представляет собой либо неизвестную задачи Z, либо некоторую переменную $y_i; i = 1, \ldots, k$. Переменные $y_1, \ldots, y_k$, выделенные целью указанного вида, называются исходными данными задачи.

В качестве примера цели, определяющей установку на оптимизацию, приведем цель "упростить" , активизирующую попытки редактирования найденного ответа путем разрешения условий на параметры относительно этих параметров, упрощения входящих в ответ выражений и упрощения логической структуры ответа.

При формулировке исходной задачи может быть использована цель "одз" , указывающая на необходимость решения задачи в области допустимых значений неизвестных и параметров. Эта цель, по сути дела, является лишь относящимся к интерфейсу системы техническим приемом, позволяющим в неявной форме задавать часть условий либо посылок задачи. Она инициирует расширение списков условий и посылок задачи рядом утверждений, характеризующих область допустимых значений переменных задачи, после чего удаляется.

Приведем несколько примеров часто встречающихся комбинаций целей задач на описание. Список { "неизвестные $x_1 \ldots x_k$" , "полный" , "пример" } встречается у задач Z, введенных при решении задачи на доказательство существования значений переменных $x_1, \ldots, x_k$, удовлетворяющих некоторому утверждению f. При этом в задаче Z требуется найти конкретные значения $x_1, \ldots, x_k$, быть может, выразив их через вспомогательные обозначения для объектов, существование которых вытекает из посылок. Последнее объясняет отсутствие цели "прямойответ", блокирующей приемы, вводящие такие вспомогательные обозначения.

Список { "неизвестные $x_1 \ldots x_k$" , "полный" , "явное" , "прямойответ" }, как правило, вместе с целью "упростить", встречается у задач, в которых требуется описать в явной форме все значения неизвестных, удовлетворяющие условиям (например, задачи на решение систем уравнений или неравенств; задачи на определение геомет-

рического места точек в геометрии, и т.п.). Если некоторые из условий такой задачи представляли собой утверждения о существовании некоторых объектов $y_1, \ldots, y_m$, то при ее решении возможно появление вспомогательных задач, неизвестными которых, помимо $x_1, \ldots, x_k$, становятся также $y_1, \ldots, y_m$. В этом случае вспомогательная задача имеет, наряду с перечисленными выше, цель "параметры $y_1 \ldots y_m$".

Список "неизвестные $x_1 \ldots x_k$" ,"полный" , "явное" , "прямойответ" , "известно $y_1 \ldots y_k$"возникает в тех случаях, когда значения неизвестных требуется выразить через исходные данные $y_1, \ldots, y_k$, представляющие собой, вообще говоря, лишь часть параметров задачи. Как правило, в этом случае список условий задачи имеет вид "равно(x_1 t_1)" , ..., "равно(x_k t_k)" , где $t_1, \ldots, t_k$ - выражения, не зависящие от неизвестных, но зависящие от некоторых параметров задачи, не являющихся исходными данными $y_1, \ldots, y_k$. Примером такого рода являются геометрические задачи на вычисление, посылки которых описывают некоторый геометрический чертеж; выделяются известные величины $y_1, \ldots, y_k$ (переменные), характеризующие этот чертеж, и требуется определить ряд связанных с ним неизвестных величин $t_1, \ldots, t_k$. Встречающиеся в описании чертежа переменные, обозначающие точки, хотя формально и являются известными, не должны входить в ответ задачи.

Если требуется проверить истинность утверждений при заданных посылках, то используется задача на описание, единственным условием которой служит данное утверждение, а цели суть "полный", "явное", "прямойответ"(обычно добавляется цель "одз"). Неизвестные задачи отсутствуют, и при решении ее будут предприниматься попытки заменить проверяемое утверждение на логическую константу "истина"либо "ложь". Эта константа и будет выдана в качестве ответа.

Цели задач на преобразование оказываются тесно связаны с конкретными предметными областями (например, разложение на множители либо преобразование к виду суммы одночленов в элементарной алгебре; приведение входящих в преобразуемое выражение тригонометрических функций к общему аргументу в тригонометрии; нахождение асимптотики выражения при отыскании пределов и т.п.). В одной задаче, как правило, сочетаются несколько таких целей (например, цель "разложитьнамножители"и цель "неизвестные $x_1 \ldots x_k$", указывающая, что разложение на множители выражения t необходимо для решения уравнения $t = 0$ относительно $x_1, \ldots, x_k$). Единственной целью сравнительно общего характера для задач на преобразование является цель "упростить", выражающая тенденции к определенной стандартизации выражения. Впрочем, понятие "упростить"трактуется приемами решателя эвристически - в контексте прочих обстоятельств, сопровождающих задачу, так что в различных случаях упрощение одного и того же выражения может дать различные результаты. С другой стороны, цель "упростить"заменяет собой большое число целей частного характера: задачи на нахождение численного значения выражения, на вычисление производной, предела, интеграла и т.п. могут формулироваться как задачи на преобразование, имеющие единственную цель "упростить". Как и в случае задачи на описание, допускается формулировка исходной задачи на преобразование с целью "одз", определяющей присоединение к списку посылок совокупности утверждений, описывающей область допустимых значений параметров.

Целевая установка задачи на исследование, возникающей в большинстве случаев как блок анализа задачи на описание, определяется спецификой этой внешней задачи. В такой целевой установке присутствует цель "неизвестные $x_1 \ldots x_k$, перенесенная из задачи на описание и направляющая вывод следствий таким образом, чтобы увеличить "степень разрешенности"утверждений относительно переменных $x_1, \ldots, x_k$. В случае задач на исследование, возникших при доказательстве утверждений "от про-

тивного", используется цель "противоречие", активизирующая попытки установить противоречивость посылок.

3.3 Примеры формулировки задач для решателя

Чтобы сделать более понятной приведенную выше общую схему формализации задач для решателя, приведем примеры постановки задач из различных разделов. Хотя в данном параграфе будет часто использоваться "скобочная"форма записи, соответствующая внутреннему логическому языку решателя, в действительности при постановке задачи и просмотре процесса ее решения применяется приближенная к стандартной математическая запись. Минимальные необходимые для ввода новых задач сведения об интерфейсе, позволяющем работать с решателем и использующем стандартную запись, будут приведены в следующем параграфе этой главы.

3.3.1 Алгебра множеств

Начнем с простейшего раздела - алгебры множеств. Если требуется доказать, например, истинность тождества $c \cup (b \setminus a) = (b \cup c) \setminus a$, предполагая истинными утверждения $a \subseteq b$ и $(b \cap c) = \emptyset$, то мы приходим к задаче на доказательство, имеющей посылки "множество(a)" , "множество(b)" , "множество(c)" ,"содержится(a b)", "равно(пересечение(b c)пусто)" и единственное условие "равно(объединение(c разность(b a)) разность(объединение(b c)a))". Заметим, что первые три посылки, указывающие тип значений переменных (множество), обычно формируются автоматически, и вручную их вводить не нужно. В последующих примерах будем для краткости опускать такие посылки.

Следующий пример - задача на упрощение выражения $(c \cap b) \cup (c \cap a) \setminus d$ в предположении, что выполняется $a \subseteq b$. Она оформляется как задача на преобразование, имеющая посылку "содержится(a b)"и условие "объединение(пересечение(c b) разность(пересечение(c a)d))". Эта задача имеет цели "упростить"и "одз". Ответом на нее служит выражение "пересечение(b c)".

Простейшим примером задачи на описание может служить задача на решение уравнений $a \cap x = b, a \cup x = c$ в множествах, если для известных множеств a, b, c выполнены соотношения $b \subseteq a, a \subseteq c$. Посылками такой задачи служат утверждения "содержится(b a)", "содержится(a c)", дополненные утверждениями о том, что a, b, c - множества. Условиями являются утверждения "равно(пересечение(a x)b)" , "равно (объединение(a x)c)"и "множество(x)". Как и в случае посылок, обычно добавление условий, указывающих тип значения неизвестных (например, "множество(x)"), выполняется автоматически; в последующих примерах такие условия опускаем. Задача имеет цели "полный" , "явное" , "прямойответ" , "одз" , "неизвестные x"и "упростить". Ответом задачи служит утверждение "равно(x объединение(b разность(c a)))".

Другая ситуация в задачах на описание - когда требуется найти не полное описание всех допустимых значений неизвестных, а привести хотя бы один пример таких значений. Так, если нужно найти пример множества x, удовлетворяющего соотношению $a \cup x = b$ в предположении, что для известных множеств a, b выполняется $a \subseteq b$, то рассматривается задача на описание, имеющая посылку "содержится(a b)"и условие "равно(объединение(a x)b)". Эта задача имеет цели "полный", "пример", "неизвестные x" , "прямойответ"и "одз". Ответом на нее может служить, например, утверждение "равно(x b)".

3.3.2 Элементарная алгебра

Примеры формулировки задач по элементарной алгебре начнем с вычисления значений константных выражений. В простейших случаях здесь применяется обычная задача на преобразование с целью "упростить". Она имеет единственную вырожденную посылку - логическую константу "истина". Условием служит упрощаемое выражение. Представление о том, что является упрощением - по существу, эвристическое. Оно возникло в процессе рассмотрения многих конкретных примеров на упрощение и соответствующих этим примерам коррекций действий решателя. Так, при упрощении выражения $\frac{(5+\sqrt{3})}{4-\sqrt{3}}$, решатель избавляется от иррациональности в знаменателе и выдает ответ $\frac{9\sqrt{3}+23}{13}$; при упрощении выражения $\arctan(\frac{1}{3})+\arctan(\frac{1}{5})+\arctan(\frac{1}{7})+\arctan(\frac{1}{8})$ выдается ответ $\frac{\pi}{4}$, и т.п. В зависимости от задачи, исходное выражение может остаться неизмененным либо (что дает обучающий материал для продолжения оптимизации приемов) быть преобразованным к худшему виду. Впрочем, в стандартных задачах с "хорошим"ответом такое явление уже сейчас наблюдается достаточно редко.

Если требуется найти точное целочисленное значение выражения, в котором встречаются степени с большим показателем, факториалы и т.п., то рекомендуется цель "упростить"сопровождать целью "число"(в интерфейсе решателя для ввода такой целевой комбинации предусмотрен специальный пункт меню). В этом случае снимаются блокировки на действия, приводящие к длинным десятичным записям, имеющие место в случае обычной задачи на упрощение. Интерпретатор языка ЛОС позволяет выполнять точные арифметические действия с десятичными записями чисел, имеющими до 3000 знаков.

Если требуется получить приближенное значение константного выражения, содержащего арифметические операции, степени, логарифмы, прямые и обратные тригонометрические функции, причем в этом приближенном значении нужно иметь заданное число точных знаков после запятой, то применяется задача на преобразование, имеющая цели "упростить"и "числоценка N", где N - число точных знаков после запятой. Производя вычисления, решатель получает гарантированные верхнюю и нижнюю оценки рассматриваемого выражения, увеличивая число цифр до тех пор, пока после округления "к ближайшему"в них не совпадут требуемые N цифр после запятой.

Наконец, для получения приближенной оценки константного выражения возможно использование встроенного в компьютер математического сопроцессора (14 значащих цифр). Здесь используется комбинация целей "числзначение"и "выч".

В случае задач на упрощение неконстантных алгебраических выражений, как и для константных, используется сочетание целей "упростить"и "одз". Например, для упрощения в области допустимых значений выражения $\frac{a}{a^2+b^2}-\frac{b(a-b)^2}{a^4-b^4}$ создается задача на преобразование, имеющая своим условием это выражение, посылками - утверждения "число(a)"и "число(b)"и указанные две цели. На нее выдается ответ $\frac{1}{a+b}$. Необходимые условия на о.д.з. (отличие знаменателей от 0) вводятся решателем в список посылок самостоятельно и в процессе решения изменяются так, чтобы не нарушалось соответствие между ними и "обслуживаемыми"ими подвыражениями условия задачи. Собственно в ответ результирующие условия на о.д.з. для параметров не включаются, но они легко могут быть считаны при пошаговом просмотре решения.

Задачи разложения на множители имеют целевую установку "упростить", "разложитьнамножители", "одз". Так, для разложения на множители выражения $a^2-b^2-c^2-2bc$ создается задача на преобразование с этими целями, условием которой служит

данное выражение, а посылки указывают тип значения переменных a, b, c. Заметим, что задачи на преобразование тригонометрических выражений к виду, удобному для логарифмирования, формулируются с теми же целями, что и обычные "алгебраические"задачи разложения на множители. Если требуется разложить на множители выражение, разрешая использование комплексных чисел, то кроме указанных выше трех целей добавляется четвертая - "видУмножение".

Чтобы разложить алгебраическую дробь в сумму простейших дробей, используется задача на преобразование, имеющая цели "простейшиедроби", "упростить", "одз". Например, решая задачу на преобразование с указанными целями и условием $\frac{1}{(a^2-a+1)(a^2+2a+3)}$, решатель выдает ответ $\frac{3a+4}{19(a^2+2a+3)} + \frac{5-3a}{19(a^2-a+1)}$.

Наконец, для упрощения алгебраических выражений с "раскрыванием скобок" предусмотрена целевая установка "упростить", "раскрытьскобки", "одз". Так, решая задачу на преобразование с данными целями и условием $a(b-c)+c(a-d)+d(c-b)$, решатель выдает ответ $ab-bd$. Хотя в этом ответе и можно было бы вынести за скобку общий множитель b, но цель "раскрытьскобки"блокирует такое действие.

Если требуется упростить некоторое выражение, вместо параметров которого должны быть подставлены другие выражения, то такую подстановку не обязательно делать непосредственно - достаточно указать в посылках задачи равенства, определяющие подставляемые выражения. Например, если имеется задача на преобразование с условием $-4a^2x^{\frac{1}{m}+\frac{1}{n}} + (x^{\frac{1}{m}} + x^{\frac{1}{n}})^2$ и посылкой $x = (\sqrt{a^2-1}+a)^{\frac{2mn}{m-n}}$ (не считая автоматически добавляемых посылок, указывающих тип значений переменных), то фактически будет упрощаться результат подстановки в условие выражения для x, определяемого посылкой.

При решении задачи на упрощение алгебраического выражения может возникнуть разбор подслучаев, в результате чего ответ будет иметь вид "условного"выражения. Так, при упрощении выражения $\sqrt{a+2\sqrt{2a-4}} + \sqrt{a-2\sqrt{2a-4}}$ решатель выдает ответ "$2\sqrt{2}$ при $a<4$, иначе $2\sqrt{a-2}$"(для большей наглядности вместо скобочной конструкции "вариант(...)"мы использовали здесь стандартную запись, прорисовываемую формульным редактором решателя).

Задачи на суммирование также оформляются как стандартные задачи на упрощение. Например, задача на преобразование с целями "упростить", "одз", условием $\sum_{m=1}^{n}(2m-1)^3$ и посылкой "натуральное(n)"приводится к ответу $(2n^2-1)n^2$. Заметим, что в этих задачах важно указывать в посылках, что значениями параметров, определяющих границы суммирования, служат целые числа, и сопровождать такие параметры всеми необходимыми неравенствами.

Задачи на решение уравнений, неравенств, систем уравнений и неравенств практически всегда имеют целевую установку "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $x_1 \ldots x_n$". Например, чтобы решить уравнение $\frac{\sqrt{1+a^2x^2}-ax}{\sqrt{1+a^2x^2}+ax} = \frac{1}{b^2}$ относительно неизвестной x, создается задача на описание с указанной выше целевой установкой, условиями которой служат данное уравнение и утверждение "число(x)"(последнее создается автоматически процедурами интерфейса решателя). Эта задача имеет посылку "число(a)". Решая задачи с параметрами, решатель аккуратно анализирует все возможные случаи, объединяя полученные для них результаты в общем ответе. Так, в указанном уравнении ответ будет объединять два подслучая: $a \neq 0, b \neq 0, x = \frac{b^2-1}{2a|b|}$ и $(b=-1 \vee b=1), a=0$ с "вынесенной за скобку"общей частью "число(x)".

Если в задаче требуется найти все значения параметров, при которых система уравнений либо неравенств имеет хотя бы одно решение, то условие записывается

с помощью квантора существования. Например, для нахождения всех значений параметра a, при которых имеет решение система уравнений $y(ax-1) = 2|x+1| + 2xy, xy+1 = x-y$, вводится задача на описание с условием $\exists xy(y(ax-1) = 2|x+1| + 2xy \& xy+1 = x-y)$. Целевая установка у нее - такая же, как и выше (но роль неизвестной играет a). Аналогично, если нужно найти все значения параметров, при которых система уравнений либо неравенств имеет заданное число решений, то используется описатель "класс", с помощью которого обозначается множество решений рассматриваемой системы. Так, для отыскания значений параметра a, при которых система $\log_2 y + 2 = \log_2(x+3y),\ y = 2(x-a)^2 + x + 2a - 4$ имеет ровно два решения, создается задача на описание с условием "мощность(класс($xy\ (\log_2 y + 2 = \log_2(x+3y)\ \&\ y = 2(x-a)^2 + x + 2a - 4)$))"$= 2$. Если требуется найти значения параметров, при которых две системы (два уравнения, и т.п.) имеют одинаковые множества решений, то условие записывается в виде эквивалентности под квантором общности. Например, чтобы найти значения параметра a, при которых множество решений уравнения $4(\cos x)^2 = a^2 - 6$ совпадает со множеством решений уравнения $1-\cos(2x) = \frac{a}{6}$, используется задача на описание с условием $\forall x(4(\cos x)^2 = a^2-6 \Leftrightarrow 1-\cos(2x) = \frac{a}{6})$. Заметим, что в перечисленных примерах под кванторами и описателем "класс"автоматически вводятся дополнительные утверждения "число(...)", уточняющие тип значений связанных переменных.

Если требуется определить число решений уравнения (системы) в зависимости от значений параметров, то возникает уже не задача на описание, а задача на преобразование. Условием такой задачи служит выражение, задающее мощность множества корней, а цели - стандартные: "упростить"и "одз". Решатель пытается получить явное представление для множества корней, решая рассматриваемое уравнение (систему) либо иными средствами определяет мощность этого множества (например, анализируя с помощью производных и пределов интервалы монотонности). Так, если нужно определить число решений уравнения $\sqrt{\sqrt{x+\frac{1}{4}} + x + \frac{1}{2}} = a$, то условием задачи на преобразование будет выражение "мощность(класс($x\ \sqrt{\sqrt{x+\frac{1}{4}} + x + \frac{1}{2}} = a$))". Ответ, получаемый здесь решателем, имеет вид "1, если $\frac{1}{4} \le a$, иначе 0".

При доказательстве тождества либо неравенства создается задача на доказательство, условием которой является это тождество либо неравенство, а посылки перечисляют ограничения на параметры. Так как задача на доказательство не имеет списка целей, то указание на пополнение списка посылок ограничениями на область допустимых значений передается ей не в виде цели "одз", а в виде комментария "одз". Это делается автоматически интерфейсом решателя. В качестве примера приведем задачу на доказательство с условием $64ab(a+b)^2 \le (\sqrt{a} + \sqrt{b})^8$ и посылками "число(a)", "число(b)". Неотрицательность параметров a, b регистрируется в списке посылок уже при решении задачи, когда обрабатывается комментарий "одз".

3.3.3 Комбинаторика

Задача по комбинаторике обычно формализуется как задача на преобразование выражения "мощность(A)". Целевая установка - стандартная, т.е. "Упростить в о.д.з.". Извлечение выражения "мощность(A)" из текста задачи выполняется вручную. Например, рассмотрим следующий текст: "Экскурсанты разделились на две равные группы для розыска заблудившегося товарища. Среди них есть только 4 человека, знакомые с местностью. Каким числом способов они могут разделиться так, чтобы

в каждую группу вошло 2 человека, знающих местность, если всего их 16 человек?". Множество всех экскурсантов обозначаем через A; подмножество экскурсантов, знакомых с местностью - D. Тогда вводим следующие три посылки задачи: $\mathrm{card}(A) = 16$, $\mathrm{card}(D) = 4$, $D \subseteq A$. Если подсчитывается количество упорядоченных пар групп, то преобразуемое условие имеет вид $\mathrm{card}(\mathrm{set}_{BC}(B - \mathrm{set}\ \&\ C - \mathrm{set}\ \&\ A = B \cup C\ \&\ B \cap C = \emptyset\ \&\ \mathrm{card}(B) = \mathrm{card}(C)\ \&\ \mathrm{card}(B \cap D) = 2\ \&\ \mathrm{card}(C \cap D) = 2))$. Если же подсчитывать количество неупорядоченных пар групп (что, собственно, и требуется в задаче), то условие несколько усложняется: $\mathrm{card}(\mathrm{set}_x(\exists_{BC}(x = \{B, C\}\ \&\ B - \mathrm{set}\ \&\ C - \mathrm{set}\ \&\ A = B \cup C\ \&\ B \cap C = \emptyset\ \&\ \mathrm{card}(B) = \mathrm{card}(C)\ \&\ \mathrm{card}(B \cap D) = 2\ \&\ \mathrm{card}(C \cap D) = 2)))$. Обе версии без труда доводятся решателем до ответа.

Приведем еще один пример, иллюстрирующий часто встречающиеся в задачах по комбинаторике конструкции с отображениями: "Из цифр 1,2,3,4,5,6,7,8,9 составляются всевозможные пятизначные числа, не содержащие одинаковых цифр. Определить количество чисел, в которых есть цифры 2,4 и 5 одновременно.". Пятизнач ное число можно рассматривать как отображение множества номеров цифр $\{1, \ldots, 5\}$ в множество цифр $\{1, \ldots, 9\}$. Условие наличия в числе x цифры m тогда запишется как $\neg(\text{слой}(x, m) = \emptyset)$; условие различия цифр числа - "взаимнооднозначно(x)". Таким образом, условие задачи на преобразование имеет вид $\mathrm{card}(\mathrm{set}_f(\text{Отображение}(f, \{1, \ldots, 5\}, \{1, \ldots, 9\})\ \&\ \text{взаимнооднозначно}(f)\ \&\ \neg(\text{слой}(f, 2) = \emptyset)\ \&\ \neg(\text{слой}(f, 4) = \emptyset)\ \&\ \neg(\text{слой}(f, 5) = \emptyset)))$.

3.3.4 Элементарная геометрия

Вычислительные задачи по планиметрии внешне напоминают задачи на решение уравнений и неравенств - в обоих случаях требуется определить значения "неизвестных"величин. Однако, здесь имеются существенные различия, приводящие к необходимости особой логической формализации. Прежде всего, параметры посылок геометрической задачи бывают двух типов - числовые, про которые обычно предполагается, что они "известны", и параметры - обозначения точек чертежа. Последние, хотя и содержатся в посылках, "известными"никоим образом не считаются и в ответ входить не должны. Более того, те величины, которые требуется определить в геометрической задаче, обычно изначально явно выражены через ее "точечные"параметры, так что с точки зрения обычной "алгебраического типа"задачи на описание, ситуация вырожденная. Эти обстоятельства потребовали ввести для геометрических задач на вычисление специальную цель "известно $a_1 \ldots a_n$", которая указывает все параметры $a_1, \ldots, a_n$, которые могут встречаться в ответе (возможен случай $n = 0$, и тогда цель сводится к логическому символу "известно"). Изначально имеющаяся информация о чертеже перечисляется в списке посылок задачи; условиями служат равенства, явно выражающие неизвестные величины через обозначения точек чертежа.

В качестве простейшего примера рассмотрим следующую задачу. В треугольнике ABC длина стороны AC равна a; угол BAC равен b, а угол BCA равен c. Требуется вычислить площадь данного треугольника. Эта ситуация формализуется в виде задачи на описание, посылками которой служат следующие утверждения: "треугольник (ABC)"; "расстояние(AC)= a"; "угол(BAC)= b"; "угол(BCA)= c". Они дополняются вводимыми автоматически утверждениями о типе значения: "точка(A)", "точка(B)", "точка(C)", "число(a)", "число(b)", "число(c)". Задача имеет условия "x =площадь (фигура(ABC))"и "число(x)". Целевая установка ее состоит из целей "неизвестные x"; "известно abc"; "полный"; "прямойответ"; "явное"; "упростить"; "одз".

При формулировке утверждений, описывающих чертеж геометрической задачи, следует учитывать некоторую ограниченность созданного на текущий момент запаса понятий. Так, например, отсутствует в явном виде понятие "медиана треугольника". Если в условии задачи говорится про медиану, то нужно вводить в рассмотрение новую точку - основание медианы, и указывать в посылках, что расстояния ее до концов стороны треугольника равны, а сама эта точка лежит на данной стороне (т.е. на отрезке с соответствующими концами). Впрочем, последнее условие можно ослабить - задать принадлежность точки не стороне, а прямой, на которой лежит эта сторона. Приведем несколько примеров простых задач, в которых встречаются такого рода "стандартные"для решателя формулировки геометрических условий.

Пусть имеется треугольник ABC, у которого длина стороны AB равна 6, длина стороны BC равна 8; медианы, проведенные из вершин A и C, взаимно перпендикулярны, причем требуется найти площадь треугольника. Посылки соответствующей задачи на описание таковы: "треугольник(ABC)"; "расстояние(AB)= 6"; "расстояние(BC)= 8"; "$D \in$ отрезок(BC)"; "расстояние(BD) = расстояние(CD)"; "$E \in$ отрезок(AB)"; "расстояние(AE) = расстояние(BE)"; "перпендикулярно(прямая(CE) прямая(AD))". Условием служит равенство "$x =$ площадь(фигура(ABC))". Задача имеет цели "неизвестные x", "известно", "полный", "прямойответ", "явное", "упростить", "одз".

Аналогичные примеры приведем для биссектрисы и высоты треугольника. В первом из них даны длины a, b, c сторон BC, AC, AB треугольника ABC. Биссектрисы треугольника, проведенные из вершин B и C, пересекаются в точке D. Требуется найти, в каком отношении точка D делит биссектрису, проведенную из вершины B. Посылками задачи для этого примера служат утверждения "треугольник(ABC)"; "расстояние(BC)= a"; "расстояние(AC)= b"; "расстояние(AB)= c"; "биссектриса ($ABCR$)"(введена точка R - основание биссектрисы угла B); "$R \in$отрезок(AC)"(точка R лежит на стороне AC); "биссектриса($ACBP$)"; "$P \in$отрезок(AB)"; (аналогично, для биссектрисы угла C введено основание P); "$D \in$отрезок(BR)"; "$D \in$отрезок(CP)" (введена в рассмотрение точка D пересечения биссектрис). Условие задачи имеет вид "$x =$расстояние(BD) / расстояние(DR)". Цели задачи суть: "неизвестныеx"; "известно abc"; "полный"; "явное"; "прямойответ"; "одз"; "упростить".

В другом примере известно, что длина стороны BC треугольника ABC равна 8; длины высот треугольника, проведенных из вершин A и B, равны, соответственно, 4 и 6.4. Требуется найти длины сторон AB и AC. Посылки задачи суть: "треугольник (ABC)"; "расстояние(BC)=8"; "$D \in$ прямая(BC)"(введено основание D высоты, проведенной из A); "перпендикулярно(прямая(AD) прямая(BC))"(высота перпендикулярна основанию); "$E \in$ прямая(AC)"(основание второй высоты); "перпендикулярно(прямая(BE)прямая(AC))"; "расстояние(AD)= 4"; "расстояние(BE)= 6.4". Условия этой задачи имеют вид: "$x =$ расстояние(AB)"; "$y =$расстояние(AC)".

Заметим, что в планиметрических задачах обычно имеется фиктивная посылка "планиметрия". Как правило, она вводится автоматически после набора задачи, однако иногда ее приходится вводить вручную (последовательным нажатием клавиш "п","и"). Эта посылка необходима для ускорения проверок принадлежности рассматриваемых прямых и точек общей плоскости. В особых случаях ее отсутствие может также привести к выдаче отказа на задачу.

Для указания на то, что точка принадлежит внутренности фигуры, ограниченной системой лучей и отрезков, используются утверждения "однасторона(...)"; "разныестороны(...)". Так, рассмотрим следующую задачу. Внутри угла BAC, величина которого равна a, взята точка M. Из нее опущены перпендикуляры на стороны угла,

причем известно, что основания перпендикуляров отстоят от вершины угла на расстояния p и q. Нужно найти длины перпендикуляров. Данная ситуация описывается следующей системой посылок: "угол(BAC)= a"; "однасторона(C, M,прямая(A B))"; "однасторона(B, M,прямая(AC))"(условие принадлежности точки M углу сформулировано как пара условий: эта точка лежит по ту же сторону от одной стороны угла, что и направляющая точка другой стороны); "перпендикулярно(прямая(PM) прямая(AB))"; "$P \in$ прямая(AB)"(P - основание первого перпендикуляра); "перпендикулярно(прямая(QM)прямая(AC))"; "$Q \in$прямая(AC)"(Q - основание второго перпендикуляра); "расстояние(AP)= p"; "расстояние(AQ)= q"; $0 < a$; $a < \pi$ (невырожденность угла); $0 < p$; $0 < q$ (эти два неравенства уточняют, что точка M находится во внутренности угла). Условия задачи суть: "x =расстояние(PM)"; "y =расстояние (QM)". Заметим, что условие принадлежности точки углу могло быть сформулировано и непосредственным образом, как "$M \in$Угол(BAC)". Приведенная выше более громоздкая формулировка нужна здесь лишь как иллюстрация возможного применения предикатов "однасторона"и "разныестороны".

Геометрические задачи на доказательство формулируются почти так же, как и задачи на вычисление: в посылках задается чертеж; условием служит утверждение, которое требуется доказать. Например, задача на доказательство того, что каждый выпуклый четырехугольник, диагонали которого суть биссектрисы его внутренних углов, является ромбом, имеет следующие посылки: "четырехугольник($ABCD$)"; "биссектриса($BADC$)"; "биссектриса($BCDA$)"; "биссектриса($ABCD$)"; "биссектриса($ADCB$)". Условие этой задачи - "ромб($ABCD$)".

Геометрические задачи на построение и на определение геометрического места точек хорошо формализуются в виде обычных задач на описание, имеющих стандартную целевую установку "неизвестные $x_1 \dots x_n$", "полный", "явное", "прямойответ", "упростить", "одз". В качестве примера рассмотрим задачу нахождения геометрического места всех таких точек C на данной прямой AB, сумма расстояний которых до точек A, B - наименьшая. Предполагается известным, что расстояние от A до B равно 50. Она формализуется как задача на описание, имеющая посылку "расстояние (AB)= 50", и условие "Минимум(отображение(C точка(C) расстояние(AC)+расстояние(BC)) прямая(AB) x y)". Неизвестные этой задачи суть x, y; значение первой из них равно наименьшей сумме рассматриваемых расстояний; значение второй - множеству точек, в которых достигается эта наименьшая сумма. Решатель выдает на задачу ответ "$y =$ отрезок(AB); $x = 50$".

Другой пример - задача на построение треугольника ABC по известным длинам a, b, c его сторон AB, AC, BC. Она формализуется в виде задачи на описание с условиями "треугольник(ABC)"; "расстояние(AB)= a"; "расстояние(AC)= b"; "расстояние (BC)= c". Неизвестными этой задачи служат A, B, C. Решатель выдает на данную задачу ответ "$0 < a + b - c$, $0 < a + c - b$, $0 < b + c - a$, точка(C), $A \in$окр(C b), $B \in$ окр(A a)$\cap$ окр(C c)". Напомним, что "окр(A b)"обозначает окружность с центром в точке A, имеющую радиус b. Разумеется, в задачах на построение вместо циркуля и линейки используется ряд вспомогательных элементарных операций и отношений, позволяющих последовательно определять новые точки по ранее найденным.

3.3.5 Математический анализ

Вычисление производных и пределов происходит при помощи задач на преобразование, имеющих цели "упростить"и "одз". Стандартный вид условия задачи на вычисление производной - "производная(отображение(y число(y) $f(y)$)x)". Здесь $f(y)$

- выражение, определяющее значение дифференцируемой функции в точке y. Заметим, что переменная y - связанная и используется только внутри конструкции "отображение(...)", задающей исходную функцию. Условие задачи здесь имеет своим значением не функцию - производную исходной функции, а лишь значение этой производной в точке x. В качестве x может фигурировать либо переменная, и тогда на экране условие будет прорисовано в виде $\frac{df(x)}{dx}$, либо некоторое другое выражение (например, константа), и тогда на экране условие будет прорисовано в виде $\frac{df(y)}{d(y=x)}$. Важным является то обстоятельство, что решатель не только выполняет формальное дифференцирование, но и осуществляет последующие упрощения полученного выражения в его области допустимых значений. Это занимает, как правило, значительно больше времени, чем само нахождение производной, но зато позволяет получать результаты удовлетворительного качества - по крайней мере для примеров из стандартных задачников.

При вычислении частных производных и производных высших порядков используется другой вид условия задачи на преобразование. Например, для функции двух переменных оно таково: "частнпроизв(отображение($u\ v$ и(число(u)число(v)) $f(u,v)$) набор($k\ m$)набор(xy))". Здесь k, m - кратности дифференцирования по первой и второй переменным (одна из них может равняться 0); $f(u,v)$ - выражение, определяющее значение дифференцируемого выражения. На экране указанное условие прорисовывается как $\frac{d^{k+m}f(x,y)}{dx^k dy^m}$ Как и в случае одной переменной, если, например, x не является переменной, то запись преобразуется к виду $\frac{d^{k+m}f(u,y)}{d(u=x)^k dy^m)}$.

Если нужно продифференцировать функцию, заданную параметрически, то приходится выбирать одну из двух возможностей логической формализации. Либо нужно вводить специальное обозначение для параметрического "определения"новой функции $y(x)$ по двум функциям $x = f(t)$ и $y = g(t)$, либо работать только с одной функцией $y(x)$, для которой задано соотношение $y(f(t)) = g(t)$. В решателе выбрана вторая возможность, как технически более простая. При такой записи, для нахождения производной функции, заданной параметрическими соотношениями $y(t) = b\ sh(t)$, $x(t) = a\ ch(t)$, вводится задача на преобразование, имеющая посылку "$\forall t$(одз(t) $\Rightarrow$ $y(a\ ch(t)) = b\ sh(t)$)". Здесь запись "одз(t)"имеет технический характер; она заменяется решателем на группу условий, определяющих принадлежность t области допустимых значений для выражений, расположенных справа от импликации. Условие задачи записывается как $\frac{dy(x)}{d(x=a\ ch(t))}$. Решатель выдает ответ $\frac{b\ cth(t)}{a}$.

В случае нахождения производной неявной функции применяется аналогичная запись. Так, чтобы продифференцировать функцию $y(x)$, заданную неявным соотношением $\sqrt{y(x)} + \sqrt{x} = \sqrt{a}$, вводится задача на преобразование, имеющая посылку $\forall x$ (одз(x) $\Rightarrow$ $\sqrt{y(x)} + \sqrt{x} = \sqrt{a}$)). Условие имеет вид $\frac{dy(x)}{dx}$. Ответ выдается в виде $-\sqrt{\frac{y(x)}{x}}$.

При вычислении пределов условие задачи на преобразование имеет вид "предел (отображение(x число(x) $f(x)$)$m\ a$)", где a - точка, в которой определяется предел; m - указатель типа окрестности этой точки (0 - двусторонняя окрестность; 1 - левая; 2 - правая). Это условие прорисовывается, соответственно, как $\lim_{x\to a} f(x)$ либо $\lim_{x\to a-0} f(x)$ либо $\lim_{x\to a+0} f(x)$. Заметим, что в случае выражения с параметрами решатель может выдать ответ с перечислением подслучаев. Так, при решении задачи с посылками $0 < a$, $0 < b$ и условием $\lim_{x\to\infty} \left(\frac{ax+c}{bx+d}\right)^x$, получается ответ, состоящий из трех подслучаев: 1) если $0 < -\frac{a}{b} + 1$, то 0; 2) если $0 < \frac{a}{b} - 1$, то ∞; 3) в остальных случаях $\exp(\frac{c-d}{b})$.

При нахождении пределов последовательностей вид условия отличается от указанного выше только тем, что внутри конструкции "отображение(...)", определяющей рассматриваемую функцию, вместо утверждения "число(x)"берется утверждение "натуральное(x)". В этом случае можно формулировать также задачи на нахождение верхнего и нижнего пределов (логические символы "верхнийпредел", "нижнийпредел" вместо символа "предел"). Чтобы при стандартной прорисовке на экране можно было отличать пределы функций вещественного переменного от пределов последовательностей, в последних выражение под знаком предела заключается в фигурные скобки (это эквивалент использования во внутреннем представлении утверждения "натуральное(x)"вместо "число(x)").

Для нахождения точных верхних и нижних граней функций одного вещественного переменного на заданных множествах также применяются задачи на преобразование с целями "упростить", "одз". Например, для точной верхней грани условие задачи имеет вид "суп(образ(отображение(x число(x) $f(x)$)M))", где M - множество, на котором ищется точная верхняя грань. На экране оно прорисовывается как "sup(образ(λ_x($f(x)$, число(x)),M))".

Если нужно найти наибольшее либо наименьшее значение функции на некотором множестве, а также определить точки, в которых достигается такое значение, то применяются уже не задачи на преобразование, а задачи на описание. Целевая установка их стандартная - "неизвестные xy", "полный", "явное", "прямойответ", "упростить", "одз". Здесь x - множество, на котором достигается искомое наибольшее либо наименьшее значение, y - это значение. Условие задачи на описание (например, в случае наибольшего значения) имеет вид "Максимум(отображение(x число(x) $f(x)$) M x y)". Здесь M - множество, на котором ищется наибольшее значение. На экране такое условие прорисовывается как "Max(λ_x($f(x)$,число(x)),M, x, y)".

При поиске экстремумов тоже используется задача на описание. Условие ее имеет вид "экстремум(отображение(x число(x) $f(x)$)y z v)", где значением неизвестной y служит точка, в которой достигается экстремум функции; значением неизвестной z - значение функции в точке экстремума; значением неизвестной v - указатель типа экстремума - логический символ "минимум"либо "максимум". Область, в которой ищутся экстремумы, можно сужать, добавляя к списку условий необходимые ограничения на y. Решатель пытается найти все экстремумы. Так, например, в задаче с условием "Extr($\lambda_x(\frac{\cos x}{\cos(2x)}$, число($x$)),$y$, z, v)" выдается ответ, состоящий из двух подслучаев: 1) $\exists k(y = 2\pi k$ & целое(k)), $z = 1, v = \min$; 2) $\exists k(y = \pi(2k+1)$ & целое(k)), $z = -1, v = \max$. При постановке задачи с экстремумами не обязательно требовать, чтобы указанные выше y, z, v были переменными. Вместо них могут быть подставлены любые выражения. Неизвестными, кроме входящих в y, z, v переменных, могут также быть какие-либо параметры, использованные при задании функции. Так, если вместо y подставить выражение без неизвестных, а неизвестными считать входящие в определение функции параметры, то смысл задачи будет состоять в отыскании таких значений параметров, для которых функция имеет в заданной точке y экстремум.

Чтобы получить качественное описание графика функции с помощью пределов и производных, применяются задачи на описание со списком целей "исследовать", "полный", "явное", "прямойответ", "функция", "неизвестные x". Условием такой задачи на описание является единственное равенство "$x =$ отображение(y число(y) $f(y)$)", определяющее исследуемую функцию. При решении задачи сразу же вводится ее блок анализа, в котором происходит логический вывод с постепенным накоплением информации о поведении функции x. В процессе вывода усматриваются те утверждения, которые целесообразно включать в итоговое описание свойств функ-

ции, и они переносятся из блока анализа в список условий задачи на описание. По исчерпании возможностей вывода выдается ответ - конъюнкция утверждений списка условий (в нем сохраняется и исходное равенство для x). В качестве примера приведем задачу с условием $x = \lambda_y(-y^3+3y, \text{число}(y))$. Решатель выдает на нее следующий ответ (без учета повторения исходного условия): "функция(x)"; "Extr($x, 1, 2, \max$)"; "убывает $(x, (1, \infty))$"; "Extr($x, -1, -2, \min$)"; "убывает($x, (-\infty, -1)$)"; "возрастает($x, (-1, 1$)
"область(x)= $\Re$ "; "мощность(корни($x, (1, \infty)$))=1"; "мощность(корни($x, (-1, 1)$))=1"; "мощность(корни($x, (-\infty, -1)$))=1".

Если в качественное описание графика нужно включить также указание на интервалы выпуклости-вогнутости, то к указанной выше целевой установке добавляется цель "выпуклавверх". Например, при наличии этой цели, на задачу с условием $x = \lambda_y(-y^3 + 3y^2, \text{число}(y))$ решатель находит ответ: "функция(x)"; "Extr($x, 2, 4, \max$)"; "убывает($x, (2, \infty)$)"; "Extr($x, 0, 0, \min$)"; "убывает($x, (-\infty, 0)$)"; "возрастает($x, (0, 2)$)"; "область(x) = $\Re$"; "выпуклавниз($x, (-\infty, 1)$)"; "выпуклавверх($x, (1, \infty)$)"; "мощность (корни($x, (2, \infty)$)) = 1"; "корни($x, (0, 2)$)=$\emptyset$"; "корни($x, (-\infty, 0)$)= $\emptyset$".

Кроме задач на общее качественное исследование функций, рассматриваются также некоторые дополнительные типы задач, сязанных с исследованием функций одного вещественного переменного. Все они имеют в составе своей целевой установки цели "исследовать", "полный", "явное", "прямойответ", "функция", "неизвестные x". К ним добавляется цель, уточняющая тип проводимого исследования, причем такая цель блокирует проведение общего качественного исследования (исключение здесь составляет указанный выше случай цели "выпуклавверх"). Решение этих задач осуществляется по указанной выше схеме - путем вывода следствий в блоке анализа и перенесении представляющих ценность результатов вывода в список условий внешней задачи на описание. При исследовании функции на четность-нечетность и периодичность добавляется цель "четнаяфункция"; при исследовании на непрерывность - добавляется цель "непрерывно"; при исследовании на равномерную непрерывность добавляются цели "непрерывно"и "равномернонепрерывно". В случае исследования на непрерывность решатель анализирует типы точек разрыва, причем в случае исследования на равномерную непрерывность анализ точек разрыва блокируется. Приведем несколько простых примеров задач указанных типов. Так, при наличии цели "четнаяфункция"на задачу с условием

$$y = \lambda_x(\frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} + \sin x, \text{число}(x))$$

решатель выдает ответ "область(y)= $\Re$"; "периодична($y, 2\pi$)"; "нечетнаяфункция(y)".
При наличии цели "непрерывно"на задачу с условием

$$y = \lambda_x(\left[\frac{1}{x}\right], \text{число}(x))$$

выдается ответ: "область(y)=$(-\infty, 0) \cup (0, \infty)$"; "$\forall n$(целое($n$) & $\neg(n = 0) \Rightarrow$ разрывпервогорода($y, \frac{1}{n}$))"; "разрыввторогорода($y, 0$)"; "непрерывно($y, ((-\infty, 0) \cup (0, \infty)) \setminus$ класс(x $\exists n$(целое(n) & $\neg(n = 0)$ & $x = \frac{1}{n}$)))".
При наличии целей "непрерывно" и "равномернонепрерывно"на задачу с условием

$$y = \lambda_x(\ln(x), \text{число}(x) \ \&\ 0 < x \ \&\ x < 1)$$

выдается ответ: "область(y) = $(0, 1)$"; "$\neg$(равномернонепрерывно($y, (0, 1)$))"; "непрерывно($y, (0, 1)$)".

Для определения числа корней функции на заданном множестве создается задача на преобразование с целями "упростить", "одз" и условием "мощность(корни(отображение(x $A(x)$) $f(x)$))", где $A(x)$ - условие принадлежности аргумента x рассматриваемому множеству; $f(x)$ - выражение, определяющее значение функции. Если условие содержит параметры, то ответ задачи может иметь вид условного выражения, определяющего искомое число корней путем разбора случаев.

В случае функций нескольких переменных задачи на отыскание экстремумов и наибольших (наименьших) значений формулируются аналогично случаю одной переменной. Это задачи на описание с целями "полный", "явное", "упростить", "одз", "прямойответ", "неизвестные $z_1 \dots z_k$". В случае безусловного экстремума условие задачи имеет вид "экстремум(отображение($x_1 \dots x_n$ и(число(x_1) ... число(x_n)) $f(x_1 \dots x_n)$) z_1 z_2 z_3)"; в случае условного - к списку утверждений "число(x_1)", ..., "число(x_n)" внутри описателя "отображение" добавляются необходимые равенства и неравенства для варьируемых переменных $x_1, \dots, x_n$. Значением неизвестной z_1 служит набор значений указанных переменных, определяющий точку экстремума; z_2 - значение функции в этой точке; z_3 - тип экстремума (логический символ "минимум" либо "максимум"). В качестве примера приведем задачу на описание, имеющую единственную посылку $0 < a$ и условие "экстремум(отображение(xyz и(число(x)число(y)число(z) $xy+xz+yz = a^2$ $0 < x$ $0 < y$ $0 < z$) xyz) u v w)". Решатель выдает на нее ответ: $u = (\frac{a}{\sqrt{3}}, \frac{a}{\sqrt{3}}, \frac{a}{\sqrt{3}}), v = \frac{a^3}{3\sqrt{3}}, w = \max$. Другой пример - задача с условием "Максимум(отображение(xy и(число(x) число(y)) $-\sin(xy) + \sin(x) + \sin(y)$) класс($xy$ и(число(x) число(y) $0 \le x$ $0 \le y$ $x + y \le 2\pi$)) u v)", в которой неизвестными являются u, v (соответственно, множество точек, где достигается наибольшее значение, и само это значение). Решатель получает ответ $u = \left\{\left(\frac{2\pi}{3}, \frac{2\pi}{3}\right)\right\}, v = \frac{3\sqrt{3}}{2}$.

Задачи на вычисление интегралов любых типов (определенных, неопределенных, кратных) оформляются как задачи на преобразование со стандартными целями "упростить", "одз". В случае неопределенного интеграла условие имеет вид "Интеграл(отображение(x число(x) $f(x)$))". В качестве ответа выдается какая-либо одна из первообразных, записанная в виде "отображение(x число(x) $g(x)$)". Разумеется, прорисовка интеграла выполняется формульным редактором в стандартном виде (после знака интеграла набирается подынтегральное выражение, и далее - как дополнительные множители - d,x; если подынтегральное выражение имеет вид дроби, то перенесение в ее числитель этих множителей не допускается). Так, на задачу с условием

$$\int \frac{1}{x\sqrt{x^4 + 2x^2 - 1}}\, dx$$

выдается ответ

$$\lambda_x \left(\frac{\arcsin(-\frac{1}{\sqrt{2}x^2} + \frac{1}{\sqrt{2}})}{2}, \text{число}(x) \right)$$

.

При вычислении определенного интеграла пределы интегрирования a, b извлекаются из списка утверждений, описывающих область определения интегрируемой функции. Условие задачи на преобразование при этом имеет вид "интеграл(отображение(x и(число(x) $a \le x$ $x \le b$) $f(x)$))". На экране оно прорисовывается в обычном виде:

$$\int_a^b f(x)dx$$

.

Задача на вычисление двойного интеграла включает в себя описание области интегрирования, которое часто оказывается достаточно громоздким. Поэтому при формулировке таких задач для обозначения области интегрирования обычно вводится вспомогательная переменная, и в посылках задачи указывается определяющее ее равенство. Условие задачи имеет тогда вид "двойнойинтеграл(отображение(xy $(x,y) \in P$ $f(x,y)$))", где P - обозначение области интегрирования. Формульный редактор прорисовывает это условие в виде

$$\iint_P f(x,y)dxdy$$

(P располагается непосредственно под знаком двойного интеграла). Разумеется, можно и не выносить задание области интегрирования в список посылок, но тогда при изображении на экране получится громоздкая многоэтажная запись. В качестве примера приведем задачу с посылкой $P =$ класс(xy число(x) & число(y) & $0 \le x$ & $0 \le y$ & $4x^2 - 3y^2 \le 4$ & $4y^2 - 3x^2 \le 4$) и условием

$$\iint_P (x^3y + xy^3)dxdy.$$

Решатель выдает для нее ответ 3.

Типичной при вычислении двойного интеграла является ситуация, когда область интегрирования задана не непосредственно с помощью неравенств, как в приведенном выше примере, а косвенно - через ограничивающие ее кривые. В этом случае она может быть задана с помощью выражения "областьграницы(G)", где G - выражение, определяющее объединение множеств точек указанных кривых. Здесь значением выражения "областьграницы(G)", как и значением выражения G, является не множество самих точек плоскости, а лишь множество их координат. Предполагается, что рассматриваемые кривые разбивают плоскость на некоторые компоненты связности, и в область интегрирования включаются все конечные компоненты (таким образом, она может оказаться и несвязной). В качестве примера приведем задачу с посылкой $P =$ областьграницы($set_{xy}(y = x$ & число(x) & число(y))$\cup set_{xy}(y = x+a$ & число(x) & число(y))$\cup set_{xy}(y = a$ & число(x) & число(y))$\cup set_{xy}(y = 3a$ & число(x) & число(y))) и условием

$$\iint_P (x^2 + y^2)dxdy.$$

Решатель выдает на нее ответ $14a^4$. Заметим, что использованное выше обозначение $set_{xy}F(x,y)$ - результат прорисовки формульным редактором выражения "класс(xy $F(x,y)$)".

При вычислении площадей плоских множеств, объемов и площадей поверхностей соответствующее множество уже является не множеством числовых пар либо троек, а множеством точек плоскости либо трехмерного пространства, имеющих своими координатами эти пары либо тройки. Поэтому вместо выражения вида "областьграницы(...)"для задания множества используется выражение вида "точки(областьграницы(...)K))", где K - прямоугольная система координат. В качестве примера вычисления площади плоского множества приведем задачу с посылками
"$P =$ точки(областьграницы($set_{xy}((x^2+y^2)^3 = a^2(x^4+y^4)$ & число(x) & число(y))), K), "прямкоорд(K)", "$0 < a$" и условием "площадь(P)". Решатель выдает на нее ответ $3\pi\left(\frac{a}{2}\right)^2$.

Пример на вычисление объема - задача на преобразование с посылками
"$P =$ точки($set_{xyz}(x^2 + y^2 \le a^2$ & $0 \le az$ & $az \le a^2 - 2y^2$ & число(x) & число(y) &

число$(z)), K)$", "прямкоорд(K)", $0 < a$ и условием "объем(P)". На нее выдается ответ $\frac{(\pi+4)a^3}{4}$.

Наконец, пример вычисления площади поверхности - задача с посылками "$P =$ точки$(set_{xyz}(2az = x^2 + y^2$ & $x^2 + y^2 \le a^2$ & $0 \le y$ & $0 \le x$ & $y \le x$ & число(x) & число(y) & число$(z)), K)$", "прямкоорд(K)"и условием "площадь(P)". На нее выдается ответ $\frac{(2\sqrt{2}-1)\pi a^2}{12}$. Заметим, что при вычислении площади поверхности допустимо ее параметрическое задание. Пример такого задания - задача с посылками "$P =$ точки$(set_{xyz}(\exists uv(x = u\cos v$ & $y = u\sin v$ & $z = 4v$ & $x^2 + y^2 \le 9$ & число(u) & число(v) & $0 \le z$ & $z \le 8\pi)), K)$", "прямкоорд(K)"и условием "площадь(P)".

В задачах на исследование сходимости рядов используется запись ряда как последовательности своих частичных сумм: "отображение$(n$ натуральное(n) суммавсех (отображение$(i$ и(целое(i) $1 \le i$ $i \le n)$ $a(i))))$". На экране такая запись прорисовывается в виде:

$$\lambda_n \left(\sum_{i=1}^{n} a(i), \text{натуральное}(n) \right).$$

В качестве примеров задач, связанных с исследованием сходимости рядов, приведем задачу на доказательство с посылкой $|a| < 1$ и условием

$$\text{сходится}(\lambda_n \left(\sum_{i=1}^{n} (a^m \sin(bn)), \text{натуральное}(n) \right)),$$

а также задачу на описание с условием

$$\text{сходится}(\lambda_n \left(\sum_{i=2}^{n} (\sqrt{i+1} - \sqrt{i})^p \ln \frac{i-1}{i+1}, \text{натуральное}(n) \right)).$$

Эта задача имеет стандартный список целей "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные p". Решатель выдает на нее ответ $0 < p$, число(p).

Для получения разложения в ряд Тейлора служит задача на преобразование с целями "рядтейлора x a", "упростить", "одз". Здесь x - переменная, по которой выполняется разложение; a - выражение, определяющее точку, в окрестности которой происходит разложение. Пример - задача с условием $\sin x + \cos x$, на которую выдается ответ

$$\sum_{n=0}^{\infty} \frac{x^n (-1)^{\left[\frac{n}{2}\right]}}{n!}.$$

Заметим, что бесконечная сумма во внутреннем представлении имеет вид "суммавсех (отображение$(n$ и(целое(n) $k \le n)$ $a(n)))$", где k - значение, с которого начинается суммирование величин $a(n)$.

Если нужно получить лишь конечное число членов разложения в ряд Тейлора, то вместо цели "рядтейлора ..." используется цель "формулатейлора x a n". Здесь x, a - те же, что и выше; n - степень, до которой (включительно) выписываются члены разложения. Пример - задача с условием $\frac{x}{\exp x - 1}$ и целью "формулатейлора x 0 4", на которую выдается ответ $1 - \frac{x}{2} + \frac{x^2}{12} - \frac{x^4}{144}$.

Решатель может обращаться к задаче на получение конечного числа членов разложения в ряд Тейлора при решении других задач, например, при подборе таких значений параметров выражения $f(x)$, что оно приобретает заданный "порядок малости"в окрестности заданной точки. Пример - задача на описание с посылкой "стремится(x

0 0)"(прорисовывается как $x \to 0$), условием $-a\sin x - b\tan x + x = O(x^5)$ и целями "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные ab". На нее выдается ответ $a = \frac{2}{3}, b = \frac{1}{3}$.

При разложении в ряд Фурье используется целевая установка "рядфурье x a b", "упростить", "одз". Здесь x - переменная, по которой ведется разложение; a, b - концы отрезка, на котором происходит разложение. Пример - задача с условием x^3 и целью "рядфурье x $-\pi$ π". На нее выдается ответ

$$2\sum_{n=1}^{\infty}\frac{(-\pi^2n^2+6)(-1)^n\sin(nx)}{n^3}\ .$$

Заметим, что для выбора целевой установки задачи создан специальный интерфейс, так что вводить указанные выше цели "рядтейлора ...", "формулатейлора ..."(или какие-либо другие стандартные комбинации целей) в явном виде вручную нет необходимости.

Наконец, отметим задачи на суммирование рядов. Они оформляются как обычные задачи на упрощение (цели "упростить", "одз"). Пример - задача с условием

$$\sum_{i=0}^{\infty}\frac{x^{4i+1}}{4i+1}\ ,$$

на которую решатель дает ответ

$$\frac{\ln\left|\frac{x+1}{-x+1}\right|}{4}+\frac{\arctan x}{2}\ .$$

3.3.6 Дифференциальные уравнения

В случае дифференциального уравнения в качестве неизвестной выступает функция. Уравнение связывает значение этой функции y в некоторой точке x, значения ее производных различных порядков в этой же точке, и само значение x, причем данная связь имеет место для всех точек некоторой области G. Формальная запись такой связи должна была бы иметь вид $\forall x(x \in G \Rightarrow F(x, y(x), \frac{dy(x)}{dx}, \ldots))$. Однако, используемые при решении дифференциальных уравнений преобразования никак не затрагивают часть этой записи, находящуюся слева от импликации, а также сам квантор общности - они должны бы были просто переписываться "вхолостую". Поэтому в решателе дифференциальное уравнение записывается без внешнего квантора общности, а условия на область, в которой изменяется варьируемая переменная x, вынесены в список посылок. Фактически, это еще одно проявление "контекстной семантики" - для адекватной логической формализации нужно на каждом шаге выполнять обратный переход, преобразуя текущие записи уравнений к указанному выше виду с квантором общности. Чтобы явно указать на наличие такой модифицированной записи, задача на описание, в которой нужно решить дифференциальное уравнение относительно y, снабжается, кроме стандартных целей "неизвестные y", "полный", "явное", "прямойответ", "упростить", "одз", также целью "связка x". Эта цель указывает, что неизвестная y не должна зависеть от параметра x, входящего в посылки. Фактически она несет несколько большую нагрузку, встречаясь только в задачах на решение дифференциальных уравнений и корректируя действия приемов в соответствии с обычными соглашениями, принимаемыми при их решении. Важным таким соглашением является представление ответа не в виде явного равенства для функции

y, а лишь в виде равенства для значения $y(x)$; более того, последнее равенство может даже не быть явно разрешенным относительно $y(x)$. Обычно ответ (для дифференциального уравнения первого порядка) представляется в виде параметрического описания $\exists C(F(x,y(x),C))$ либо дизъюнкции нескольких таких описаний. Как и условие задачи, этот ответ для получения адекватной в логическом отношении записи должен преобразовываться к виду $\exists C(\forall x(x \in G \Rightarrow F(x,y(x),C)))$, причем сама область G (возможно, суженная в процессе решения по сравнению с исходными ограничениями на x, имевшимися в списке посылок)обычно в ответах явно не указывается. В качестве примера приведем задачу на описание с посылкой "число(x)", условиями "функция(y)"и "$x^2\frac{dy(x)}{dx}+y(x)^2 = xy(x)\frac{dy(x)}{dx}$. Эта задача имеет указанный выше список целей. Решатель выдает на нее ответ $\exists C(x\ln(Cy(x))-y(x)=0 \,\&\, \text{число}(C))\vee y(x)=0$.

Если нужно решить дифференциальное уравнение с заданными дополнительными условиями, позволяющими идентифицировать произвольные постоянные, то дополнительные условия просто присоединяются к списку условий задачи на описание. Например, если требуется найти решение дифференциального уравнения $y(x)\frac{d^2y(x)}{dx^2} = 2x\left(\frac{dy(x)}{dx}\right)^2$, принимающее в точке 2 значение 2 и имеющее в этой точке производную 0.5, то дополнительно вводятся условия $y(2)=2$ и $\frac{dy(x)}{d(x=2)} = 0.5$. Решатель выдает на эту задачу ответ $y(x) = -2^{\frac{4}{5}}\sqrt[5]{\frac{x+2}{2(x-3)}}$.

Если для уравнения, не разрешенного относительно производной, ответ получается в параметрической форме: $y = f(t,C); x = g(t,C)$, то такое параметрическое задание записывается решателем в виде $y(g(t,C)) = f(t,C)$. Эта запись позволяет экономить на вводе обозначений для вспомогательных функций $y(t)$,$x(t)$ (ведь $y(t)$ - попросту новая функция, связанная с искомой функцией y лишь косвенным образом). Пример: решая задачу с условием $\left(\frac{dy(x)}{dx}\right)^3 + y(x)^2 = xy(x)\frac{dy(x)}{dx}$, система выдает ответ:
$\exists C\left(y(-\frac{C+3t}{\sqrt{C+2t}}) = -\frac{t^2}{\sqrt{C+2t}} \,\&\, \text{число}(C)\right) \vee \exists C\left(y(\frac{C+3t}{\sqrt{C+2t}}) = \frac{t^2}{\sqrt{C+2t}} \,\&\, \text{число}(C)\right) \vee$
$\vee\, y(x) = 0$.

Для обозначения производной k - го порядка ($k>1$) в дифференциальных уравнениях используется уже встречавшаяся выше запись "частнпроизв(отображение(y число(y) $f(y)$) k x)", прорисовываемая на экране как $\frac{d^k f(x)}{dx^k}$. Ответ дифференциальных уравнений k - го порядка представляестя в виде параметрического описания $\exists c_1 \ldots c_k(\ldots)$ либо дизъюнкции таких описаний (в вырожденных подслучаях длина связывающей приставки может быть меньше k или вообще отсутствовать квантор существования). Так, на уравнение $x^3\frac{d^2y(x)}{dx^2} = \left(-x\frac{dy(x)}{dx}+y(x)\right)\left(-x\frac{dy(x)}{dx}+y(x)-x\right)$ выдается ответ $\exists cd(y(x) = -x\ln(c\ln x + d) \,\&\, \text{число}(c) \,\&\, \text{число}(d)) \vee \exists c(y(x) = cx \,\&\, \text{число}(c))$.

В случае системы дифференциальных уравнений применяется целевая установка того же типа, что и в случае одного уравнения - просто в списке неизвестных указывается несколько переменных.

3.3.7 Аналитическая геометрия и линейная алгебра

Геометрические задачи на вычисление, использующие векторную алгебру либо непосредственно связанные с отысканием неизвестных векторов, формализуются аналогично рассмотренным выше вычислительным задачам по геометрии. Небольшое исключение здесь составляют задачи на решение уравнений в векторных операциях,

формализуемые так же, как задачи на решение уравнений в элементарной алгебре. При использовании в задаче аффинной либо прямоугольной системы координат K в список ее посылок может быть занесено равенство $K = (A, B, C)$ либо $K = (A, B, C, D)$, вводящее обозначения A, B, C, D для начала координат и концов координатных векторов. Этого можно не делать, если каких-либо иных условий на точки A, B, C, D в задаче не накладывается. Во всяком случае, для прямоугольной системы координат требуется вводить посылку "прямкоорд(K)". Приведем два простых примера задач на координаты точек. В первой из них известны координаты (-4,2) точки окружности и координаты (2,0) точки ее касания с осью абсцисс прямоугольной системы координат; найти требуется координаты центра окружности. Эта задача формализуется как задача на описание с целями "полный","явное","прямойответ", "упростить","одз", "неизвестные x", "известно". Она имеет посылки "прямкоорд(K)", "$K = (A, B, C)$", "касательная(прямая(AB) окружность(DE))", "коорд(E, K) = (-4,2)", "$F \in$ прямая (AB)", "$F \in$ окружность(DE)", "коорд(F, K) = (2,0)", "планиметрия"и условие "$x =$ коорд(D, K)". Во второй задаче нужно найти координаты центра вписанной в треугольник окружности, если известны координаты его вершин. Здесь уже не нужно явно вводить обозначения для тройки точек, определяющих систему координат. Посылки задачи имеют вид: "треугольник(ABC)", "вписана(окружность(MN)фигура(набор(ABC)))", "прямкоорд(K)", "коорд(A, K) = (9,2)", "коорд(B, K) = (0,20)", "коорд(C, K) = (-15,-10)", "планиметрия".

Задачи на определение геометрического места точек формализуются в виде задач на преобразование. Условием такой задачи служит выражение "класс(X $P(X)$)", где $P(X)$ - условие на принадлежность точки рассматриваемому множеству (прорисовывается оно как $set_X P(X)$). Задача имеет цели "упростить", "одз", "класс", причем последняя цель указывает на необходимость преобразовать условие к виду, не использующему описателей "класс"и "отображение". В качестве примера рассмотрим задачу на нахождение геометрического места точек, сумма квадратов расстояний которых до двух заданных точек A, B равна $2a^2$; при этом предполагается известным расстояние $2c$ между точками A, B, и $c < a$. Посылки данной задачи суть: "$\neg(A = B)$"; "расстояние(AB) = $2c$"; "$c < a$". Условие имеет вид "set_X(точка(X) & $l(AX)^2 + l(BX)^2 = 2a^2$)". Решатель выдает ответ "окр(доляотрезка($B, A, 1/2$),$\sqrt{a^2 - c^2}$)", то есть окружность с центром в середине отрезка AB и радиусом $\sqrt{a^2 - c^2}$ (см. подраздел "элементарная геометрия"главы 2).

В задачах на уравнения кривых и поверхностей для обозначения уравнения кривой либо поверхности M используется запись вида "коорд(M K) = $set_{xy}(F(x, y) = 0$ & число(x) & число(y))" либо "коорд(M K) = $set_{xyz}(F(x, y, z) = 0$ & число(x) & число(y) & число(z))". В случае кривой в трехмерном пространстве вместо одного равенства $F(x, y, z) = 0$ используются два - $F_1(x, y, z) = 0$ & $F_2(x, y, z) = 0$. В качестве простого примера такой задачи приведем задачу на нахождение уравнения стороны BC треугольника ABC, у которого известны уравнения сторон AB, AC и координаты точки пересечения высот F. Посылки здесь имеют вид: "треугольник(ABC)"; "прямкоорд(K)"; "коорд(прямая(AB)K) = $set_{xy}(x + 3y - 1 = 0$ & число(x) & число(y))"; "коорд(прямая(AC)K) = $set_{xy}(3x+5y-6 = 0$ & число(x) & число(y))"; "перпендикулярно(прямая(BE)прямая(AC))"; "перпендикулярно(прямая(AD)прямая(BC))"; "$F \in$ прямая(BC)"; "$F \in$ прямая(AD)"; "коорд(F K) = $(0, 0)$". Условие задачи - "z = коорд(прямая(BC)K)"; цели - "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные z". Решатель выдает ответ "$z = set_{xy}(39x-9y-4 = 0$ & число(x) & число(y))".

Ряд задач по аналитической геометрии связан с определением взаимного распо-

ложения прямых, точек и плоскостей. Для описания способа такого расположения в логический язык введены несколько специальных предикатных символов и символов констант. Так, при описании взаимного размещения двух прямых A, B используется запись "двепрямые(A B s)", где s - константа для конкретного типа размещения - "пересекаются", "параллельно", "равны". Если в задаче требуется определить, при каких условиях на параметры уравнений имеет место заданный тип взаимного расположения, то она формализуется как задача на описание, имеющая своим условием обозначение типа расположения и цели "полный", "прямойответ", "редакция". Например, для определения условия пересечения двух прямых, одна из которых задана каноническим уравнением, а другая - параметрическим, создается задача на описание с посылками "Прямая(P)", "коорд(P, K) = $set_{xy}(Ax + By + C = 0$ & число(x) & число(y))", "Прямая(Q)", "коорд(Q, K)= $set_{xy}(\exists t(x = at + p$ & $y = bt + q$ & число (t)))" и условием "двепрямые(P, Q, пересекаются)". На нее выдается ответ "$\neg(aA + bB = 0)$".

Задачи на определение вида кривой либо поверхности второго порядка, заданной своим уравнением, формализуются аналогично задачам на качественное исследование вида графика функции вещественного переменного. Они представляются как задачи на описание с целями "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные E", "исследовать", к которым в случае кривых второго порядка добавляется цель "линия", а в случае поверхностей - цель "эллипсоид". Условием служит уравнение исследуемой кривой E. Решение такой задачи происходит путем вывода следствий в ее блоке анализа, с отбором и перенесением в список условий элементов стандартной характеризации кривой. В качестве примера приведем задачу с условием "коорд(E, K) = $set_{xy}(4x^2 - y^2 - 16x - 6y + 3 = 0$ & число(x) & число(y))"и посылкой "прямкоорд(K)". Решатель выдает на нее ответ (за вычетом повторного переписывания уравнения кривой): "гипербола(E)", "каноничкоорд(Q, E)", "$Q = (A, B, C)$", "коорд(A, K) = (2,-3)", "коорд(B, K) = (3,-3)", "коорд(C, K) = (2,-2)", "мнимаяполуось(E) = 2", "действполуось(E) = 1", "коорд(E, Q) = $set_{xy}(4x^2 - y^2 - 4 = 0$ & число(x) & число(y))".

Для матриц в задачах используется два типа обозначений. Если матрица имеет фиксированный порядок n, то она задается явным перечислением своих элементов, при помощи выражения "строки(набор($A_1 \ldots A_n$))", где A_i - выражение вида "набор ($a_{i1} \ldots a_{in}$)", задающее i-ю строку. Фактически при этом матрица вводится и прорисовывается на экране в обычной записи. Если же порядок матрицы - переменная величина, то она задается выражением вида "отображение(ij и(принадлежит(i номера(1 n))принадлежит(j номера(1 n))) $a(i, j)$)". В обоих случаях решатель рассматривает матрицу как функцию от двух переменных, определенных на начальном отрезке натурального ряда, и применяет при работе с ней все приемы, которые связаны с такими функциями (например, при расшифровке записи "значение (M (i, j))", где M - явно заданная матрица). Обычно при задании матриц "общего вида"приходится использовать многоэтажные условные выражения, определяющие для различных частей матрицы различные аналитические зависимости элементов от их номеров. В качестве примера приведем задачу на вычисление определителя матрицы n- го порядка, у которой над главной диагональю расположены элементы, равные номеру столбца; под этой диагональю - равные номеру столбца, взятому с минусом; левый верхний элемент равен 1, а прочие элементы равны 0. Условие этой задачи на преобразование имеет вид "определитель(отображение(ij и(принадлежит(i номера(1 n))принадлежит (j номера(1 n))) вариант($i < j$ j вариант($j < i$ $-j$ вариант($i = 1$ 1 0)))))". Единственная посылка задачи - "натуральное(n)"; цели -

"упростить", "одз". Решатель выдает ответ $n!$.

В задачах на нахождение собственных значений и собственных векторов условия имеют вид "собствзначение(A, x, y)", "собствектор(A, x, z)". Здесь матрица A задается равенством, помещаемым в посылки задачи; x, y, z - неизвестные задачи. Напомним (см. главу 2), что x - собственное значение; y - его кратность; z - собственный вектор, отвечающий данному собственному значению. В ответе собственный вектор z описывается как элемент множества линейных комбинаций некоторого списка векторов. Пример - задача с посылкой

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

и указанными выше двумя условиями. На нее выдается ответ, состоящий из двух подслучаев:

$$x = -2, y = 1, z \in \text{линкомбинации} \left\{ \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\};$$

$$x = 2, y = 3, z \in \text{линкомбинации} \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right\}.$$

Задачи на преобразование полиномиальной матрицы к нормальной диагональной форме и на приведение матрицы к жордановой нормальной форме суть задачи на описание с условиями "каноничматрица(A, k, x)"и "жордформа(A, x, y)"соответственно. Здесь A - явно задаваемая в условии задачи матрица; k - переменная, относительно которой рассматриваются многочлены - элементы матрицы A. x - неизвестная, значением которой служит результат приведения матрицы к соответствующему виду; y - неизвестная, определяющая матрицу, приводящую исходную к жордановой форме. Цели задачи суть "полный", "явное", "прямойответ", "одз", упростить", "неизвестные x"(либо "неизвестные xy").

3.3.8 Комплексные числа

Начиная с данного раздела, следует помнить, что коллекция приемов, занесенных в решатель, была составлена на сравнительно небольшом обучающем материале. Поэтому для многих, даже не очень сложных, задач может быть получен "отказ". С другой стороны, качественное разнообразие рассмотренных задач достаточно велико, чтобы можно было, по аналогии с ними, продолжить начатое обучение. Задачи, на которые получен "отказ", в особенности если они не очень сложны, существенно интереснее и полезнее задач, решаемых данной системой, так как они позволяют научиться создавать свои собственные приемы.

Комплекснозначные операции и элементарные функции вводятся формульным редактором так же, как их веществennозначные аналоги. При создании внутреннего логического представления решатель пытается из контекста определить, какая именно версия рассматривается в задаче. Однако, автоматический выбор версии пока

не очень развит, и в сомнительных случаях лучше контролировать его, переходя к скобочному (текстовому) режиму отображения задачи.

Задачи на упрощение комплекснозначных выражений формулируются аналогично вещественнозначным. Целевая установка - "Упростить в о.д.з.". Вводя для такой установки условие $\arg(7i + 1/3 - 4i)$, получаем после запуска решения ответ $3\pi/4$. Задачи на решение уравнений в комплексных неизвестных требуют лишь добавления условий вида "x - комплексное".

Рассматривались задачи на геометрическую характеризацию множеств точек комплексной плоскости, заданных аналитически. Для формулировки такой задачи вводится посылка "прямкоорд(K)", выделяющая какую-то прямоугольную систему координат K на плоскости. Тип задачи - "описать"; целевая установка состоит из элементов "полный", "явное", "прямойответ", "одз", "упростить", "исследовать", "точки", "неизвестные A". Условия ее имеют вид "множество(A)" и A = точки(вещформа($\mathrm{set}_z P(z)), K$). Здесь $P(z)$ - аналитическое соотношение, задающее множество комплексных чисел z. Смысл последнего условия состоит в том, что определяется множество пар координат точек, соответствующих допустимым комплексным числам, и вводится множество A точек с данными координатами. Целевая установка задачи предопределяет вывод следствий в ее блоке анализа для получения качественной характеризации множества A. Совокупность найденных характеристик составляет ответ.

Рассматривалась также обратная задача - получение аналитического соотношения, задающего множество комплексных чисел с заданными геометрическими характеристиками. В этом случае задача на описание имеет цели "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные x", "известно". Напомним, что такую же целевую установку имели геометрические задачи на вычисление. Условие задачи имеет вид x = комплформа(коорд(Q, K)). В список посылок заносятся утверждение "прямкоорд(K)" и необходимые для уточнения вида множества точек Q дополнительные данные.

Задачи на нахождение пределов числовых последовательностей и на исследование сходимости числовых рядов формулируются так же, как в вещественном случае. Чтобы определить образ множества комплексных чисел при заданном комплекснозначном отображении, создается задача на преобразование выражения вида "образ($\lambda_z(f(z), z -$ комплексное), $\mathrm{set}_z(P(z), z -$ комплексное))". Целевая установка - "Упростить в о.д.з.".

Задачи на комплексное дифференцирование создаются аналогично вещественным. Рассматривались задачи на определение области дифференцируемости функции. Условие такой задачи имеет вид "Дифференцируема($\lambda_z(f(z), z -$ комплексное), x)", где x - неизвестная.

Для восстановления аналитической функции по ее вещественной или мнимой компоненте используется задача на описание, имеющая целевую установку "явное", "прямойответ", "упростить", "неизвестные f A". Ее условия суть: "функция(f)", "аналитическая(f,A)", "вещчастьфунк(f, $\lambda_{xy}(P(x, y), x -$ число & $y -$ число))". В случае восстановления по мнимой части берется запись "мнимчастьфунк".

Задачи на построение конформного отображения представляются как задачи на описание, имеющие целевую установку "полный", "пример", "прямойответ", "одз", "упростить", "неизвестные fA". Условия ее состоят из утверждений "функция(f)", "аналитическая(f A)", "образ(f,$\mathrm{set}_z P(z)) = \mathrm{set}_z Q(z)$.

В случае интегрирования приходится сначала определять в посылках задачи на преобразование ориентированную комплекснозначную кривую C, и лишь затем за-

писывать в условии задачи интеграл вдоль данной кривой. Целевая установка, как и для веществennозначного случая, - "Упростить в о.д.з.".

Задачи на определение области сходимости функционального ряда формулируются так же, как для вещественного случая; вместо символа "сходится" здесь используется символ "Сходится". Для определения радиуса сходимости степенного ряда, коэффициент при n-й степени которого есть $a(n)$, создается задача на преобразование с условием "радиусссходимости($\lambda_n(a(n), n -$ натуральное))".

Для суммирования комплекснозначного ряда формулируется задача на преобразование, в посылки которой заносится утверждение $z \to a$, указывающее, что сумма вычисляется в окрестности заданной точки a. Это утверждение обеспечивает проверку применимости приемов, используемых при суммировании.

Задачи на разложение функции в ряды Тейлора и Лорана формулируются одинаково. Их целевая установка состоит из целей "упростить", "рядтейлора(z a)", "одз". Отличие от веществennозначного случая состоит лишь в том, что добавляется посылка "комплексное(z)". Если нужно получить лишь первые n членов ряда, целевая установка приобретает вид "упростить", "формулатейлора(z a n)", "одз".

Чтобы найти особые точки функции f и определить их тип, вводится задача на описание, имеющая цели "полный", "явное", "прямойответ", "одз", "исследовать", "особыеточки", "функция", "неизвестные(f)". Ее условия суть утверждения "функция(f)" и $f = \lambda_z(A(z), z -$ комплексное).

3.3.9 Теория вероятностей

При вводе в решатель задачи по теории вероятностей приходится вручную осуществлять переход от исходного ее текста к логической формализации. Этот переход не всегда однозначен, и если формализация проведена каким-либо способом, достаточно сильно отличающимся от способов, на которые было ориентировано обучение решателя, то задача будет не решена. Приведенные в задачнике решателя примеры по теории вероятностей сопровождаются исходным их текстом. Чтобы вывести этот текст на экран, достаточно при просмотре задачи нажать клавишу "н". После просмотра нажатие клавиши "курсор влево" или "Esc" возвращает к формальному описанию задачи. Такой же способ используется и для задач по комбинаторике или элементарной физике, где исходное текстовое условие требует существенной переработки для ввода в решатель. Обычно полная формализация задачи в перечисленных разделах бывает достаточно громоздкой, причем пропуск хотя бы одного элемента приведет к тому, что задача решена не будет. Чтобы научиться вводить в решатель такие задачи, рекомендуется ознакомиться с примерами, собранными в его задачнике. Здесь даже правильно введенное условие задачи далеко не гарантирует получение ответа - его можно рассматривать лишь как отправную точку работы по синтезу новых приемов и регулировке решающих правил уже имеющихся приемов. Впрочем, можно все-таки констатировать, что в процессе проработки обучающего материала для перечисленных областей с "текстовыми" формулировками задач нередкими были и случаи, когда решатель находил ответ на вновь введенную задачу самостоятельно.

В этом разделе ограничимся несколькими типичными ситуациями. Прежде всего, рассмотрим задачи на непосредственный подсчет вероятностей. Они характеризуются заданием некоторого конечного множества A, все элементы которого полагаются равновероятными, а также конечного подмножества B, вероятность которого требуется вычислить. Создается задача на описание, имеющая цели "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные p", "известно $a_1 \ldots a_n$", где $a_1, \ldots a_n$

- данные параметры; p - искомая вероятность. Условия задачи суть утверждения "число(p)" и "p = вероятность(B, равновер(A))". Такая формулировка задачи совершенно аналогична формулировке геометрических задач на вычисление. Впрочем, в отличие от геометрии, где основные действия по решению предпринимались в блоке анализа задачи, у задач по теории вероятностей преобладают эквивалентные преобразования условий.

Рассмотрим следующий текстовый пример: "Два стрелка, независимо один от другого, делают по два выстрела (каждый по своей мишени). Вероятность попадания в мишень при одном выстреле для первого стрелка равна p_1, для второго p_2. Выигравшим соревнование считается тот стрелок, в мишени которого будет больше пробоин. Найти вероятность того, что выиграет первый стрелок".

Обозначим через E вероятностное пространство; A, B - события, состоящие в том, что первый стрелок попал в мишень при первом и втором выстрелах; C, D - аналогичные события для второго стрелка. Из текста сразу извлекаются следующие посылки: "вероятность(A,E) = p_1", "вероятность(B,E) = p_1", "вероятность(,E) = p_2", "вероятность(D,E) = p_2", "незавсобытия(набор(A,B,C,D),E)". Обозначаем далее через F событие, состоящее в том, что выиграет первый стрелок. Для него вводим посылку: $F = \mathrm{set}_x(\mathrm{card}(\mathrm{set}_y(y \in \{A, B\}\ \&\ x \in y)) > \mathrm{card}(\mathrm{set}_y(y \in \{A, B\}\ \&\ x \in y)))$. Целевая установка задачи прежняя; условие - "x = вероятность(F,E)". По этой схеме формулируются различные задачи из разделов "Теоремы сложения и умножения вероятностей", "Формула полной вероятности", "Повторение опытов".

Еще один текстовый пример, относящийся к случайным величинам: "При работе некоторого прибора в случайные моменты времени возникают неисправности. Время T работы прибора от его включения до возникновения неисправности распределено по показательному закону с параметром m. При возникновении неисправности она мгновенно обнаруживается, и прибор поступает в ремонт. Ремонт продолжается время p, после чего прибор снова включается в работу. Найти плотность распределения и функцию распределения промежутка времени T^* между двумя соседними неисправностями. Найти его математическое ожидание и дисперсию. Найти вероятность того, что время T^* будет больше $2p$".

Обозначаем вероятностное пространство через B; случайную величину, равную времени работы прибора от его включения до возникновения неисправности - A; случайную величину, равную времени между двумя соседними неисправностями - C. Из текста извлекаются следующие посылки: "случвеличина(A,B)"; "плотнраспред(A,B) = $\lambda_t((0$ при $t < 0,$ иначе $m\exp(-mt)), t -$ число)"; $C = A + \lambda_x(p, x \in$ элемсобытия(B)); $0 < m$; $0 < p$. Условия задачи суть: "x = плотнраспред(C,B)", "y = функраспред(C,B)", "z = вероятность(прообраз(C,$(2p, \infty)$),B)", "M = матожидание(C,B)", "D = дисперсия(C,B)".

3.3.10 Элементарная физика

Ввод в решатель новой задачи по элементарной физике обычно требует расширения логического языка и базы приемов. Даже в случаях, где языковых возможностей достаточно и решатель мог бы решить задачу самостоятельно, легко упустить из виду необходимую для решения подробность или воспользоваться таким, внешне корректным, способом записи, для которого приемы еще не создавались. Поэтому рекомендуется ознакомиться с тем, как формализованы условия задач, имеющихся в задачнике, и вводить новые условия "по аналогии". Отступления от данного правила, конечно, возможны, и даже желательны, но лишь в контексте продолжения

обучения решателя. В физике, как, впрочем, и в других разделах, при обучении работал принцип наименьших обобщений, позволявший в какой-то мере бороться с ненужными действиями, однако существенно ослаблявший возможности решателя на момент проработки недостаточного объема обучающего материала. Этот принцип, в конечном счете, приводит к получению некоего пограничного между отказами и перебором состояния базы приемов. Несмотря на не очень высокое качество, получаемая база приемов позволяет почувствовать специфику управления логическими процессами в предметной области и представляет собой ценный материал для последующего анализа и создания более совершенных версий.

Кроме проблем развития языка и базы приемов, физические задачи заставляют искать и новые решения по интерфейсу их ввода. Ясно, что для "обычного" пользователя режим детальной логической формализации условий совершенно непригоден. Вероятно, здесь понадобится развивать какие-либо средства для наглядного отображения сценариев в виде схем, диаграмм и т.п. Однако, с точки зрения изучения логических процессов, это второстепенные обстоятельства, и мы их пока затрагивать не будем. Другая возможность - ввод условий физических задач на естественном языке. Пока она рассматривается лишь как обучающий материал для процедур семантического анализа. Масштабы работы, которую здесь придется проделать, весьма велики, и для создания практически пригодной версии понадобится время.

Разберем несколько типичных примеров формализации физических задач. Начнем с простейшего примера - по существу, текстовой арифметической задачи "на движение": "Два лица выезжают одновременно из городов A и B навстречу друг другу. Первый проезжает в час на 2 км больше второго и приезжает в B часом раньше, чем второй в A. Расстояние между A и B равно 24 км. Сколько километров проезжает каждый в час?".

Задача оформляется как задача на описание; целевая установка ее - того же типа, как у геометрических задач на вычисление. Посылка "движение(a, b, c)" указывает на то, что a есть процесс движения первого лица, обозначенного b, по траектории c. Движению второго лица q соответствует посылка "движение(p, q, r)". Так как траектории этих двух процессов движения являются взаимно обратными, вводится посылка "r =обратныйпуть(c)". Посылка "исхмомент(a)=исхмомент(p)" выражает тот факт, что движение началось одновременно. Связь между скоростями движения вдоль траекторий дается посылкой "скорость(a, c) = скорость(p, r)+ 2 км/час". Связь между моментами завершения движения дается посылкой "послмомент(a)= послмомент(p) - 1 час". Длина траектории определяется равенством "длина(c) = 24 км". Условия задачи суть "x = скорость(a, c)"и "y = скорость(p, r)". Неизвестные задачи - x, y. Решатель выдает ответ "x = 8км/час, y = 6км/час".

Следующий пример - задача по динамике: "На длинной нити, перекинутой через блок, подвешены на одном уровне одинаковые грузы. От одного из грузов отделяется часть, масса которой равна 1/5 массы груза, и через 1 сек. падает на землю. Через какое время после этого достигнет земли другой груз? ".

Составляем список посылок, задающих данный сценарий. Прежде всего, вводим посылку "блок(a,b,c)". Здесь a - процесс движения блока; b, c - процессы движения грузов. Считаем, что эти процессы начинаются в тот момент, как началось падение отделившейся части. Поэтому один из движущихся грузов уже неполный. Судя по условию задачи, блок закреплен, так что добавляем посылку "неподв(a, P)", где P - обозначение блока как физического тела. Чтобы можно было сослаться на сами грузы, вводим посылки "движение(b,d,e)" и "движение(c,f,g)", где эти грузы обозначены через d,f. Одновременно получаем обозначения e,g для траекторий

движения грузов. Чтобы указать на то, что события развертываются в поле действия земного тяготения, добавляем посылку "прямкоорд(K)". Далее можно использовать прямоугольную систему координат K, ориентированную стандартным образом относительно поверхности Земли. Так как грузы подвешены, вводим посылки "подвеска(b,a)", "подвеска(c,a)". Записываем условие на массы грузов, считая, что часть отделилась от первого из них: "масса(d) = 4масса(f)/5". Условие нахождения грузов на одном уровне формулируем с использованием системы координат K: "крд(Место(b, исхмомент(b)),К,3) = крд(Место(c, исхмомент(c)),К,3)". В исходный момент грузы неподвижны. Косвенная информация об этом извлекается из слова "подвешены", однако заметим, что такую подробность при беглом чтении текста задачи легко и пропустить. Формулируем соответствующие посылки: "Скорость(b, K, исхмомент(b)) = вектор0"; "Скорость(c,K,исхмомент(c)) = вектор0". Еще одна важная деталь - равенство исходных моментов процессов движения грузов: "исхмомент(b) = исхмомент(c)".

Далее переходим к движению отделившейся части. Вводим посылку, определяющую процесс ее движения: "движение(p,q,r)". Эта часть движется только под действием силы тяжести, поэтому добавляем посылку "бросок(p)". Можно было указать на отсутствие горизонтальной составляющей скорости и добавить также посылку "вертикаль(r,K)", однако в данной задаче эта информация не нужна, и на поведении решателя она практически никак не сказывается. В начальный момент высоты и скорости груза и отделившейся части совпадали: "крд(Место(p, исхмомент(p)),К,3) = крд(Место(b, исхмомент(b)),К,3)"; "Скорость(p,K,исхмомент(p)) = вектор0". Длительность процесса падения p равна секунде, и в последний момент высота падающей части равна 0: "длительность(p) = 1сек", "крд(Место(p,послмомент(p)),K,3) = 0". Так как нам нужно найти время достижения земли грузом f, то считаем, что в последний момент его движения высота равна 0: "крд(Место(c послмомент(c)), K, 3)=0".

После ввода посылок остается лишь ввести целевую установку (аналогично предыдущей задаче) и добавить условие "x = длительность(c)".

На этом примере уже хорошо видно, насколько более громоздкой, по сравнению с текстом, оказывается логическая запись задачи. Вместе с тем, ничего лишнего мы не добавили - все утверждения необходимы для адекватной характеризации сценария. Как уже говорилось, предпринимаются попытки обучения решателя самостоятельному чтению текста и переводу его на логический язык. Логика восстановления элементов задачи "по умолчанию", в общем, реализуется в виде приемов без особых трудностей. Однако, приемов понадобится очень много - эта работа равнозначна созданию в решателе достаточно полной системы представлений о мире. Безусловно, выполнить ее все равно придется, и тогда откроются новые перспективы сразу во многих важных направлениях. До этого же упрощение ввода в решатель задач "текстового" характера придется обеспечивать какими-то другими средствами.

В заключение приведем пример на закон сохранения энергии: "С горки высотой 2 м и длиной основания 5 м съезжают санки, которые останавливаются, пройдя горизонтально некоторый путь от основания горы. Чему равен этот путь, если коэффициент трения на всем пути 0.05?".

Движение санок с горки выражаем посылкой "движнаклплоск(a,b,K,p)". Здесь a - процесс движения санок; b - процесс движения наклонной плоскости; K - прямоугольная система координат, связанная с поверхностью Земли; p - угол наклона плоскости. Для уточнения положения K добавляем посылку "поверхнземли(K)"; для указания на то, что горка неподвижна - посылку "неподв(b,B)". В исходный момент санки

неподвижны: "Скорость(a,K,исхмомент(a)) = вектор0". Условия на высоту горки и длину ее основания выразим через координаты санок в начале и конце первой части пути: "крд(Место(a, исхмомент(a)),К,3) - крд(Место(a, послмомент(a)),К,3) = 2м"; "крд(Место(a, исхмомент(a)),K,1) - крд(Место(a, послмомент(a)),К,1)=5м".

Чтобы задать коэффициент трения, вводим в рассмотрение процесс f силового воздействия горки на санки: "воздействие(f,a,b)"; "коэффтрения(f, Период(a)) = 0.05".

Вводим процесс движения санок по горизонтальному пути. Представляем его как процесс движения по наклонной плоскости с уклоном 0: "движнаклплоск(c,d,K,0)"; "неподв(d,D)". Чтобы связать процессы движения a и c, заносим посылку "продолжбезудара(a,c)", которая означает отсутствие потерь энергии на стыке двух процессов, а также обеспечивает привязку по координатам и скорости.

Для указания коэффициента трения на горизонтальном участке вводим соответствующее силовое воздействие: "воздействие(h,c,d)", "коэффтрения(h, Период(c)) = 0.05".

Чтобы указать, что на всей горизонтальной траектории санки двигались, а остановились лишь в ее конце, вводим посылки "движется(c,K)", "Скорость(c,K, послмомент(c)) = вектор0". Наконец, для ссылки на горизонтальную траекторию добавляем посылку "движение(c,m,r)". Условием задачи служит равенство "x = длина(r)".

Иногда решение задачи можно ускорить или обеспечить получение ответа за счет ввода какого-либо дополнительного утверждения, легко усматриваемого из текста задачи. В последнем примере можно было бы указать, что направление движения санок в начале горизонтальной траектории продолжает направление их движения с горки: "крд(Скорость(c,K,исхмомент(c)),К,1) $<$ 0". Это немного ускорило бы появление ответа. Такого рода "усиление" решателя особенно эффективно для планиметрических задач. Обычно подсказка, помогающая решателю, указывает на тот или иной пробел в его базе приемов, и при ее обнаружении следует данный пробел устранить.

3.3.11 Вычислительные задачи

Начато обучение решателя задачам "на программирование" - составлению программ для вычислений с традиционными нелогическими структурами данных (числами, матрицами, и т.п.). В процессе решения задачи создается логическое представление вычислительного алгоритма, которое затем преобразуется компилятором ГЕНОЛОГа в программу на ЛОСе. Если задача не имела варьируемых параметров, то составленная программа сразу же запускается и выдается найденный ответ (набор чисел, график функции и т.п.). В противном случае после создания программы открывается специальный интерфейс для ввода значений параметров. Он позволяет многократно обращаться к программе для варьируемых значений. Синтезированная программа сохраняется, и ее можно запускать без повторного решения задачи.

Способность решателя создавать собственные программы можно будет использовать для различных целей. Во-первых, для ускоренного развития в рамках логической системы больших вычислительных комплексов, отдельные блоки которых будут определяться на математическом языке формулировок соответствующих задач. Во-вторых, для освоения смешанных логико-вычислительных режимов работы решателя. Вычисления, проводимые в процессе аналитического исследования, могут подсказать какие-либо гипотезы, позволить вывести приближенные зависимости и с их помощью довести задачу до конца, и т.п. В-третьих, компиляция вычислений непосредственно с уровня теорем логической системы позволит ускорить накопление

библиотек вычислительных процедур. Она же необходима для исследований по логическому выводу в теории, ориентированному на вычислительные задачи. Последнее является естественным дополнением исследований по логическому выводу в теории, ориентированному на создание новых приемов.

Хотя число вычислительных задач, на которых выполнялось обучение решателя, совсем невелико (около 30), однако они и не требовали слишком большого разнообразия приемов. Поэтому обычно решатель начинал работать после одного - двух освоенных примеров задач заданного типа. Пока создаются лишь простейшие схемы вычислений, работающие в более-менее регулярных ситуациях; для получения более изощренных процедур обучение решателя должно быть продолжено.

Прежде всего, рассматривались задачи на построение графика функции. Заметим, что для этого существует альтернативная возможность, указанная ниже при описании интерфейса ввода задач. В нашем случае, для построения графика функции $f(x)$ создается задача на описание с единственным условием $y = \lambda_x(f(x), x \in [a, b])$. Здесь a, b - либо конкретные числа, либо параметры. Задача имеет цели "полный", "явное", "прямойответ", "одз", "упростить", "вычисление", "неизвестные y", "известно". Данный комплект целей является стандартным для вычислительных задач; на экране он отображается в виде фразы "Вычислить ... при заданных ...".

Задача на вычисление определенного интеграла I, быть может, содержащего параметры, задается аналогично предыдущему; ее единственное условие имеет вид $y = I$.

Был рассмотрен пример, когда зависимость, для которой нужно написать программу, определяется после решения задачи по физике. В посылках задачи перечисляются, как это было продемонстрировано в предыдущем разделе, утверждения, описывающие физический сценарий. Затем вводится условие вида $y = \lambda_x(F(x), x \in [a, b])$. Решатель находит ответ физической задачи, после чего определяется выражение $F(x)$. Затем он составляет логическую схему вычислений - цикл вычисления значения $F(x)$ с тем шагом, который требуется для построения графика. Наконец, компилятор создает программу вычислений, с помощью которой и строится график. Хотя приведенная в задачнике ситуация очень проста (бросок под углом к горизонту), пример демонстрирует интересную принципиальную возможность системы. Аналогичные задачи можно было бы создать и для других рассмотренных при обучении решателя сценариев.

Если нужно на одном графике отобразить несколько различных зависимостей, вводится несколько соответствующих условий задачи.

Чтобы составить программу численного решения обыкновенного дифференциального уравнения первого порядка методом Рунге-Кутта, вводятся следующие условия задачи:

$$y(a) = b; \frac{dy(x)}{dx} = f(x, y(x)).$$

Иногда к ним добавляются дополнительные ограничения на параметры, которые могут понадобиться на этапе предварительных преобразований формул. Аналогичным образом создается задача для решения системы обыкновенных дифференциальных уравнений. Здесь применяется метод Эйлера.

Если нужно найти множество всех корней уравнения в некоторой подобласти, вводятся условия: $y = \mathrm{set}_x(f(x) = g(x)\ \&\ P(x))$, "точность($y$, a)". Здесь $P(x)$ - условие, определяющее область, в которой ищутся корни; a - параметр, задающий точность вычислений - как только в итеративной схеме уточнения корней значения y начинают отличаться меньше, чем на a, цикл обрывается. В случае системы уравнений первое

условие приобретает вид $y = \mathrm{set}_z(F_1\ \&\ \ldots\ \&\ F_n\ \&\ P(z))$, где z - список неизвестных; $F_1, \ldots, F_n$ - уравнения; $P(z)$ - условие на рассматриваемую подобласть. Заметим, что для случая системы уравнений программа составляется лишь тогда, когда решателю удается выразить аналитически все неизвестные через одну оставшуюся. Здесь обычно получается полный список корней системы в указанной области. Если же таким способом система не решается, можно поставить задачу на создание программы уточнений исходного значения по методу Ньютона. Здесь система уравнений заносится в посылки и вводятся условия $y = (x_1, \ldots, x_n)$, "точность(y, a)", "уточнение(y, y_0)", где $x_1, \ldots, x_n$ - все неизвестные; y_0 - начальная точка.

3.4 Интерфейс ввода задач

3.4.1 Оглавление задачника

В решателе имеется сборник задач, в котором сохраняются (для нужд тестирования и регулировки) рассматривавшиеся при обучении задачи либо вводятся новые задачи. Этот сборник снабжен древовидным оглавлением, разбивающим его на разделы, подразделы, и т.п. Вход в оглавление задачника осуществляется из главного меню при нажатии клавиши "з"(здесь и далее, если не оговорено особо, все буквенные названия клавиш - русские) либо нажатии левой кнопки мыши на пункте меню "Оглавление задачника". После этого происходит переход к тому пункту оглавления, с которым велась работа при предыдущем пребывании в задачнике. Переход между пунктами оглавления выполняется с помощью клавиш курсора: нажатие клавиши "курсор влево"приводит к переходу в надраздел текущего раздела; клавиши "курсор вверх"и "курсор вниз"позволяют выбирать текущий пункт в текущем разделе; клавиша "курсор вправо"приводит к просмотру содержимого текущего пункта. Для ускоренного выбора пункта можно использовать курсор мыши: нажатие левой клавиши на выбранном пункте приводит к переходу внутрь этого пункта; нажатие (в любом месте) правой клавиши мыши - к возвращению в надраздел. Пункты каждого раздела пронумерованы, причем если пункт соответствует входу в подраздел, то после его номера стоит закрывающая скобка, если же пункт соответствует отдельной задаче задачника, то после его номера стоит точка. Из любого пункта оглавления можно вернуться к главному меню, нажав клавишу "End". Пункты корневого раздела имеют названия "Дискретная математика", "Алгебра множеств", "Элементарная алгебра", "Элементарная геометрия", "Математический анализ", "Дифференциальные уравнения", "Аналитическая геометрия", и др.

Оглавление задачника организовано таким образом, что все его концевые пункты (соответствующие отдельным задачам) собраны в специальные концевые разделы. Все пункты такого концевого раздела - только концевые. Обычно в концевом разделе перечисляются только номера задач, после которых стоят три тире. Прочие (неконцевые) пункты оглавления имеют текстовые подзаголовки. В принципе, с любым пунктом оглавления (концевым либо неконцевым) можно связать произвольные текстовые примечания. Для этого достаточно выбрать нужный пункт и нажать клавишу "р" (рус.) - возникает курсор текстового редактора (особенности текстового редактора решателя - см. в последующих разделах, посвященных его интерфейсу). После завершения редактирования (оно происходит по нажатии "Enter") сохраняется измененный текст пункта.

После выбора текущего концевого пункта и нажатия клавиши "курсор вправо"

происходит переход к просмотру задачи этого пункта. Все задачи одного и того же концевого раздела соединены при таком просмотре в одну "ленту", которую можно прокручивать вверх и вниз при помощи клавиш "курсор вверх - вниз" и "Page Up - Down", не выходя обратно в оглавление задачника. Одна задача отделяется на этой "ленте" от другой сплошной горизонтальной линией. Кроме указанных выше, при просмотре задач одного концевого раздела удобно использовать клавиши "Ctr-Page Up" и "Ctr-Page Down". Первая из них позволяет переходить к просмотру начала предыдущей задачи, вторая - к просмотру начала следующей задачи.

3.4.2 Просмотр задач и основные операции над задачами

Область между горизонтальными линиями, ограничивающими описание задачи (для последней задачи нижняя линия отсутствует), называется полем задачи. Элементы задачи размещаются в ее поле следующим образом. Вначале идут посылки задачи. Затем располагается текст-формульный элемент, задающий целевую установку задачи. Далее (кроме задач на исследование) идут условия задачи (в случае задач на доказательство либо преобразование - единственное условие). Если задача ранее была решена системой, то в конце располагается найденный на нее ответ. Задача по геометрии может быть сопровождена чертежом, который размещается до ее посылок. Если задача не имеет невырожденных посылок (отличных от константы "истина" и простейших указателей на типы значений переменных), то целевая установка размещается непосредственно в начале поля.

В некоторых разделах (теория вероятностей, элементарная физика, комбинаторика) логическая формализация текста задачи выглядит громоздко и трудна для восприятия. Поэтому в решателе предусмотрена возможность сохранять также исходный текст задачи. Если этот текст был сохранен, и при просмотре задачи нажимается клавиша "н", то текст появляется на экране. Данная операция адресуется той задаче, к которой относится верхняя из имеющихся на экране отделяющих горизонтальных линий. Чтобы вернуться к просмотру формальной версии условия, нажимается "End" или "Esc". При нажатии клавиши "н" формальная запись задачи пропадает с экрана. Поэтому более удобным для ее сопоставления с исходным текстом задачи является другой режим, включаемый при нажатии на клавишу "+" или выборе в верхнем меню пункта "+". Тогда в нижней части экрана возникает выделенный голубым цветом текст задачи. Как и выше, он относится к задаче, определяемой по верхней горизонтальной линии на экране. Далее можно выполнять произвольную прокрутку - этот текст будет сохраняться до нажатия клавиши пробела.

Чтобы удобнее было анализировать логическую формализацию задачи, можно использовать специальный режим просмотра, при котором на экран будут выводиться пояснения к встречающимся в формулах понятиям. Для перехода в этот режим достаточно нажать клавишу "?" либо выбрать в верхнем меню пункт "?". Чтобы получить пояснения к какому-либо понятию, нужно поместить на него курсор мыши и нажать левую кнопку. Пояснения прорисовываются голубым цветом в нижней части экрана и исчезают при нажатии любой клавиши. Чтобы выполнять описываемые далее операции над задачами, связанные с выделением их элементов, следут сначала выйти из режим просмотра пояснений, для чего нажать клавишу пробела.

Независимо от режима просмотра пояснений к задаче, можно входить в оглавление логического языка системы из произвольного оглавления либо из общего просмотра списка задач. Для этого достаточно нажать клавишу F2.

Существуют различные режимы прорисовки задачи. По умолчанию, задачи про-

рисовываются с помощью стандартной математической записи и с пропуском ряда простейших посылок и условий. Чтобы перейти к режиму, в котором используется внутренний логический язык (далее называемому скобочной записью), нажимается клавиша "Ctr-c"; она же возвращает обратно в режим стандартной математической записи. Для перехода в режим полного просмотра посылок и условий и выхода из этого режима служит клавиша "ы".

Для выполнения различных операций с задачей и ее элементами выполняется выбор задачи либо ее элементов с помощью мыши. Курсор мыши подводится к выбираемому элементу либо помещается в поле выбираемой задачи. При выборе элемента задачи нажимается левая клавиша мыши, при выборе всей задачи - правая. Чтобы произошел выбор задачи, на экране обязательно должна быть прорисована ее верхняя отделяющая линия. Выбранный элемент перекрашивается в голубой цвет. Чтобы отменить выбор элемента, выполняется повторно такое же нажатие клавиши мыши. Если нужно отменить выбор сразу всех выбранных на текущий момент элементов и задач, нажимается клавиша пробела. Можно выделять не только отдельную посылку либо условие задачи, но и ее фрагмент. Для этого, сразу после выделения посылки либо условия F, следует нажать клавишу "курсор вправо". Тогда будет выделен (перекрашен в голубой цвет) первый операнд F. Далее клавишами "курсор вправо - курсор влево" можно переходить от операнда к операнду на одном уровне; клавишей "курсор вниз" - переходить к подоперандам текущего операнда; клавишей "курсор вверх" - возвращаться к надоперанду. Нажатие любой клавиши, отличной от клавиш курсора, завершает данный режим выделения фрагмента.

Перечислим некоторые операции, которые можно совершать после выделения терма задачи. Прежде всего, нажатие клавиши "с" позволяет переходить от стандартной математической записи этого терма к скобочной и обратно. При нажатии клавиши "Enter", включается режим вставки формульным редактором нового терма задачи непосредственно перед выделенным термом (если он располагается до целевой установки, то становится посылкой задачи, если после - условием). Ниже будут приведены основные правила использования формульного редактора при вводе элементов задачи. Если нужно вставить новый терм перед выделенным термом с помощью текстового редактора (то есть в скобочной записи), то нажимается клавиша "Т" (русск.). Для изменения ранее набранного терма имеется три возможности. Первая из них - использование текстового редактора для работы с текстом скобочной записи этого терма. Для входа в этот режим нажимается "Ctr - т" (т - русск.). Вторая - использование формульного редактора для повторного набора новой версии терма. Новая версия набирается непосредственно под старой, которая во время набора сохраняется, а затем удаляется. Для входа в этот режим используется "Ctr-ф". Наконец, можно войти в формульный редактор для продолжения набора выделенного терма, начиная с его конца. При необходимости (используя Backspace) здесь можно сначала исключить часть ранее набранных последних символов, а затем добавить новые. Для входа в такой режим нажимается "ф". Чаще всего для изменения ранее введенного терма применяется первый режим (скобочной записи), так как он позволяет вносить изменения в любом месте текста. Чтобы удалить выделенный элемент задачи, нажимается клавиша "Ctr-Del". Если была выделена вся задача, то нажатие "Ctr-Del"удаляет всю задачу целиком. Выделенную задачу можно скопировать нажатием клавиши "к"(русск.). Если выделена только одна задача, то копия располагается в конце текущего списка задач (т.е. текущего концевого раздела оглавления задачника). Если нужно разместить эту копию перед некоторой ранее введенной задачей списка (текущего либо относящегося к другому разделу), то перед нажатием

клавиши "к"выделяется также последняя задача. Чтобы исключить из задачника ранее найденный на выделенную задачу ответ, нажимается клавиша "Ctr-F4". Выделенный терм задачи можно скопировать, выделив предварительно тот терм, перед которым нужно поместить копию (если ее нужно поместить в конец списка посылок либо списка условий, то выделяется вся задача) и нажав затем клавишу "Insert". Аналогичным образом можно перенести выделенный терм на новое место; для этого служит клавиша "Ctr-Insert".

Если выделить несколько термов либо фрагментов термов задачи, то их можно использовать при наборе новых термов формульным редактором. Следует лишь запомнить порядок, в котором онм выделялись. Если в режиме формульного редактора нажать сначала "Insert", а затем номер (начиная с 1 и не более 9) выделенного указанным образом терма, то произойдет автоматическая вставка его в набираемый терм начиная с текущей позиции. Такую вставку заданного терма в процессе набора можно выполнять многократно.

3.4.3 Ввод новой задачи

Ввод новой задачи инициируется одним из двух способов. Первый из них состоит в том, чтобы перейти в конец просматриваемого списка задач и нажать клавишу "з"(если нужно ввести новую задачу перед ранее введенной, то сначала выделяется последняя задача, а затем нажимается "з"). Тогда возникает новая горизонтальная линия, которая является первым элементом новой задачи. Второй способ начать ввод новой задачи - выйти из просмотра списка задач в концевой раздел оглавления задачника, соответствующий данному списку, и ввести новый его концевой пункт, который автоматически становится заготовкой новой задачи. Если этот пункт вводится в конце раздела, то нажимается клавиша "к"(русск.); если же его нужно ввести перед некоторым другим пунктом, то происходит выбор последнего (клавишами "курсор вверх- вниз") и нажимается "К". При входе в список задач через новый концевой пункт оглавления попадаем на начало новой задачи, которое состоит из единственного ее элемента - верхней горизонтальной отделяющей линии. Если новая задача вводится перед ранее введенной нажатием клавиши "з", то она автоматически при этом выделяется (ее верхняя линия окрашена в голубой цвет).

После того, как введена верхняя отделяющая линия задачи, можно переходить к вводу посылок, условий и целевой установки задачи. В принципе, их допустимо набирать в произвольной последовательности. Рекомендуемым является следующий порядок, при котором уменьшается число операций с клавиатурой: сначала вводятся посылки задачи, затем целевая установка, и в конце - условие (условия). Чертеж геометрической задачи обычно вводится в последнюю очередь либо формируется автоматически (путем нажатия клавиши "Ч"), хотя можно начинать ввод задачи и непосредственно с чертежа. Для ввода очередной посылки нажимается "Enter чтобы войти в формульный редактор,- либо "Т"(русск.) - чтобы войти в текстовый редактор (последнее выполняется лишь в исключительных случаях, если для рассматриваемых логических символов еще не обеспечена их прорисовка формульным редактором). Вырожденные посылки, указывающие тип значений переменных либо необходимые для указания области допустимых значений, вводить вообще не нужно - они создаются решателем автоматически (кроме случаев, когда тип значения переменной не подсказывается выражениями из описания задачи, в которых эта переменная встречается). По окончании ввода посылок (если посылок нет, то сразу же) начинается формирование целевой установки задачи. Здесь имеется два режима

- с использованием оглавления типов целевых установок и путем непосредственного набора перечня целей текстовым редактором. Рекомендуется использовать первый режим. Для входа в него нажимается клавиша "ц". Эта же клавиша используется для изменения ранее введенной целевой установки - предварительно надо выделить задачу либо убедиться, что задача является последней в текущем списке задач. После нажатия клавиши "ц"на экране появляется раздел оглавления типов целевых установок, с которым происходила работа в предыдущий раз. Это оглавление (и вообще все оглавления решателя) устроено аналогично оглавлению задачника. Корневой раздел оглавления типов целевых установок содержит следующие пункты:

1) "Доказать утверждение". Этот пункт выбирается при создании задачи на доказательство. После выбора его ("курсор вверх-вниз") и нажатия "курсор вправо" автоматически выбирается тип задачи "доказать"и происходит возвращение в набор текста задачи.

2) "Проверить истинность утверждения". Здесь создается задача на описание, предназначенная для проверки истинности ее единственного условия. Цели ее суть "полный", "явное", "прямойответ", "одз". Возможные ответы (кроме отказа) - "истина"либо "ложь". Как и выше, для выбора данной целевой установки нажимается "курсор вправо".

3) "Преобразовать выражение". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на преобразование (см. ниже).

4) "Найти значения неизвестных". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на описание (см. ниже), связанных с нахождением неизвестных.

5) "Исследовать свойства объекта". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на общую характеризацию свойств объектов различных типов. Все они оформляются как задачи на описание, причем ответом служит некоторая группа утверждений, полученных в процессе анализа ситуации и дающих типовую характеризацию требуемого вида.

6) "Переформулировать условие". Создается задача на описание, имеющая цели "полный", "прямойответ", "редакция". Это - эквивалент словесной формулировки "найти условия, при которых ...", означающей необходимость эквивалентного преобразования исходной системы условий к более простому и явному виду.

7) "Построить график". Пункт позволяет создавать задачу на построение графика. Это - задача на преобразование с целями "числзначение", "график". Ее условие - выражение, определяющее зависимость, для которой строится график. При решении задачи инициируется диалог для определения варьируемой переменной, промежутка ее значений и значений фиксированных параметров. Для работы с возникающим после этого графиком предусмотрен специальный интерфейс, позволяющий изменять масштаб, параметры и область просмотра (подробнее о нем - в нижеследующих разделах). После выбора данного пункта и нажатия клавиши "курсор вправо"происходит возвращение в набираемую задачу, где прорисовывается строка "Построить график:".

8) "Игровые задачи". Пункт является подразделом оглавления, который на текущий момент используется для обучения решателя игре в шахматы. Предусмотрена инициализация шахматной партии либо создание заданной позиции для ее анализа решателем. Обучение находится на начальном этапе, и использовать эту возможность интерфейса рекомендуется лишь тем, кто захотел бы самостоятельно продолжить развитие решателя.

Подраздел "Преобразовать выражение"оглавления типов целевых установок содержит следующие пункты:

1) "Упростить выражение в области допустимых значений". Этот пункт позволяет вводить задачу на преобразование с целями "упростить","одз". Такой набор целей - стандартный для большинства задач на преобразование, включая вычисление пределов, производных, интегралов и определителей матриц.

2) "Вычисление значения константного выражения". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на нахождение точного либо приближенного численного значения константного выражения (см.ниже).

3) "Раскрыть скобки". Создается задача на преобразование, ориентированная на приведение выражения к виду суммы одночленов. Она имеет цели "упростить", "раскрытьскобки", "одз".

4) "Разложить на вещественные множители". Создается задача на преобразование, в которой нужно разложить выражение на вещественные множители. Она имеет цели "упростить", "разложитьнамножители", "одз".

5) "Разложить на комплексные множители". Аналогично предыдущему, но допускается переход к выражениям с мнимой единицей. Задача имеет те же цели, что и выше, к которым добавляется цель "видУмножение". В случае появления комплексных множителей вместо вещественной операции "умножение"возникает комплексная операция "Умножение".

6) "Представить в виде суммы простейших дробей". Вводится задача на преобразование, в которой нужно представить выражение в виде суммы простейших дробей. Предполагается, что это выражение имеет единственную переменную (в действительности процедуры решателя рассчитаны на приведение к виду суммы простейших дробей выражения с несколькими переменными, относительно явно указанной одной из них, что применяется, например, при интегрировании, однако это - "внутренняя" возможность, не вынесенная во внешний интерфейс). Цели задачи - "упростить", "простейщиедроби", "одз".

7) "Разложить в ряд степенной ряд". Создается задача на разложение заданного выражения по заданной переменной x в степенной ряд в окрестности заданной точки t. Здесь имеется в виду получение бесконечной суммы, содержащей все члены ряда. Если в посылках задачи содержится утверждение "комплексное(x)", то предпринимается попытка получить разложение в комплекснозначный ряд Тейлора либо Лорана. После выбора данной целевой установки ("курсор вправо") происходит возвращение в текст набираемой задачи, к которому добавляется строка "Разложить в степенной ряд по переменной". Под этой строкой в левой части экрана размещается курсор формульного редактора, которым нужно ввести переменную разложения x. После ввода этой переменной (ввод завершается нажатием "Enter") к указанной строке добавляется "x в точке", и снова в левой части возникает курсор формульного редактора. Здесь уже нужно ввести выражение t, определяющее точку разложения. По окончании ввода (нажатие "Enter") t перерисовывается в конце строки, задающей целевую установку, и эта установка завершается двоеточием - далее можно вводить выражение, которое следует разложить в ряд. Такого рода диалоги используются в различных описываемых далее целевых установках. Цели задачи на разложение в степенной ряд - "упростить", "рядтейлора x t", "одз".

8) "Получить заданное число членов разложения в ряд Тейлора". Аналогично предыдущему, но требуется получить заданное конечное число n членов ряда. Диалог ввода целевой установки происходит по тому же сценарию, что и в предыдущем

пункте. Однако, после ввода выражения t для точки разложения, к строке текста целевой установки добавляются слова "до членов степени", и далее вводится формульным редактором константа n. Цели получаемой задачи - "упростить", "формулатейлора x t n", "одз".

9) "Разложить в ряд Фурье". Вводится задача на разложение заданного выражения по заданной переменной x в ряд Фурье на отрезке $[a, b]$. Вначале диалога ввода целевой установки прорисовывается строка "Разложить в ряд Фурье по переменной", после чего формульным редактором вводится переменная разложения x. Далее (после x) к строке добавляются слова "на отрезке", и формульным редактором вводится выражение $[a, b]$ (для ввода числового промежутка используются клавиши "Ctr-квадратная скобка"). Цели получаемой задачи - "упростить", "рядфурье x a b", "одз".

10) "Получить явное описание класса". Это - целевая установка задач, в которых множество, изначально заданное с помощью описателя "класс", нужно переформулировать в явном виде, не использующем описателей "класс"и "отображение". К числу таких задач относятся, например, задачи на определение геометрического места точек. Возникающая здесь целевая установка состоит из целей "упростить", "класс", "одз".

11) "Получить асимптотическую оценку". Это - целевая установка задачи на преобразование, в которой требуется получить асимптотическую оценку исходного выражения для предельного поведения переменных, заданного посылками "стремится(...)". Установка состоит из целей "асимптоценка", "упростить", "одз".

Подраздел "Вычисление значения константного выражения" раздела "Преобразовать выражение" содержит следующие пункты:

1) "Найти точное целочисленное значение". В этом случае условие задачи на преобразование построено при помощи целочисленных операций (сложение, вычитание, умножение, степень с натуральным показателем, факториал, модуль и т.п.) из целочисленных констант, и требуется найти его точное численное значение. Число десятичных знаков ограничивается лишь возможностями интерпретатора (свыше 600) и ограничителями в приемах, которые легко ослабить либо вовсе убрать. Задача имеет цели "упростить"и "число".

2) "Вычислить с заданной точностью". Здесь определяется приближенное значение константного выражения, в котором гарантированы первые n знаков после запятой. После выбора данной целевой установки происходит возвращение в набор текста задачи, где прорисовывается строка "Вычислить с точностью до", и под ней - курсор формульного редактора. После набора формульным редактором числа n нажимается "Enter"; тогда n переносится в конец указанной выше строки, и после него добавляется "знаков после запятой:". Задача имеет цели "упростить", "числоценка n".

3) "Найти приближенное значение". Здесь определяется приближенное значение константного выражения без каких-либо гарантий относительно числа точных знаков. Для вычислений используется математический сопроцессор; числа представляются в формате с плавающей запятой и с двойной точностью. Задача имеет цели "числзначение"и "выч".

Подраздел "Найти значения неизвестных"содержит следующие пункты (все они относятся к задачам на описание):

1) "Получить полное явное описание значений неизвестных". Это - целевая установка для обычных задач "с неизвестными", например, для решения систем уравнений и неравенств из элементарной алгебры. Для геометрических задач на вычисление

и задач на решение дифференциальных уравнений используются другие целевые установки (см.ниже). В целевую установку входят цели "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $x_1 \dots x_n$". После выбора целевой установки на экране прорисовывается строка "Найти", под которой формульным редактором вводятся неизвестные задачи - переменные, отделенные друг от друга запятыми. Затем нажимается "Enter".

2) "Найти пример значений неизвестных". Это - задачи на нахождение конкретного примера значений неизвестных, при которых истинны условия. Они имеют цели "полный", "пример", "прямойответ", "одз", "упростить", "неизвестные $x_1 \dots x_n$". После выбора целевой установки на экране прорисовывается строка "Найти пример для". Далее формульным редактором перечисляются неизвестные и нажимается "Enter".

3) "Выразить значения неизвестных через заданные параметры". Это - задачи на вычисление, в которых требуется выразить значения неизвестных (уже явно выраженных через некоторые вспомогательные переменные посылок) через заданные известные величины. Такие задачи на описание - наиболее часто встречающиеся в элементарной и аналитической геометрии, в физике и т.п. Они имеют целевую установку "полный", "явное", "прямойответ", "одз". "упростить", "известно $a_1 \dots a_m$", "неизвестные $x_1 \dots x_n$". Здесь $x_1, \dots, x_n$ - неизвестные, $a_1, \dots, a_m$ - известные параметры. Возможен случай $m = 0$ - если искомые значения неизвестных суть константы; в этом случае соответствующая цель состоит из логического символа "известно". После выбора целевой установки на экране прорисовывается строка "Выразить", и под ней формульным редактором набираются неизвестные (через запятую). После этого неизвестные автоматически переносятся в конец указанной строки, к ней добавляется слово "через", и формульным редактором под строкой перечисляются известные параметры. После нажатия "Enter"(которое происходит сразу же в случае $m = 0$) известные параметры перерисовываются в конце строки. Если этих параметров не было, то вся строка перерисовывается виде "Найти значения $x_1, \dots, x_n$.

4) "Выразить значения неизвестных через известные параметры, предпочтительно в виде парамерического описания". Это - редко встречающаяся разновидность предыдущей целевой установки. Она возникла в аналитической геометрии, для задач на нахождение уравнений линий в параметрическом виде. Диалог по вводу установки такой же, как в предыдущем пункте. В конце диалога к строке целевой установки добавляются слова "Предпочтительно параметрическое описание". К списку целей (по сравнению с предыдущим пунктом) добавляется "вспомпараметр".

5) "Решить функциональные уравнения". Целевая установка задач на решение дифференциальных уравнений и других задач, в которых используется аналогичная контекстная семантика (по умолчанию навешивается квантор общности по значениям варьируемых переменных). Здесь варьируемые переменные $x_1, \dots, x_m$, от которых зависят неизвестные функции $y_1, \dots, y_n$, указываются в цели "связка $x_1 \dots x_m$". Остальные цели - "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $y_1 \dots y_n$". В процессе диалога прорисовывается строка "Найти", после которой формульным редактором должны быть набраны выражения $y_1(x_1, \dots x_m), \dots y_n(x_1, \dots, x_m)$.

6) "Найти значения неизвестных, для которых существуют заданные объекты". Это - обычные задачи "с неизвестными"(см. пункт 1), у которых для части неизвестных не требуется найти их явное значение, а достаточно установить существование такого значения. Такие "несущественные"неизвестные, как правило, в ответ задачи вообще не входят. Цели задачи суть "полный", "явное", "прямойответ", "одз", "упро-

стить", "неизвестные $x_1 \dots x_n$", "параметры $y_1 \dots y_m$". Здесь $y_1, \dots, y_m$ - те из неизвестных $x_1, \dots, x_n$, для которых не требуется искать явное значение. При вводе целевой установки используется следующий диалог. Сначала прорисовывается строка "Найти"; затем формульным редактором вводятся неизвестные $x_1, \dots, x_n$; далее к строке добавляются слова "для которых существуют", и формульным редактором вводятся несущественные неизвестные $y_1, \dots, y_m$.

7) "Вычислить значения неизвестных при заданных значениях параметров". Целевая установка используется для задач на описание, ответом которых служит программа вычислений, определяющая численные значения неизвестных по заданным численным значениям параметров. Такая программа создается в два этапа. Сначала решается обычная задача на описание, преобразующая исходную систему утверждений, связывающих значения параметров и неизвестных, в систему утверждений, определяющих алгоритм вычислений. Обычно здесь применяются стандартные вычислительные формулы (метод трапеций, метод Рунге-Кутта, метод Ньютона и т.п.). Могут использоваться возможности решателя, связанные с задачами по физике, разумеется, достаточно скромные для текущего этапа его обучения. Полученные утверждения передаются компилятору ГЕНОЛОГа, который создает по ним ЛОС-программу для вычислений. Если задача не имела параметров, то полученная программа сразу же запускается, и выдается найденный ответ. В противном случае появляется интерфейс для задания параметров. В верхней части экрана прорисовываются сначала параметры, под ними - неизвестные. Для ввода значений параметров нажимается "в", и далее - через запятую перечисляются равенства, фиксирующие численные значения параметров. Точка в конце не ставится; по окончании ввода нажимается Enter. Этот интерфейс можно использовать многократно, не производя повторного синтеза программы. Если задача уже была решена ранее, то можно сразу входить в интерфейс вычислений по ее программе, нажимая вместо "о" (кир.) клавишу "Ctr - o".

Целевая установка состоит в рассматриваемом случае из целей "полный", "явное", "прямойответ", "одз", "упростить", "вычисление", "неизвестные ...", "параметры ...".

На текущий момент проработано несколько типов вычислительных задач, что позволяет использовать решатель, например, для вычисления определенных интегралов и решения систем уравнений - обычных или дифференциальных. Однако, в целом работа по вычислительным задачам в логической системе находится на самом начальном этапе. Основной ее целью является такое развитие компилятора ГЕНОЛОГа, которое позволило бы создавать достаточно эффективные вычислительные программы непосредственно с уровня теорем, как это имеет место для логических приемов. Данная постановка проблемы представляется совершенно естественной, так как подлинным источником любых вычислительных процедур в математике служат теоретические знания (в обобщенном смысле - теоремы). Подключение к компиляции предварительных логических процедур, работающих с такими знаниями - в виде решения задач либо логического вывода теорем - существенно увеличит возможности системы по бытрому созданию вычислительных блоков и объединению их в более сложные вычислительные агрегаты. Следует выделить здесь также интересную перспективу освоения смешанных логико-вычислительных режимов при автономном исследовании решателем свойств различных математических моделей.

8) "Получить явное описание значений неизвестных, не обязательно полное". Установка применяется в задачах на описание, для которых требуется получить не полный, а лишь возможно более полный ответ. Она состоит из целей "явное",

"прямойответ", "одз", "упростить", "неизвестные ...". Была использована пока в единичных случаях, и лишь совсем немногие приемы ее учитывают. По существу, представляет собой заготовку для последующего развития системы.

9) "Получить полное явное описание значений неизвестных, используя приближенные вычисления". Как и в предыдущем случае, это заготовка для последующего развития системы. Пока использована лишь для указания на режим использования приближенных вычислений в машинном формате "с плавающей запятой" при численном решении систем линейных уравнений. Процедура, выполняющая последнее, совершенно стандартна, и представляет интерес лишь как пример создания компилятором ГЕНОЛОГа обычной вычислительной программы с того же "теоремного" уровня, с которого создаются логические приемы.

Подраздел "Исследовать свойства объекта"содержит следующие пункты:

1) "Исследовать поведение функции". Это - задачи на качественное исследование графика функции одной вещественной переменной (область определения, промежутки монотонности, экстремумы, число корней на промежутках). Они имеют цели "полный", "явное", "прямойответ", "одз", "исследовать", "функция", "неизвестные x", где x - исследуемая функция. Условие такой задачи сводится к равенству $x = \lambda_y(f(y), y-$число$)$, определяющему функцию. Диалог ввода целевой установки прост: сначала прорисовывается строка "Исследовать поведение функции", затем вводится переменная x.

2) "Исследовать поведение функции с учетом выпуклости и вогнутости". Этот пункт аналогичен предыдущему; при исследовании предпринимается дополнительно попытка найти промежутки выпуклости и вогнутости. К списку целей предыдущего пункта добавляется цель "выпуклавверх".

3) "Исследовать функцию на монотонность". Этот пункт аналогичен пункту 1, но предпринимается исследование только интервалов монотонности. К списку целей пункта 1 добавляется цель "монотонно".

4) "Исследовать функцию на четность и периодичность". Этот пункт аналогичен пункту 1, но предпринимается исследование только четности и периодичности (в случае 1 такое исследование не выполнялось). К списку целей пункта 1 добавляется цель "четнаяфункция".

5) "Исследовать функцию на непрерывность". Этот пункт аналогичен пункту 1, но исследуется только непрерывность: находятся интервалы непрерывности, точки разрыва, и определяются типы этих точек. К списку целей пункта 1 добавляется цель "непрерывно".

6) "Исследовать функцию на равномерную непрерывность". Этот пункт аналогичен пункту 1, но исследуется только равномерная непрерывность (без исследования точек разрыва). К списку целей пункта 1 добавляются цели "непрерывно"и "равномернонепрерывно".

7) "Исследовать свойства линии, заданной своим уравнением". Здесь выполняется определение типа кривой второго порядка, заданной своим уравнением, и предпринимается стандартная характеризация этой кривой (включая нахождение канонической системы координта и канонического уравнения). Цели задачи суть: "полный", "явное", "прямойответ", "одз", "исследовать", "линия", "неизвестные x", где x - исследуемая кривая. Условие задачи имеет вид "коорд$(x, K) = set_{xy}(x -$ число $\&\ y -$ число $\&\ f(x, y) = 0)$". В действительности элементы $x-$число,$y-$число этого условия можно не вводить - они автоматически добавляются решателем, если соответствующая переменная явно встречается в $f(x, y)$ под какой-либо арифметической операци-

ей. В посылках задачи должно иметься утверждение "прямкоорд(K)". Диалог создания установки аналогичен пункту 1: после прорисовки строки "Исследовать кривую"вводится неизвестная x, и далее к строке добавляются слова "заданную своим уравнением".

8) "Исследовать свойства поверхности, заданной своим уравнением". Аналогично предыдущему пункту, но для поверхности второго порядка. Вместо цели "линия" берется цель "эллипсоид".

9) "Определить вид множества точек, заданного условием на координаты точек". Целевая установка на преобразование описания множества точек к бескоординатной форме. Состоит из целей "полный", "явное", "прямойответ", "одз", "упростить", "исследовать", "точки", "неизвестные ...". Была использована в задачах на качественную характеризацию вида подмножества точек комплексной плоскости, удовлетворяющих заданным соотношениям.

10) "Исследовать функцию комплексного переменного на особые точки". Целевая установка на нахождение и характеризацию особых точек заданной функции комплексного переменного. Состоит из целей "полный", "явное", "прямойответ", "одз", "исследовать", "особыеточки", "функция", "неизвестные ...". Как данная, так и предыдущая установка связаны с разделами, для которых обучение решателя лишь начато. Их могут использовать те, кто захотел бы научиться создавать свои приемы для новых задач - по аналогии с теми, которые уже имеются в решателе и обеспечивают решение аналогичных задач его задачника.

Кроме режима ввода целевой установки с помощью оглавления типов целей, можно набирать эту установку непосредственно текстовым редактором. Для входа в режи набора установки текстовым редактором служит клавиша "Ctr-ц". Сначала набирается тип задачи ("преобразовать","описать") и далее перечисляются в скобочной записи цели. Между целями оставляются пробелы; запятые не используются. Если цель имела вид набора ($fa_1 \dots a_n$), то она вводится как $f(a_1 \dots a_n)$.

Чтобы изменить ранее введенную целевую установку, используются те же клавиши "ц"и "Ctr-ц", что и для ее изначального создания. При этом следует помнить, что если задача, для которой изменяется целевая установка, не является последней в текущем списке задач, то предварительно ее следует выделить.

Можно просматривать оглавление типов целевых установок независимо от процесса ввода либо изменения задачи. Для входа в это оглавления достаточно нажать (находясь в просмотре списка задач) клавишу "Ц". Обычно данная возможность применяется для редактирования оглавления типов целевых установок.

После ввода целевой установки задачи выполняется ввод ее условия либо (в случае задачи на описание) нескольких условий. На этом процесс создания новой задачи завершается.

Если нужно сохранить текстовую версию формулировки задачи, нажимается клавиша "н", и далее эта версия вводится с помощью текст-формульного редактора. Подробнее об интерфейсе текст-формульного редактора см. в главе 6.

3.4.4 Формульный редактор

Приведем краткое описание формульного редактора, с помощью которого происходит ввод условий и посылок задач. При входе в формульный редактор появляется прямоугольник курсора (коричневатого цвета). Этот курсор, в отличие от курсора текстового редактора, нельзя перемещать клавишами перемещений курсора -

его положение зафиксировано и соответствует текущему вводимому символу формулы. Ситуация при использовании формульного редактора "линейная можно либо ввести очередной символ, либо отменить последний введенный символ нажатием "Backspace". В процессе ввода формулы происходят автоматические масштабирование, "центровка"и перенесение на новую строку элементов изображения. Для завершения ввода формулы нажимается "Enter". Эта же клавиша служит для указания на завершение набора отдельных фрагментов формулы - дробей, радикалов, интегралов и т.п. Применение ее в таких случаях должно быть аккуратным, так как повторное нажатие приведет к преждевременному обрыву набора формулы (впрочем, для возвращения в редактирование терма задачи достаточно выделить этот терм и нажать "ф"). Чтобы отменить начатое редактирование формулы и выйти из формульного редактора, нажимается "Esc". Заметим, что если формула заведомо не набрана до конца либо набрана с принципиальными ошибками, то формульный редактор игнорирует завершающие нажатия клавиши "Enter". В этом случае следует либо довести набор формулы до конца, либо вернуться (Backspace) к ошибке и исправить ее, либо вообще повторно набрать всю формулу целиком.

При входе в формульный редактор автоматически осуществляется переход в латинский режим ввода символов с клавиатуры; при выходе - автоматическое возвращение в "кириллицу". Несмотря на "латинский"режим, приводимые далее коды клавиш и сочетаний клавиш, используемые в формульном редакторе, часто даются через "кириллические"обозначения клавиш (никакого переключения режима клавиатуры при их использовании не предполагается - они приводятся в таком виде просто для удобства запоминания). В тех случаях, когда возможна путаница, явно указывается на использование кириллицы (в скобках ставится пометка "рус.").

Заметим, что в любых режимах логической системы для ручного перехода в латинский режим клавиатуры достаточно нажать F12, а для перехода в режим кириллицы - F11. Однако, использовать эти возможности рекомендуется только при работе в текстовом редакторе. Следует помнить, что все управляющие клавиши и сочетания клавиш в логической системе ориентированы на кириллический режим ввода, так что переход к латинскому режиму будет означать просто отключение соответствующих управляющих воздействий.

Переменные в формульном редакторе набираются либо без индексов (малые и большие латинские буквы), либо с числовыми индексами. Для набора индекса после ввода переменной нажимается клавиша "курсор вниз", и далее вводится индекс. По окончании ввода индекса нажимается "Enter".

Некоторые логические символы обозначаются в "стандартной"математической записи теми же словами и словосочетаниями, что и во внутренней скобочной записи. Это относится в первую очередь к понятиям, для которых в математике не предусмотрено специальной символики. Разумеется, по мере обучения решателя доля таких понятий постоянно увеличивается. В особых случаях, для наиболее часто встречающихся понятий, вводятся специальные клавиши либо комбинации клавиш (обычно 2 последовательно нажимаемые клавиши). В остальных случаях для ввода логического символа, изображаемого словом либо словосочетанием, нажимается клавиша "Ctr-Enter", переводящая в режим побуквенного набора данного символа текстовым редактором. По окончании набора нажимается клавиша "Enter", возвращающая в режим формульного редактора.

Чтобы в процессе набора формулы можно было получить справку относительно редко встречающихся понятий, набираемых указанным способом, можно пользоваться оглавлением логического языка системы. В нем перечислены способы построения

термов в скобочной записи для всевозможных используемых системой понятий. Переход в оглавление из набора формулы обеспечивается нажатием клавиши F2. Если найден необходимый концевой пункт и на нем нажата клавиша "курсор вправо", то происходит автоматическое возвращение в редактирование формулы. При этом название выбранного символа уже прорисовано, а под ним размещен текст пояснений. После того, как пояснения к символу становятся не нужны, они убираются нажатием Ctr-F2.

Переход на новую строку выполняется формульным редактором автоматически. Однако, при использовании указанного выше способа ввода логического символа через "Ctr-Enter" может оказаться, что для набора этого символа текстовым редактором на текущей строке места недостаточно. Тогда перед набором следует обеспечить переход к новой строке вручную, нажав клавишу "Ctr-курсор вниз".

Подробное перечисление поддерживаемых формульным редактором логических символов и способов их ввода приведено в справочнике по логической системе. Этот справочник доступен, например, из главного меню (по F1); в процессе набора формулы также можно перейти к нему, нажав клавишу F1. После нажатия возникает оглавление справочника. В этом оглавлении следует найти корневой раздел, и в нем - выбрать пункт "Формульный редактор". Для возвращения в набор формулы из просмотра справочника нажимается "End"(из оглавления справочника - однократно, из концевого текста справочника - "End"нажимается дважды). Здесь мы приводим достаточно полное перечисление возможностей формульного редактора, опуская только те, которые используются крайне редко.

1. Общелогические символы. Отрицание $\neg A$ утверждения A вводится путем последовательного нажатия клавиш n,o (лат.) и последующего набора "A". Символы "и","или"вводятся, соответственно, как & и Ctr-v. Символ "если-то", встречающийся только под квантором общности, изображается стрелкой и вводится нажатием клавиши "курсор вправо". Символ "эквивалентно"обозначается двусторонней стрелкой и вводится нажатием клавиши "курсор влево". Для ввода кванторной записи "$\forall x_1 \dots x_k(A_1\& \dots \& A_n \to A_0)$"сначала дважды нажимается большая латинская буква A и появляется знак квантора общности. Затем перечисляются (без каких-либо пробелов) переменные $x_1, \dots, x_k$ кванторной приставки; нажимается "Enter"(здесь появляется открывающая скобка), после чего вводится конъюнкция $A_1\& \dots \& A_n$. Далее нажимается "курсор вправо"(появляется стрелка для "если-то"), вводится утверждение A_0, и в конце ставится закрывающая скобка. Для ввода кванторной записи "$\exists x_1 \dots x_k(A)$"сначала дважды нажимается большая латинская буква E и появляется знак квантора существования. Затем набирается кванторная приставка $x_1 \dots x_k$; нажимается "Enter"(появляется открывающая скобка); вводится утверждение A, и ставится закрывающая скобка. Возможно использование квантора "существует и единственно". После знака квантора существования здесь идет восклицательный знак. Для появления такого обозначения следует вместо двукратного нажатия "E"нажать последовательно клавиши "E"и "T"(лат.). Равенство вводится обычным образом. Логические константы "истина"и "ложь"прорисовываются как "truth"и "false". Они вводятся, соответственно, двукратным нажатием клавиши t либо f. Условное выражение (A при P, иначе B), принимающее значение A, если истинно условие P, и значение B в противном случае, набирается следующим образом. Ставится открывающая скобка; вводится выражение A; нажимается Ctr-e (е-кир., от слова "если"); вводится P; нажимается Ctr-и (и - от слова "иначе"); вводится B, и ставится закрывающая скобка.

2. Арифметические операции и отношения. Числовые константы набираются обы-

чным образом; в случае десятичных дробей используется точка. Операции "плюс" (произвольное число слагаемых, большее или равное 2) и "минус"вводятся клавишами "+"и "−". Умножение вводится нажатием клавиши "звездочка", причем в случае однократного нажатия этой клавиши (перед очередным сомножителем) фактической прорисовки чего-либо на экране не происходит. Если нужно убедиться в том, что нажатие клавиши и ввод операции умножения состоялись, то клавиша "звездочка"должна быть нажата еще дважды - в этом случае для умножения будет прорисована точка. Следует особенно аккуратно вводить произведения, часть сомножителей которых заключена в скобки. Если пропустить нажатие клавиши "звездочка"после A, то произведение $A(B+C)$ будет воспринято как значение функции A в точке $B+C$. Дробное выражение $\frac{A}{B}$ вводится следующим образом. Сначала нажимается клавиша "двоеточие" - после этого прорисовывается красным цветом горизонтальная черта дроби, над которой размещается курсор. Затем вводится числитель A. В процессе ввода горизонтальная черта дроби удлиняется автоматически; если вводится многоэтажное выражение, то для обеспечения необходимого места происходит автоматическая коррекция размеров и размещения всей ранее введенной части формулы. После ввода числителя нажимается клавиша "Enter" - тогда курсор перемещается в начало знаменателя, и предпринимается ввод знаменателя B. Наконец, после ввода знаменателя нажимается клавиша "Enter", завершающая ввод дроби. При этом горизонтальная черта дроби из красной становится черной, и происходит автоматическая центровка числителя и знаменателя для получения симметричной записи. Степенное выражение A^B вводится следующим образом. Сначала набирается выражение A (если оно само является результатом применения более чем одноместной операции, то обязательно заключается в скобки). Затем нажимается клавиша "курсор вверх" - курсор поднимается вверх для прорисовки показателя степени. После ввода выражения B (его не обязательно заключать в скобки) нажимается "Enter" - курсор опускается обратно, и ввод степени считается законченным.

Отношения "меньше","больше"вводятся с помощью клавиш $<$, $>$; отношения "меньшеилиравно","большеилиравно" - с помощью клавиш "левая квадратная скобка","правая квадратная скобка". Для ввода утверждения "A - число"сначала набирается выражение A, затем нажимаются клавиши "/"и "ч".

Конечная сумма

$$\sum_{i=a}^{b} f(i)$$

вводится следующим образом. Сначала нажимается "Ctr-s" - появляется большая буква "сигма"малинового цвета, причем курсор находится под этой буквой. Здесь набирается $i = a$ и нажимается "Enter". Тогда курсор переводится в положение над буквой "сигма", где набирается выражение b и снова нажимается "Enter". После этого курсор оказывается справа от буквы "сигма", где набирается выражение $f(i)$. По окончании набора нажимается "Enter", перекрашивающее букву "сигма"в черный цвет - набор суммы на этом закончен. Если требуется задать множество допустимых значений индекса суммирования косвенным образом, то используется запись

$$\sum_{i,P(i)} f(i).$$

Ввод ее происходит аналогично предыдущему, но в положении курсора над "сигмой" сразу нажимается "Enter".

Конечные произведения

$$\prod_{i=a}^{b} f(i), \qquad \prod_{i,P(i)} f(i)$$

вводятся совершенно так же, как конечные суммы, но вместо клавиши "Ctr-s"нажимается клавиша "Ctr-p"(здесь p - латинское).

3. Элементарные функции. Для ввода квадратного корня $\sqrt{A}$ последовательно нажимаются клавиши "s","r"(появляется радикал малинового цвета), вводится выражение A и нажимается "Enter" - радикал перекрашивается в черный цвет. Если нужно ввести корень $\sqrt[n]{A}$, где n - целое от 3 до 9, то последовательно нажимаются клавиши "r","t"(появляется малиновый радикал, над которым расположен курсор), "n"(курсор переводится под радикал), и далее - как для квадратного корня. Для ввода максимума либо минимума выражений $A_1, \ldots, A_n$ последовательно нажимаются, соответственно, латинские клавиши "m","a"либо "m","i", и далее в скобках перечисляются указанные выражения. Для ввода модуля выражения A сначала нажимается "Ctr-m", затем вводится A, затем снова нажимается "Ctr-m".

Выражение $\log_a b$ набирается следующим образом. Сначала последовательно нажимаются клавиши "l","o"(лат.). Затем набирается основание логарифма a. Далее нажимается "Enter" - курсор переводится в позицию справа от логарифма, и набирается выражение под логарифмом b. Если оно представляет собой многоместную операцию, то его обязательно следует заключить в скобки - иначе под логарифмом окажется лишь первый операнд этой операции. Это же замечание о скобках относится и ко всем приводимым ниже элементарным функциям. В случае натурального логарифма достаточно нажать клавиши "l","n"и ввести выражение под логарифмом.

Экспонента $\exp(A)$ вводится последовательным нажатием клавиш "e","x"(лат.). Синус $\sin(A)$ вводится последовательным нажатием клавиш "s","i". Заметим, что степень синуса вводится не совсем стандартным образом: необходимо сначала набрать все выражение $\sin(A)$ (для большей наглядности, хотя и необязательно, заключенное в скобки), и лишь затем нажать "курсор вверх", переходя к вводу показателя степени. Помещать показатель степени сразу же после sin нельзя. Это же замечание относится и к другим элементарным функциям.

Оставшиеся элементарные функции вводятся однотипным с синусом образом - последовательным нажатием двух либо трех латинских клавиш (в ряде случаев, как указано ниже, требуется вводить не малую, а большую букву). Мы просто перечислим для них эти пары либо тройки клавиш:

косинус - "c","o"; тангенс - "t","g"; котангенс - "c","t"; секанс - "s","e"; косеканс - "c","s"; арксинус - "a","s"; арккосинус - "a","c","o"; арктангенс - "a","t"; арккотангенс - "a","c","t"; сигнум (минус единица для отрицательных, не определен в нуле, единица для положительных) - "s","g"; сигнум (0 для отрицательных, 1 для неотрицательных) - "S","g"; гиперболический синус - "s","h"; гиперболический косинус - "c","h"; гиперболический тангенс - "t","h"; гиперболический котангенс - "c","T".

4. Символьные константы. Константа "e" вводится двойным нажатием латинской клавиши "e" (важно выполнять эту операцию аккуратно, чтобы не получить вместо константы "e" переменную "e"; по внешнему виду они несколько отличаются друг от друга). Константа "пи"вводится двойным нажатием латинской клавиши "p". Константа "плюс-бесконечность"водится двойным нажатием клавиши "i". Для получения минус-бесконечности перед плюс-бесконечностью помещается знак "минус"(хотя во внутреннем представлении минус-бесконечность представлена отдельным логическим символом). Мнимая единица вводится двукратным нажатием клавиши "c".

5. Операции и отношения для целых чисел. Утверждения "A - целое", "A - натуральное", "A - рациональное", "A - четное"вводятся следующим образом: сначала набирается выражение A, затем нажимается "/" , и далее, соответственно, клавиша "ц"(целое) либо "н"(натуральное), либо "q"(рациональное), либо "е"(четное;лат.). Выражения "числитель(A)"и "знаменатель(A)"вводятся последовательным нажатием клавиш "ч","и"либо, соответственно, "з","н"(далее набирается A, при необходимости - в скобках).

Утверждение "$A|B$"(A делит B) вводится в следующей последовательности: сначала набирается A, затем нажимается "Ctr-д", затем B. Выражения "нод(A,B)"и "нок(A,B)"(наибольший общий делитель и наименьшее общее кратное) вводятся, соответственно, последовательным нажатием клавиш "н","д"либо "н","к"(кир.) и дальнейшим набором (в скобках и через запятую выражений A,B. Аналогично вводятся утверждения "простое(A)" (двойное нажатие "п") и "взаимнопросты(A,B)" (клавиши "в","п").

Целая часть $[A]$ выражения A набирается следующим образом: сначала нажимается клавиша "Ctr-е"(е - лат.), что приводит к появлению левой квадратной скобки. Затем вводится A и снова нажимается "Ctr-е для правой квадратной скобки.

Для вычета $A(modB)$ сначала набирается выражение A, затем открывающая скобка, затем нажимаются клавиши "м","д"(кир.), затем набирается B, и в конце - закрывающая скобка.

Число сочетаний из n по k вводится следующим образом. Сначала нажимается "Ctr-с", что приводит к появлению большой буквы "С"малинового цвета; курсор находится справа в нижней части этой буквы. Затем набирается n и нажимается "Enter" - курсор перемещается в верхнюю часть справа от буквы "С". Далее набирается k и снова нажимается "Enter" - буква перекрашивается в черный цвет и ввод числа сочетаний завершается. При необходимости в процессе ввода вертикальные размеры буквы "С"автоматически увеличиваются.

Факториал $n!$ вводится последовательным набором n (при необходимости - в скобках) и нажатием "Ctr-f".

6. Многочлены. Многочлен P над полем вещественных чисел (см. пояснение про многочлены из раздела "Логический язык решателя") вводится следующим образом. Сначала дважды нажимается клавиша "m"(лат.) - появляется греческая буква "мю", справа от которой и чуть ниже располагается курсор. Далее вводятся все переменные, от которых формально зависит многочлен, и нажимается "Enter" - курсор поднимается вверх до уровня буквы "мю". После этого набирается сумма одночленов, определяющая значения многочлена, и в конце ставится закрывающая скобка.

7. Алгебра множеств. Утверждение "A - множество"вводится последовательным набором A, нажатием "/"и "s". Символ $\emptyset$ пустого множества вводится последовательным нажатием клавиш "e","m"(лат.). Символ $\cup$ объединения множеств вводится последовательным нажатием клавиш "пробел", "u", "s"; символ $\cap$ пересечения множеств - нажатием "пробел", "i", "s". Для ввода символа $\setminus$ разности множеств нажимается клавиша "\". Символ $\in$ принадлежности множеству вводится последовательным нажатием клавиш "пробел", "b", "e"(лат.); символ $\subset$ включения множеств - "пробел", "s", "u". Символ $\times$ прямого произведения множеств вводится нажатием клавиш "пробел", "p", "s"(лат.).

Конечное множество $a_1, \ldots, a_n$, заданное перечислением своих элементов, вводится путем последовательного набора этих элементов (через запятую) внутри фигурных скобок.

Конечное объединение

$$\bigcup_{i=a}^{b} f(i)$$

вводится следующим образом. Сначала нажимается "Ctr-u" - появляется большой знак "объединение"малинового цвета, причем курсор находится под ним. Здесь набирается $i = a$ и нажимается "Enter". Тогда курсор переводится в положение над "объединением", где набирается выражение b и снова нажимается "Enter". После этого курсор оказывается справа от "объединения", где набирается выражение $f(i)$. По окончании набора нажимается "Enter", перекрашивающее знак "объединение"в черный цвет - его набор закончен. Если требуется задать множество допустимых значений индекса объединения косвенным образом, то используется запись

$$\bigcup_{i,P(i)} f(i).$$

Ввод ее происходит аналогично предыдущему, но в положении курсора над "объединением"сразу нажимается "Enter".

Конечные пересечения

$$\bigcap_{i=a}^{b} f(i), \qquad \bigcap_{i,P(i)} f(i)$$

вводятся совершенно так же, как конечные объединения, но вместо клавиши "Ctr-u"нажимается клавиша "Ctr-x"(здесь x - латинское).

Класс $set_x P(x)$ всех объектов (если x - список переменных, то наборов) x, удовлетворяющих условию $P(x)$, вводится следующим образом. Сначала дважды нажимается клавиша "S" - появляется "*set*". Затем вводится переменная либо список переменных x; нажимается "Enter"и набирается условие $P(x)$.

Мощность $card(A)$ множества A вводится нажатием клавиш "с","а"(лат.) и набором выражения A. Заголовки утверждений "конечное(A)", "непересек(A, B)", "семействомножеств(A)", "разбиение(A, B)"вводятся, соответственно, последовательными нажатиями пар клавиш (везде кириллица): "к","о"; "н","п"; "С","м"; "р","а". Заголовок выражения "подмножества(A)"вводится последовательным нажатием клавиш "П","м".

Мощности счетного множества и континуума вводятся, соответственно, нажатиями клавиш "с","ч"и "к","м". Символ пустого слова вводится нажатием клавиши "Ctr-щ".

Утверждения "убывмножества(A)", "возрастмножества(A)"о том, что последовательность множеств A является убывающей либо возрастающей по включению, вводятся последовательными нажатиями клавиш "У","М"и "В","М"(кир.).

8. Числовые множества. Для ввода промежутка числовой оси с концами A, B (они могут являться символами бесконечности) сначала нажимается "Ctr-("(если конец A не относится к промежутку)либо "Ctr-["(если этот конец относится к промежутку). Затем набираются отделенные запятой выражения A, B. Если конец B не относится к промежутку, то далее нажимается "Ctr-)", иначе - "Ctr-]".

Все множество вещественных чисел вводится двойным нажатием клавиши "R". Множество рациональных чисел вводится двойным нажатием клавиши "Q"; множество целых чисел - двойным нажатием "Z"; множество натуральных чисел - двойным нажатием "N"; множество целых неотрицательных чисел - последовательным нажатием "Ц","Н"(кир.).

Конечный отрезок $\{A, \ldots, B\}$ натурального ряда, начинающийся с A и кончающийся B, набирается следующим образом: левая фигурная скобка, выражение A, запятая, "Ctr-точка", запятая, выражение B, правая фигурная скобка

Точная нижняя грань $\inf A$ множества A вводится с помощью нажатия клавиш "I","n"; точная верхняя грань $\sup A$ - с помощью "S","u".

Утверждения "нижняягрань(x, A)" (x - нестрогая нижняя грань числового множества A), "Нижняягрань(x, A)" (строгая нижняя грань), "верхняягрань(x, A)" (нестрогая верхняя грань), "Верхняягрань(x, A)" (строгая верхняя грань), "наибольший(x, A)" (x - наибольший элемент числового множества A), "наименьший(x, A)"вводятся, соответственно, с помощью последовательных нажатий пар клавиш (везде кириллица): "н","г"; "Н","г"; "в","г"; "В","г"; "Н","О"; "Н","Е". Утверждения "огрснизу (A)", "огрсверху(A)"об ограниченности числового множества A снизу либо сверху вводятся с помощью нажатий пар клавиш "г","н"и "г","в".

Выражения "внутренность(A)", "граница(A)", "замыкание(A)"вводятся, соответственно, при помощи нажатий пар клавиш "в","н"; "Г","р"; "З","З"(везде кир.).

9. Простейшие обозначения, связанные с функциями. Утверждение "A - функция"вводится последовательным набором A, нажатием "/"и "ф". Область определения $Dom(f)$ функции f вводится с помощью последовательного нажатия клавиш "d","o"(лат.). Множество значений $Val(f)$ функции f вводится с помощью последовательного нажатия клавиш "v","a". Для ввода значения $f(a)$ функции f в точке a сначала набирается обозначение функции f (если оно не односимвольное, то заключается в скобки), затем левая скобка, a и правая скобка. Выражения "образ(f, A)" (образ множества A), "прообраз(f, A)" (прообраз множества A) и "слой(f, a)" (полный прообраз элемента a) набираются при помощи последовательных нажатий пар клавиш "о","м"; "п","м"; "с","л"(везде - кириллица). Утверждение "взаимноодно-значно(f)" (f инъективно) вводится при помощи последовательного нажатия клавиш "в","о" (кир.). Множество корней числовой функции f на множестве A обозначается $roots(f, A)$ и вводится при помощи последовательного нажатия клавиш "r","o"(лат.).

Функция, значения которой определяются выражением t, а условие на принадлежность аргумента x (этот аргумент может быть как единственной переменной, так и набором переменных) области определения есть $P(x)$, обозначается $\lambda_x(t, P(x))$. Для ввода ее сначала дважды нажимается клавиша "L" - прорисовывается "лямбда", причем курсор располагается справа от нее и чуть ниже. Затем вводится x (переменная либо группа переменных, без запятых), нажимается "Enter"(курсор перемещается вверх на уровень "лямбды") и набираются в скобках $t, P(x)$.

Утверждение "Отображение(f, A, B)"(f есть отображение A в B) вводится при помощи двукратного нажатия клавиши "о"(кир.).

Функция "конст(A, b)", определенная на множестве A и принимающая во всех его точках значение b, вводится при помощи двукратного нажатия "к"(кир.). Функция "$a \to b$", определенная на одноэлементном множестве, состоящем из элемента a, и принимающая на этом элементе значение b, вводится следующим образом: сначала набирается a, затем "Ctr-курсор вправо", затем b. Тождественная функция "тождфунк(A)" с областью определения A вводится при помощи нажатия клавиш "т","ф".

Посредством "таблица$(A_1, \ldots, A_n)$"обозначается функция, график которой есть объединение графиков функций $A_1, \ldots, A_n$, принимающих на пересечениях своих областей определения одинаковые значения. Символ "таблица"вводится последовательным нажатием клавиш "т","а"(кир.).

Выражения "сужение(f, A)" (сужение функции f на множество A) и "доопределение(f, A, b)" (доопределение функции f на тех точках множества A, где она еще не

определена, значением b) вводятся при помощи последовательных нажатий клавиш "с","у"и "д","о"(кир.). Выражение "обрфункция(f)"(функция, обратная к взаимнооднозначной функции f) вводится при помощи последовательного нажатия "о","ф".

Выражение "числопеременных(f)"вводится при помощи клавиш "ч","п"; выражение "подфункиця(f, A, B)"(функция, полученная из f подстановкой элементов набора B вместо переменных, номера которых образуют набор A; нумерация начинается с 1) вводится при помощи клавиш "п","Ф". Утверждение "существперем(i, f)" (i-я переменная функции f является существенной) вводится при помощи клавиш "С","У".

10. Упорядоченные наборы. Всюду далее понимаем под "набором"только упорядоченный набор (вектор); в справочнике по решателю используется такое же соглашение. Фактически "набор" здесь - это отображение начального отрезка натурального ряда на некоторое множество. Набор элементов $a_1, \ldots, a_n$ вводится простым перечислением через запятую этих элементов. Этот набор при вводе можно заключать в скобки, а можно и не заключать. Однако, для ввода одноэлементного набора ($n = 1$) необходимо использовать другой способ - сначала нажать "Ctr-н"(кир.) и затем ввести a_1; если a_1 неоднобуквенное, то оно должно быть заключено в скобки.

При вводе выражений "префикс(A, B)"(добавление элемента A в начале набора B) и "суффикс(A, B)"(добавление элемента A в конце набора B) логические символы "префикс","суффикс"набираются текстовым редактором (вход в него - через "Ctr-Enter"). Исключение здесь составляет случай выражений "перечень(префикс($A\,B$))", которые прорисовываются в виде $\{A; B\}$ и набираются с помощью клавиши ";". Инфиксная операция конкатенация (последовательное соединение наборов) также вводится при помощи фигурных скобок и клавиши ";". Соответственно, запись $\{A, B, C; D\}$ обозначает конечное множество, перечисляемое конкатенацией наборов (A, B, C) и D.

Всюду далее предполагаем, что нумерация элементов набора начинается с 1. Выражения "поднабор(A, i, j)"(поднабор набора A, образованный элементами начиная с i-го и кончая j - м) и "Вставка(A, i, b)" (результат вставки в набор A перед i-м элементом элемента b) вводятся при помощи последовательных нажатий клавиш "п","н"и "В", "с"(кир.). Выражение "выборка(A, B)"обозначает результат исключения всех разрядов набора A, номера которых не принадлежат множеству B. Логический символ "выборка"вводится текстовым редактором. Выражение "наложение(A, B, N)"обозначает такой результат перемешивания элементов наборов A,B (с сохранением относительного порядка элементов каждого из них), в котором элементы набора A расположены на местах, номера которых образуют набор N. Логический символ "наложение"вводится текстовым редактором.

Выражение "наборномеров(i, j)"(набор, образованный целыми числами начиная с i и кончая j) вводится при помощи последовательного нажатия клавиш "н","о"(кир.).

Выражения "упорядвозр(A, f)" (результат упорядочения элементов набора по возрастанию значений числовой функции f на этих элементах) и "упорядубыв(A, f)" вводятся при помощи последовательных нажатий клавиш "у","в"; "у","ы".

11. Математический анализ. Предел $\lim_{x\to a} f(x)$ вводится следующим образом. Сначала последовательно нажимаются клавиши "l", "i" - рисуется знак предела, а курсор оказывается справа от него и чуть ниже. Затем вводится переменная x, нажимается клавиша "курсор вправо"и набирается выражение a (в случае левого либо правого одностороннего предела набирается $a - 0$ либо $a + 0$; если нужно рассмотреть общую ситуацию, не уточняя тип предела, то вместо a набирается $a\backslash b$, где b - указатель типа предела: 0 - двусторонний, 1 - левый, 2 - правый). Далее нажимает-

ся "Enter" - курсор возвращается на уровень "предела", и вводится выражение $f(x)$. Если это выражение представляет собой сумму, произведение и т.п., то его обязательно следует заключить в скобки - иначе под знаком предела окажется лишь первый операнд.

Предел последовательности обозначается путем заключения выражения $f(x)$ в фигурные скобки: $\lim_{n\to\infty}\{f(n)\}$.

Верхний предел $\overline{\lim}$ и нижний предел $\underline{\lim}$ вводятся, соответственно, при помощи последовательного нажатия клавиш "L","i"и "l","I".

Используемые в асимтотических оценках обозначения $O(x)$,$o(x)$ вводятся, соответственно, при помощи двукратного нажатия клавиши "O"либо "o"(лат.).

Производная $\frac{df(x)}{dx}$ вводится как обычная дробь: сначала нажимается клавиша ":" ,затем "d", "звездочка", выражение $f(x)$, "Enter", "d", "звездочка", "x"и "Enter". Если требуется задать значение производной в точке a, то вместо dx в знаменателе помещается $d(x = a)$ (после d - обязательно нажимается звездочка). Заметим, что если a - снова переменная, то автоматически произойдет изменение обозначений, так что при повторной прорисовке вместо $\frac{df(x)}{d(x=a)}$ будет изображено $\frac{df(a)}{da}$. При рассмотрении производных высших порядков и частных производных используются обычные записи типа $\frac{d^3f(x)}{dx^3}$,$\frac{d^4f(x,y,z)}{dx^2dydz}$. Они вводятся аналогично первой производной; символ дифференцирования d рассматривается при этом как обычный множитель.

Утверждения "Производная(f, g)" (значения функции g в области ее определения суть значения производной функции f) и "Частнпроизв(f, i, g)" (значения функции g в области ее определения суть значения частной производной функции f по i-му аргументу) вводятся путем последовательных нажатий клавиш "П","р"и "Ч","п". Выражение "дифференциал(f, n, x, a)" (значение полного дифференциала порядка n функции f в точке x при наборе a значений приращений переменных) вводится путем последовательного нажатия клавиш "Д","и".

Неопределенный интеграл

$$\int f(x)dx$$

вводится следующим образом. Сначала нажимается "Ctr-j"(прорисовывается малиновым цветом знак интеграла), затем вводится подынтегральное выражение $f(x)$, нажимается "звездочка", "d", вводится переменная интегрирования x и нажимается "Enter" - знак интеграла из малинового становится черным.

Определенный интеграл

$$\int_a^b f(x)dx$$

вводится при помощи "Ctr-i". После этого курсор оказывается снизу от знака интеграла. Здесь набирается выражение a, затем нажимается "Enter" - курсор перемещается вверх от интеграла; набирается выражение b и снова нажимается "Enter". Далее - все как при наборе неопределенного интеграла.

Двойной интеграл

$$\int\int_P f(x, y)dxdy$$

вводится при помощи нажатия клавиши "Ctr-2". Курсор оказывается непосредственно под знаком двойного интеграла, где набирается выражение для области интегрирования P. Обычно здесь просто помещается вспомогательная переменная, а в посылках указывается равенство $P = M$, явно задающее область - иначе изображение

интеграла становится чрезмерно громоздким. После ввода P нажимается "Enter" - курсор перемещается вправо от знака интеграла. Здесь набирается подынтегральное выражение $f(x,y)$, "умноженное"на $dxdy$. В конце нажимается "Enter", перекрашивающее знак интеграла в черный цвет. Аналогичным образом вводится тройной интеграл - здесь используется клавиша "Ctr-3".

Ряд с общим членом $f(i)$ вводится как последовательность частичных сумм - $\lambda_n(\sum_{i=1}^{n} f(i), n -$ натуральное). Утверждение "сходится(f)"о сходимости последовательности (в частности, ряда; f - функция натурального аргумента, обычно вводимая через λ_n) вводится при помощи последовательного нажатия клавиш "с","д"(кир.). Сумма ряда с общим членом $f(i)$ обозначается обычным образом -

$$\sum_{i=m}^{\infty} f(i).$$

Правила набора здесь те же, что и для конечных сумм.

Утверждения "переменазнака(f,a)"(функция f меняет знак в окрестности точки a), "убываетвточке(f,a)", "возрастаетвточке(f,a)", "огрвточке(f,a)" (функция f ограничена в окрестности точки a) вводятся, соответственно, при помощи последовательных нажатий клавиш "п","з"; "У","т"; "В","т"; "О","Т"(кир.).

Запись $x \to a\backslash b$, уже встречавшаяся в связи с пределами, может использоваться как независимая посылка, для указания на то, что задача должна решаться в сколь угодно малой окрестности точки a ($\backslash b$ указывает тип окрестности и может отсутствовать либо быть замененным на $+0,-0$).

Утверждение "ограничено(A)"используется в контексте некоторого списка посылок и означает, что в области истинности этих посылок выражение A ограничено по модулю некоторой константой. Логический символ "ограничено"вводится последовательным нажатием клавиш "О","Г". Обычно это утверждение не встречается в формулировке задачи, а вводится при необходимости решателем в процессе решения.

Утверждения $Max(f,A,B,c)$ (функция f достигает на множестве A максимума c в точках подмножества B), $Min(f,A,B,c)$ вводятся при помощи последовательных нажатий клавиш "M","a"; "M","i" (лат.). Утверждение $Extr(f,a,b,c)$ (функция f имеет в точке a экстремум; b - значение ее в этой точке, а c - тип экстремума) вводится нажатием клавиш "E","x"(лат.).

Выражения "стационарныеточки(f)"(точки из внутренности области определения функции, в которых все частные производные равны нулю) и "особыеточки(f)" (точки из внутренности области определения, в которых хотя бы одна частная производная не определена) вводятся при помощи клавиш "с","ц"; "о","ч".

Утверждения "Выпуклавверх(f,A)" (функция f выпукла вверх на множестве A) и "Выпуклавниз(f,A)" вводятся при помощи нажатий клавиш "в","в"; "в","з"(кир.). Утверждения "убывает(f,A)"(функция f строго убывает на множестве A), "возрастает(f,A)", "неубывает(f,A)", "невозрастает(f,A)"вводятся при помощи последовательных нажатий клавиш "У","ы"; "В","о"; "Н","у"; "Н","в".

Утверждения "четнаяфункция(f)", "нечетнаяфункция(f)", "периодична(f,T)" (функция f периодична на числовой оси и имеет периодом, не обязательно наименьшим, число T) вводятся при помощи нажатий клавиш "ч","ф"; "н","ф"; "п","Е" (кир.).

Утверждения "непрерывно(f,A)"; "равномернонепрерывно(f,A)" вводятся при помощи "н","н"; "р","н"(кир.). Утверждения "устранимыйразрыв(f,a)" (функция f имеет в точке a устранимый разрыв), "разрывпервогорода(f,a)", "разрывторогорода(f,a)" вводятся с помощью текстового редактора.

Операции "умножфунк", "плюсфунк", "минусфунк"умножения, сложения и изменения знака числовых функций с общей областью определения вводятся формульным редактором так же, как обычные операции умножения, сложения и изменения знака чисел. Формульный редактор пытается усмотреть из контекста, что значениями операндов служат функции, и автоматически заменяет операции над числами на соответствующие операции над функциями. В тех случаях, когда усмотрение функционального типа операндов из контекста сомнительно, следует контролировать результат ввода, переходя к просмотру набранного терма в текстовом редакторе и осуществляя указанные коррекции вручную.

12. Планиметрия. В планиметрии фигуры и их числовые характеристики обозначаются только через соответствующие "опорные"точки ("прямая(AB)", "треугольник(ABC)"и т.п.); использование вспомогательных переменных для обозначения самих фигур ("прямая A"; "треугольник B"и т.п.) в приемах решателя не предусмотрено. В стереометрии, наоборот, тела обозначаются не с помощью их "опорных"точек, а с помощью вспомогательных переменных ("куб(A)", "пирамида(B)"и т.п.).

Утверждение "A-точка"вводится нажатием, после набора A, клавиш "\" и "р" (лат.). Обычно такие утверждения вообще не вводятся вручную, а создаются решателем самостоятельно в начале решения задачи.

Выражения "прямая(AB)", "отрезок(AB)", "интервал(AB)", "луч(AB)", "обратныйлуч(AB)" (луч с началом в точке A, противоположный лучу AB) вводятся последовательными нажатиями пар клавиш "п","р"; "о","т"; "и","н"; "л","у"; "о","л" (везде -кириллица). Заметим, что между A и B здесь не ставится запятая, в отличие от обычных многоместных операций и отношений формульного редактора. Такие исключительные случаи перечисления операндов без запятой используются только в планиметрии, причем лишь для некоторых (явно указываемых далее) логических символов. В качестве A, B могут быть использованы либо буквенные переменные, либо переменные с индексом. Если имеется хотя бы одна переменная с индексом, то между операндами необходим разделитель - клавиша "звездочка"(при ее нажатии на экране ничего не меняется, но становится возможен ввод следующего операнда).

Расстояние $l(AB)$ между точками A, B вводится при помощи двукратного нажатия клавиши "l". Величина $\angle ABC$ угла ABC вводится при помощи нажатия клавиш "у","г". В обоих случаях запятые между операндами не ставятся. Если нужно обозначить не величину угла, а множество точек плоскости, лежащих внутри него (включая граничные точки), то используется обозначение "Угол(ABC)", вводимое клавишами "У","г".

Выражения "расстдопрямой(A, B)"(расстояние от точки A до прямой B), "расстмежду(A, B)"(расстояние между двумя множествами точек A, B вводятся при помощи клавиш "Р","П"; "Р","Ы". Здесь уже нужна запятая между операндами.

Утверждение "биссектриса($ABCD$)"(луч BD есть биссектриса угла ABC) вводится двойным нажатием клавиши "и". Запятая между операндами не ставится.

Утверждения "$a \| b$"(прямые либо плоскости a, b параллельны) и "$a \perp b$"(прямые либо плоскости a, b перпендикулярны) вводятся, соответственно, при помощи клавиш "Ctr-="и "Ctr-т"(т - кир.).

Утверждения "однасторона(A, B, a)" (точки A, B лежат по одну сторону от прямой либо плоскости a, возможно, попадая на a), "разныестороны(A, B, a)" (здесь тоже допускается попадание точек на a), "симметричны(A, B, a)" (точки A, B расположены симметрично относительно точки, прямой либо плоскости a) вводятся при помощи клавиш "О","С"; "Р","С"; "с","и"(кир.). Выражения "полоса(A, B)"(полоса между параллельными прямыми A, B), "полуплоскость(A, B)" (полуплоскость, огра-

ниченная прямой A и содержащая точку B, не лежащую на B), "обрполуплоскость (A, B)" (полуплоскость, ограниченная прямой A и не содержащая точку B, не лежащую на прямой B) вводятся при помощи нажатий клавиш "п","с"; "п","П"; "о","П".

Для уточнения того, что вводимая задача относится к планиметрии - все рассматриваемые в ней точки лежат в общей плоскости - используется вспомогательная посылка "планиметрия". Обычно она вводится решателем автоматически после набора задачи (перед ее решением); вручную ее можно ввести последовательным нажатием клавиш "п","и".

Утверждения "$\triangle(ABC)$" (точки A, B, C образуют вершины треугольника), "параллелограмм$(ABCD)$" (точки A, B, C, D, взятые в данной последовательности, образуют вершины параллелограмма); "ромб$(ABCD)$"; "прямоугольник$(ABCD)$", "квадрат$(ABCD)$", "четырехугольник$(ABCD)$" (точки A, B, C, D образуют вершины выпуклого четырехугольника) вводятся последовательными нажатиями пар клавиш "т","р"; "п","а"; "р","о"; "п","р"; "к","в"; "ч","е" (кир.). Запятые между операндами не ставятся.

Утверждение "трапеция$(ABCD)$" (точки A, B, C, D образуют вершины трапеции, углы при большем основании которой - не тупые, причем точки A, D лежат на большем основании) вводится при помощи клавиш "т","п". Утверждение "Трапеция $(ABCD)$" (то же, но углы при большем основании - любые, а точки A, D лежат на основании, которое не обязательно большее) вводится при помощи клавиш "Т","п".

Утверждения "многоугольник(A)"(A - набор вершин простой замкнутой ломаной), "правмногоугольник(A)" (A - набор вершин правильного многоуольника) вводятся последовательными нажатиями клавиш "м","н"; "п","й". Множество точек внутри многоугольника (включая границу), определяемого набором вершин A, обозначается "фигура(A)". Точки набора A во всех перечисленных случаях отделяются друг от друга запятыми. Выражение "Фигура$\{A, B, \ldots, C\}$" (множество точек, ограниченное составной границей с частями $A, B, \ldots, C$) вводится при помощи клавиш "Ф","и".

Уиверждения "Медиана$(ABCD)$", "Высота$(ABCD)$", "Биссектреуг$(BACD)$" (точка D является основанием, соответственно, медианы, высоты либо биссектрисы треугольника ABC, проведенной из вершины A) вводятся при помощи пар клавиш "М","Е"; "В","Ы"; "И","Т".

Выражения "окружность(AB)", "круг(AB)"(окружность и круг с центром в точке A и точкой на окружности B) вводятся нажатием клавиш "о","к"; "к","р". Выражения "дуга(ABC)"(меньшая из дуг окружности с центром в точке A и концами B, C); "большаядуга(ABC)" (большая из указанных дуг); "дугаугла(ABC)" (дуга, на которую опирается вписанный угол ABC); "сектор(ABC)" (сектор окружности с центром в точке A и концами дуги B, C; берется меньшая из дуг); "сегмент(ABC)" вводятся при помощи нажатий клавиш "д","у"; "д","У"; "д","в"; "С","е"; "с","г". Выражение "кольцо(ABC)" (кольцо с центром в A, внутренняя окружность которого проходит через B, а внешняя - через C) вводится при помощи "к", "О".

Площадь $S(A)$ плоской фигуры A (например, A может иметь вид "фигура(B)", "круг(BC)", "сектор(BCD)"и т.п.) вводится при помощи двукратного нажатия клавиши "S". Выражение "периметр(A)"(A обычно имеет вид "фигура(B)") вводится при помощи последовательного нажатия клавиш "п","е". Выражение "длина(A)" (длина кривой A) вводится при помощи "д","л".

Утверждение "центр(A, B)"(точка A является центром фигуры B) вводится при помощи "ц","е".

Утверждение "A - касательная к B" вводится набором A, нажатием "Ctr-q"и

набором B. Утверждения "внешкасательная(A,B,C)" (прямая A является внешней касательной к окружностям B,C); "внутркасательная(A,B,C)" вводятся нажатиями "Ш","к"и "У","к"соответственно.

Утверждения "окружность(AB)вписана в фигура$(C_1 \dots C_n)$"; "окружность(AB) описана около фигура$(C_1 \dots C_n)$" вводятся нажатием клавиш "Ctr-э"и "Ctr-ф" соответственно (предварительно вводится "окружность(AB)").

Утверждения "внешкасаются(A,B)" (внешнее касание окружностей) и "внутркасаются(A,B)" вводятся последовательными нажатиями клавиш "ш","к"; "у","к".

Утверждения "выпукло(A)", "осьсимметрии(A,B)" вводятся при помощи клавиш "в","ы"; "С","и".

Утверждения $A \sim B$ (подобие фигур) и $A \equiv B$ (конгруентность фигур) вводятся при помощи клавиш "Ctr-п" и "Ctr-к" (кир.).

13. Стереометрия. Выражение "плоскость(ABC)" (плоскость, проходящая через точки A,B,C) вводится при помощи клавиш "п","л". Для ввода выражения "плоскостьфигуры(a)" (плоскость, в которой расположена плоская фигура a) используются клавиши "п","ф". Выражение "уголмежду(a,b)" (величина острого угла между прямыми либо плоскостями a,b) вводится при помощи "у","м".

В стереометрии для задания окружности мало указания ее центра A и точки B на окружности - нужно задать еще плоскость окружности, для чего в решателе используется ссылка на некоторую третью точку C, лежащую в данной плоскости. Соответственно, возникают выражения "Окружность(ABC)"и "Круг(ABC)", для ввода которых используются клавиши "О","к"и "К","р".

Утверждение "биссектрплоск(C прямая(AB) D E)"(биссекторная плоскость двугранного угла, опирающегося на прямую AB и ограниченного двумя полуплоскостями, содержащими, соответственно, точки C и D, проходит через точку E) вводится при помощи двукратного нажатия клавиши "И". Утверждение "плоскостьсимметрии (A,B)" (A есть плоскость симметрии поверхности либо тела B) вводится клавишами "П","И".

Выражение "объем(a)" вводится двукратным нажатием клавиши "О"(кир.). Выражения "площповерхности(a)", "боковаяповерхность(a)" (площадь боковой поверхности тела a), "высота(a)"(высота тела a), "радиус(a)"(радиус сферы либо шара a) вводятся, соответственно, при помощи пар клавиш "щ","п"; "Щ","п"; "В","ы".

Утверждения "грань(a,b)" (фигура a является гранью многогранника b), "основание(a,b)" (фигура a является основанием многогранника b) вводятся при помощи клавиш "г","р"; "О","с"(кир.). Утверждения "ребро(a,b)" (отрезок a есть ребро многогранника b); "вершина(a,b)" (точка a есть вершина многогранника b); "Вершина(a,b)" (точка a является вершиной пирамиды либо конуса b); "диагональ(a,b)" (отрезок a есть главная диагональ параллелепипеда b) вводятся парами клавиш "р","е"; "в","ш"; "В","ш"; "д","и". Утверждение "осевоесечение(a,b)" (прямоугольник либо треугольник a является осевым сечением, соответственно, цилиндра либо конуса b) вводится клавишами "С","Е".

Утверждения "куб(a)", "призма(a)", "параллелепипед(a)", "пирамида(a)", "усечпирамида(a)", "конус(a)", "шар(a)", "сфера(a)"вводятся, соответственно, при помощи нажатий пар клавиш "к","у"; "з","м"; "п","д"; "р","м"; "к","с"; "ш","а"; "ф","р".

Утверждения "прямой(a)" (прямая призма, прямой параллелепипед); "правильный(a)" (правильная призма, правильная пирамида, правильная усеченная пирамида); "прямоугольный(a)" (параллелепипед) вводятся при помощи пар клавиш "п", "Я"; "п","в"; "п","я".

Выражение "трехгранугол($ABCD$)" (A - вершина трехгранного угла; AB,AC,AD - направляющие векторы сторон) вводится при помощи "т","у". Выражения "двугрУгол(A,B,C,d)" (двугранный угол между плоскостями A,B, выделяемый при помощи точки C и указателя d ее принадлежности: 0 - внутри угла, 1 - внутри угла, смежного с данным через A, 2 - внутри угла, смежного с данным через B, 3 - внутри угла, вертикального с данным), "двугругол(A,B,C,d)" (величина указанного выше двугранного угла), "двугранУгол(A,B,C)" (двугранный угол, в основании которого лежит прямая B, а направления сторон определяются лежащими на них точками A,C), "двугранугол(A,B,C)"(величина указанного выше двугранного угла) вводятся при помощи пар клавиш "Д","У"; "Д","у"; "Д","Г"; "Д","г".

Выражения "полупространство(A,B)" (полупространство, отделенное плоскостью A и содержащее точку B) и "обрполупространство(A,B)"(полупространство, отделенное плоскостью A и не содержащее точки B) вводятся при помощи клавиш "П","В"; "О","П". Выражение "Полоса(A,B)" (полоса между параллельными плоскостями A,B) вводится при помощи клавиш "П","С".

14. Системы координат. Выражение "коорд(A,K)" (координаты точки A либо множество координат точек множества A в аффинной системе координат K) вводится при помощи клавиш "к","д". Заметим, что аффинная система координат отождествляется в решателе в двумерном случае с тройкой точек (P,Q,R) (P - начало системы координат; PQ, PR - координатные векторы), а в трехмерном - с четверкой точек (P,Q,R,S). При необходимости в список посылок задачи заносится равенство, задающее K в виде такой тройки либо четверки. Утверждение "прямкоорд(K)" (K - прямоугольная система координат) вводится при помощи клавиш "п","к".

Выражение "полкоорд(A,K)" (координаты точки A в полярной системе координат K) вводится при помощи клавиш "п","о". Полярная система координат K отождествляется с тройкой точек (P,Q,R), где P - начало системы координат, а полярный угол отсчитывается в направлении от луча PQ к лучу PR.

Выражение "сферкоорд(A,K)" (координаты точки A в сферической системе координат K) вводится при помощи "ф","к". Сферическая система координат отождествляется с четверкой точек (P,Q,R,S), где P - начало системы координат, широта отсчитывается от луча PQ в направлении луча PR, а долгота - в направлении от луча PS к плоскости PQR.

Выражение "цилкоорд(A,K)"(координаты точки A в цилиндрической системе координат K) вводится при помощи "ц","к". Цилиндрическая система координат отождествляется с четверкой точек (P,Q,R,S), где P - начало координат; полярный угол отсчитывается от луча PQ в направлении луча PR; PS - координатная ось.

Выражение "точки(A,K)"(множество точек, координаты которых в аффинной системе координат K образуют множество A) вводится при помощи двукратного нажатия клавиши "т".

15. Векторы. Выражение "вектор(AB)" вводится при помощи "в","е"(кир.). Операции "плюсвект", "минусвект", "умножвект" сложения векторов, изменения знака вектора и умножения вектора на число (это число является обязательно первым сомножителем) вводятся формульным редактором так же, как обычные операции сложения, изменения знака и умножения. Формульный редактор самостоятельно преобразует их в указанные выше, усмотрев наличие векторных операндов; если этого ему сделать не удалось, то коррекцию обозначений данных операций можно осуществить вручную, перейдя после набора формулы в режим текстового редактора. Утверждение "Вектор(A)" (A есть вектор) вводится при помощи "В","е".

Нулевой вектор "вектор0"вводится тектовым редактором. Величина "уголмежду

(A,B)"угла между векторами вводится при помощи "у","м". Условие ортогональности векторов $A\perp B$ вводится так же, как условие перпендикулярности прямых (Ctr-т; т - кир.).Утверждения "коллинеарны(A,B)", "компланарны(A,B,C)" вводятся при помощи клавиш "к","л"и "к",п". Длина вектора "длина(A)"вводится при помощи "д","л".

Выражения "скалумнож(A,B)" (скалярное произведение векторов) и "вектумнож (A,B)" вводятся при помощи клавиш "с","к"и "в","Е"(кир.).

Выражение "ориентация(A,K)" (ориентация упорядоченного набора A векторов в аффинной системе координат K - для двумерного либо трехмерного случаев; эта ориентация принимает значения 1 и -1) вводится при помощи "О","р" (кир.). Выражение "оруголмежду(A,B,K)" (величина ориентированного угла между векторами A,B при ориентации плоскости, задаваемой системой координат K) вводится при помощи "О","м". Выражения "орплощадь(K,A,B)" (ориентированная площадь параллелограмма, порожденного векторами A,B при положительной ориентации, задаваемой системой координат K), "оробъем(K,A,B,C)" (аналогично для параллелепипеда, порожденного векторами A,B,C) вводятся текстовым редактором. Так же вводится и утверждение "направление(A,B,C,D)" (вектор D трехмерного пространства направлен по отношению к плоскости векторов A,B в ту же сторону, что и вектор C).

Утверждение "делениеотрезка(A,B,C,a,b)" (точки A,B,C лежат на прямой и отношение вектора AB к вектору BC равно отношению a к b) вводится при помощи "д","О". Выражение "конецвектора(A,B)" (конец вектора B, отложенного от точки A) вводится текстовым редактором. Ортогональная проекция "проекция(A,B)" точки, прямой либо вектора A на прямую либо плоскость B вводится при помощи "п","Р".

16. Уравнения линий. Утверждение "Прямая(A)"(множество точек A есть прямая на плоскости либо в пространстве) вводится при помощи клавиш "П","Р". Уравнение прямой AB на плоскости обычно вводится в виде "коорд(прямая(AB),K) = $set_{xy}(ax+by+c=0\ \&\ x-$число $\&\ y-$число)". Если переменная x либо y явно входит в равенство под описателем "*set*", то указывать численный тип ее значения не обязательно - такое указание будет сформировано автоматически в начале решения задачи. Аналогичным образом задаются уравнения других плоских линий. В трехмерном случае уравнение прямой, проходящей через точку с координатами (a,b,c) и имеющей направляющий вектор (d,e,f), обычно записывается в виде "коорд(прямая(AB),K) = $set_{xyz}($
пропорцнаборы$((x-a,y-b,z-c),(d,e,f))\ \&\ x-$число $\&\ y-\ \&\ z-$число)". В случае отличных от нуля a,b,c указатель численного типа значения соответствующей переменной x,y,z можно опускать. Логический символ "пропорцнаборы"(указание на линейную зависимость двух числовых наборов одинаковой длины) вводится клавишами "п","Н"(кир.).

Утверждение "направлпрямой(A,B,C)"(C - координаты направляющего вектора прямой A в системе координат B) вводится с помощью клавиш "н","П".

Утверждение "линвторпорядка(A)"(A - линия второго порядка) вводится текстовым редактором. Утверждения "эллипс(A)", "гипербола(A)", "равнгипербола(A)" (равносторонняя гипербола), "парабола(A)"вводятся, соответственно, при помощи пар клавиш "Е","Л"; "Г","И"; "Р","Г"; "П","А".

Выражения "фокпараметр(A)" (фокальный параметр кривой A), "полуось(A,B)" (величина полуоси эллипса B в направлении его оси A), "эксцентриситет(A)" (эксцентриситет кривой A), "фокхорда(A)" (длина хорды, проходящей через фокус кривой

второго порядка A перпендикулярно фокальной оси) вводятся при помощи пар клавиш "ф","п"; "П","О"; "е","к"; Ф","Д". Утверждение "фркус(A,B)" (A есть фокус кривой второго порядка B) вводится при помощи "ф","о".

Утверждения "направлпараболы(A,B)" (вектор B задает направление оси параболы A), "действось(A,B)" (прямая A есть действительная ось гиперболы B), "мнимаяось(A,B)", "директриса(A,B,C)" (прямая A есть директриса кривой второго порядка C, соответствующей фокусу B), "асимптота(A,B)"(прямая A является асимптотой кривой B) вводятся, соответственно, при помощи пар клавиш "Н","п"; "Д","О"; "М","О"; "Д","И"; "А","С".

Выражение "ветвькривой(A,B)" (ветвь кривой B, содержащая точку A) вводится при помощи клавиш "Е","К".

Утверждения "диаметр(A,B)" (прямая A является диаметром кривой второго порядка B), "сопряжнаправления(A,B,C)" (направления прямых A,B являются сопряженными относительно кривой второго порядка C)вводятся при помощи клавиш "Д","М"; "С","Н"(кир.).

Выражения "большаяось(A)" (длина большой оси эллипса A), "малаяось(A)", "действполуось(A)" (длина действительной полуоси гиперболы A), "мнимаяполуось (A)" вводятся текстовым редактором.

Утверждение "каноничкоорд(A,B)" (A есть каноническая система координат для кривой либо поверхности второго порядка B) вводится при помощи клавиш "К","Ч". Утверждение "фоксистема(A,B)" (A есть прямоугольная система координат с центром в фокусе кривой второго порядка B, положительное направление оси абсцисс которой в случае параболы совпадает с направлением оси параболы; в случае эллипса совпадает с направлением на другой фокус; в случае гиперболы противоположно этому направлению) вводится с помощью текстового редактора.

17. Уравнения поверхностей. Утверждение "Плоскость(A)" (A есть множество точек плоскости в трехмерном пространстве) вводится последовательным нажатием клавиш "П","Л". Уравнение плоскости вводится в виде "коорд(поскость(ABC),K) $= set_{xyz}(ax+by+cz+d=0$ & $x-$число & $y-$число & $z-$число)". Если переменная x,y,z явно входит в равенство под описателем "set", то указывать численный тип ее значения не обязательно. Аналогичным образом задаются уравнения других поверхностей. Уравнения кривых в трехмерном пространстве задаются с помощью двух равенств под описателем "set".

Утверждение "плоскбазис(P,K,a,b)" (a,b - два координатных набора для ортогональных единичных векторов, лежащих на плоскости P, относительно прямоугольной системы координат K в трехмерном пространстве) вводится при помощи клавиш "П","о".

Утверждения "Цилиндр(a)" (a - цилиндрическая поверхность), "Конус(a)" (a - коническая поверхность), "эллипсоид(a)", "гиперболоид(a)", "параболоид(a)"вводятся, соответственно, при помощи пар клавиш "Ц","И"; "К","О"; "Е","Д"; "Г","Д"; "П","Д". Утверждение "повторпорядка(a)"вводится текстовым редактором.

Уточняющие тип поверхности утверждения "круглый(a)" (цилиндр, конус), "эллиптический(a)", "гиперболический(a)", "параболический(a)", "однополостный(a)", "двуполостный(a)" вводятся, соответственно, при помощи пар клавиш "к","р"; "Е","Ч"; "Г","Ч"; "П","Ч"; "О","Й"; "Д","Й".

Утверждения "осьцилиндра(A,B)" (прямая A - ось цилиндра B), "направляющая(A,B)" (кривая A - направляющая цилиндра B), "образующая(A,B)" (прямая A - образующая цилиндра B), "оськонуса(A,B)" (прямая A - ось конуса B), "осьвращения(A,B)" (прямая A - ось вращения поверхности B) вводятся, соответственно, при

помощи пар клавиш "О","Ц"; "Н","П"; "О","Щ"; "О","К"; "о","в"(кир.).

Утверждение "направлпараболоида(A, B, C)" (координатный набор C определяет направление оси параболоида A в системе координат B) вводится с помощью текстового редактора.

В случае эллипсоида вращения A длина той оси эллипса, относительно которой происходит вращение, обозначается "высота(A)" ; длина другой полуоси эллипса - "радиус(A)".

Выражение "полуось(A, B)" (длина полуоси эллипсоида B по направлению его оси A) вводится при помощи "П","О".

Утверждения "диаметрплоскость(A, B)" (A есть диаметральная плоскость поверхности второго порядка B), "сопряжнаправления(A, B, C)" (направления плоскости A и прямой B являются сопряженными относительно поверхности второго порядка C) вводятся при помощи пар клавиш "д","п"; "С","Н"(кир.).

Утверждение "собствкоорд(A, B)" (A есть прямоугольная система координат, единичные векторы которой задают направления главных осей поверхности второго порядка B) вводится текстовым редактором.

18. Матрицы. Матрица переменной или чрезмерно большой размерности вводится как обычная функция двух натуральных аргументов - $\lambda_{ij}(a(i,j), i \in 1,\ldots,n \;\&\; j \in 1,\ldots,m)$. Если размеры матрицы невелики, то ее можно прорисовать на экране стандартным образом. Для этого сначала нажимаются последовательно клавиши "м","а" (кир.) - появляется левая скобка матрицы. Затем набираются элементы первой строки матрицы. Вместо разделителя после каждого элемента нажимается "Enter". После последнего элемента строки вместо "Enter"нажимается "Page Down для перехода к следующей строке. Если строка была последней, то вместо "Page Down"после ввода ее последнего элемента нажимается "End". Размеры скобок матрицы и уровень текущей строки корректируются во время набора автоматически. Наконец, можно прорисовать матрицу константных наборов путем перечисления ее строк либо столбцов - в виде "строки$((A_1 1 \ldots A_{1n}) \ldots (A_{m1} \ldots A_{mn}))$"либо "столбцы $((A_1 1 \ldots A_{m1}) \ldots (A_{1n} \ldots A_{mn}))$". Логические символы "строки"и "столбцы"вводятся: первый - путем двукратного нажатия клавиши "м", второй - "М" (кир.).

Выражение "ранг(A)"вводится при помощи двукратного нажатия "р"(кир.).

Операции сложения матриц и изменения знака матрицы - те же, что и для произвольных веществennозначных функций ("плюсфунк", "минусфунк"). Они вводятся так же, как обычные операции сложения и изменения знака, и в начале решения задачи предпринимается попытка автоматической их коррекции. Это же относится и к операциям умножения матрицы на число ("матркоэфф") и умножения двух матриц ("умножматр").

Утверждения "собствзначение(A, a, k)" (a есть собственное значение матрицы A, имеющее кратность k) и "собствектор(A, a, v)" (v есть собственный вектор - столбец, соответствующий собственному значению a матрицы A) вводятся при помощи клавиш "с","з"и "с","в".

Выражение "линкомбинации(A)" (множество линейных комбинаций с вещественными коэффициентами матриц множества A), а также утверждения "каноничматрица(A, x, B)" (B - матрица, являющаяся канонической формой матрицы A, элементы которой рассматриваются как многочлены относительно переменной x) и "жордформа(A, B, C)" (матрица B является канонической жордановой формой матрицы A, причем C - матрица, трансформирующая A в B) вводятся текстовым редактором.

19. Комплекные числа. Утверждение "a - комплексное" вводится в следующем порядке: сначала набирается a, затем "\", затем "к" (кир.). Мнимая единица вводит-

ся двукратным нажатием клавиши "c". Арифметические операции и элементарные функции для комплексных чисел вводятся так же, как для вещественных; преобразование их в комплекснозначные операции внутреннего языка решателя выполняется автоматически (хотя и требует контроля).

Выражения $Re(a)$, $Im(a)$, $Arg(a)$ (главное значение аргумента) вводятся, соответственно, при помощи клавиш "R","e"; "I","m"; "A","r"(лат.). Выражение "Корень(a, n, k)" (k-й корень натуральной степени n из комплексного числа a) вводится при помощи "К","о" (кир.). Выражение "сопряженное(a)"вводится при помощи "с","о" (кир.).

3.4.5 Текстовый редактор

В решателе используется интерфейс текстового редактора, несколько отличающийся от стандартных версий. Этот редактор используется, главным образом, для работы с программами приемов, однако он может понадобиться и при вводе элементов задачи. Приведем краткое описание основных его функций.

Текстовый редактор позволяет работать только с текстами, целиком помещающимися на экране. Для работы с большими текстами использующие текстовый редактор процедуры имеют режим "перелистывания", определенный в описании их интерфейса.

При входе в текстовый редактор появляется прямоугольный курсор светло - коричневого цвета и автоматически устанавливается режим кириллицы; для перехода в режим латиницы нажимается F12, для возвращения в режим кириллицы - F11. Для ускорения перемещений курсора вводится режим увеличенного шага (5-кратное смещение по вертикали либо 8-кратное по горизонтали). Переключение между одношаговым режимом и ускоренным осуществляется нажатием клавиши "TAB". Завершение редактирования происходит при нажатии клавиши "Enter"; отмена редактирования - при нажатии "Esc". При всех операциях с текстом выполняется принцип: позиция, следующая за крайней правой позицией строки, есть начальная позиция следующей строки. При достижении правого края окна редактирования курсор автоматически переходит в начало следующей строки этого окна.

Для вставки пустой строки начиная с текущей позиции (все, что было правее этой позиции, смещается на одну строку вниз) используется клавиша F7. Для исключения одной строки начиная с текущей позиции (все, что было на следующей строке правее этой позиции, сдвигается вверх на одну строку) нажимается F6. Переход в начало либо в конец строки - как обычно, с помощью "Home"и "End".

Для вставки текста перед заданной позицией курсор перемещается к этой позиции, и нажимается клавиша "Insert". Курсор после этого становится голубым - включается режим вставки. В этом режиме каждый вводимый с клавиатуры символ вставляется непосредственно перед курсором (который сдвигается так, что расположен все время над исходным символом). По завершении вставки снова нажимается "Insert" - курсор перекрашивается в светло-коричневый цвет. Если во время вставки нажать "Backspace для отмены последнего введенного символа, то также происходит выход из режима вставки. В режиме вставки курсор не реагирует на нажатия клавиш сдвига курсора.

Для удаления фрагмента текста курсор сначала подводится к началу этого фрагмента (первый удаляемый символ) и нажимается клавиша "Delete". Курсор перекрашивается в красный цвет. Фактически, это лишь маркер начала удаляемой части, наложенный поверх курсора, а курсор (светло-коричневого цвета) обычным обра-

зом реагирует на клавиши сдвига. Далее курсор подводится к концу удаляемого фрагмента (первый сохраняемый символ) и снова нажимается "Delete" - выбранный фрагмент при этом удаляется. Если после выбора начала удаляемой части нажать "Backspace", то удаление отменяется, и красный маркер исчезает.

Возможна операция перенесения выбранной части текста на новое место. Для ее выполнения сначала курсор подводится к началу этой части и нажимается F2. Курсор перекрашивается в светло-зеленый цвет. Фактически, это лишь наложенный поверх курсора маркер начала переносимой части. Затем курсор подводится к концу переносимой части (последний относящийся к ней символ) и снова нажимается F2. Появляется наложенный поверх курсора светло-зеленый маркер конца. Наконец, курсор подводится к тому месту, начиная с которого должен быть расположен выбранный фрагмент, и снова нажимается F2 - при этом происходит перенесение фрагмента на данное место. На каждом этапе нажатие "Backspace" удаляет последний введенный маркер.

Для получения подсказки о клавиатуре текстового редактора (если уже началось его использование) нажимается F3. После этого нажатие любой клавиши возвращает в прерванное редактирование. Для получения справочной информации, связанной с заданным логическим символом, нажимается клавиша F4. Возникает диалогове окно, в котором вводится название нужного логического символа. По окончании ввода нажимается "Enter", и появляется необходимая справочная информация. Выход из ее просмотра - по нажатию любой клавиши, кроме клавиш, используемых при просмотре такой информации (например, при нажатии клавиши "пробел"). При этом восстанавливается прерванное редактирование.

Предусмотрен буфер для сохранения фрагмента текста. Практически во всех случаях содержимое этого буфера при смене различных интерфейсов системы, и его можно использовать при любом повторном обращении к текстовому редактору. Для занесения фрагмента текста в буфер сначала курсор подводится к началу этого фрагмента и нажимается клавиша "PageDown". Появляется синий маркер начала фрагмента, наложенный поверх курсора. Затем курсор подводится к концу фрагмента и снова нажимается "PageDown" - маркер начала исчезает, и буфер сохраняет выбранный фрагмент. Если до второго нажатия "PageDown" нажать "Backspace", то маркер исчезает и операция отменяется. Для извлечения содержащегося в буфере фрагмента курсор подводится на ту позицию, начиная с которой должен быть вставлен текст. Затем нажимается "PageUp". Содержимое буфера при этом сохраняется.

Возможно изменение цвета фона (Ctr-F4) и символов (Ctr-F5). Эти цвета меняются циклическим образом (длина цикла 16). Если внешняя процедура, обратившаяся к текстовому редактору, игнорирует цвета, то они при сохранении в файле будут утеряны.

Ряд специальных символов вводится с помощью клавиш "Ctr- ...". Так, символ $\neg$ вводится нажатием клавиши "Ctr-n" ; символ $\exists$ - нажатием "Ctr-e" (лат.); символ $\forall$ - нажатием "Ctr-a" (лат.). Заметим, что перечисленные символы используются только при наборе текстовым редактором обычных текстов; при наборе термов вместо них должны использоваться слова "не" , "существует" ,"длялюбого".

Если был набран некоторый текст, то при сохранении в файле автоматически отбрасывается его пустой конец. Если такой конец все же нужно сохранить в файле, то для обозначения его завершающей позиции используется символ $\perp$, вводимый клавишей F9.

Каждому коду клавиатуры в языке программирования ЛОС (описываемом в последующих разделах) сопоставлен некоторый логический символ. Если при про-

граммировании процедуры интерфейса нужно определить этот символ, то курсор текстового редактора подводится на ту позицию, начиная с которой должно быть записано название символа; нажимается клавиша F8, и сразу же за этим - нажимается та клавиша либо сочетание клавиш, для которой нужно узнать кодирующий ее логический символ.

Термы вводятся текстовым редактором только в скобочной записи $F(A_1 \dots A_m)$. Логический символ F вводится в виде слитно написанного слова; между операндами A_i необходимы разделители, причем разделителем служит пробел либо скобка. Запятые в качестве разделителей не допускаются (запятая сама является логическим символом). Числа вводятся как последовательности расположенных подряд цифр. В случае десятичных дробей используется запятая, а не точка, как в формульном редакторе. Знак "минус", как и прочие операции, применяется в сочетании со скобкой: "минус(A)". Для удобства работы со скобками предусмотрена операция перехода от скобки к парной ей скобке. Курсор помещается на одну из скобок, и нажимается клавиша F1, переводящая его на парную скобку.

3.4.6 Геометрический редактор

Геометрический редактор позволяет создавать простые чертежи, состоящие из некоторой системы обозначенных буквами точек, между которыми проведены линии (отрезки прямых либо окружности). Чертежи используются в задачах по геометрии и в описании геометрических приемов. Какой-либо функциональной нагрузки в работе решателя они пока не имеют, а используются лишь для обеспечения необходимой наглядности. Тем не менее, при выполнении по ходу решения задачи дополнительных построений решатель сам пополняет чертеж новыми точками и линиями. Все, что нужно для этого сделать - ввести в описании задачи чертеж, обозначения точек на котором согласованы с используемыми в условии задачи.

Если вводится новая геометрическая задача, для которой нужен чертеж, то лучше всего начинать ее набор прямо с чертежа. Для этого достаточно ввести бланк задачи (прорисована верхняя горизонтальная линия, обозначающая начало задачи) и нажать клавишу "ч". Если уже была набрана часть текста задачи, то вход в геометрический редактор осуществляется тем же самым нажатием клавиши "ч". В обоих случаях вся отведенная для задачи часть экрана занимается прямоугольником геометрического редактора, а набранная ранее часть текста задачи пропадает с экрана. Она будет восстановлена по окончании (либо отмене) создания чертежа и размещена под чертежом. Можно попробовать создать чертеж с помощью автоматической процедуры, имеющейся в решателе. Для этого нажимается клавиша "Ч". Если до этого была введена ручная версия чертежа, то она будет заменена автоматической; восстановление предыдущей версии чертежа при этом достигается путем выделения чертежа (левой клавишей мыши) и нажатия "Ctr-Del". Если нужно вручную изменить ранее введенный чертеж задачи (в том числе, автоматически введенный), то, как и при первоначальном вводе чертежа, нажимается "ч".

Заметим, что по окончании редактирования чертежа либо при отмене начатого редактирования автоматически выполняется прокрутка списка задач, так что в верхней части экрана оказывается начало обрабатываемой задачи. Указанным выше способом можно входить в редактирование чертежа только для последней задачи просматриваемого списка задач. Если же задача не последняя, то ее следует сначала выделить (нажатие правой клавиши мыши в любой точке поля задачи), и лишь затем нажать "ч".

Окно геометрического редактора представляет собой прямоугольник, ограниченный линией синего цвета. В правом верхнем углу отделена небольшая область, в которой появляется указатель текущего режима редактирования. Управление сменой режимов осуществляется с помощью меню геометрического редактора, появляющегося в верхней части экрана. Предусмотрены следующие режимы работы:

1. Нейтральный режим. Этот режим устанавливается при входе в геометрический редактор. У него прямоугольник указателя режима - пустой. Переход из любого другого режим в этот режим обеспечивается нажатием клавиши пробела. В нейтральном режиме можно выполнять следующие операции:

а) Перемещение точки либо сопровождающего ее буквенного обозначения. Для этого точка выделяется нажатием левой клавиши мыши после подведения к ней курсора мыши. Затем можно сдвигать точку нажатием клавиш курсора (постоянное удержание клавиши нажатой приводит к медленному движению точки в выбранном направлении). Аналогично, можно сдвигать обозначение выделенной точки нажатием клавиш "Ctr-курсоры". При движении точки корректируются все ведущие к ней отрезки прямых.

б) Изменение обозначения точки. Сначала нажимается буквенная клавиша (быть может, сопровождаемая несколькими цифровыми при указании индекса) для нового обозначения, затем выделяется нужная точка (левая клавиша мыши) и нажимается -". Как и в случае добавления новой точки (см. ниже), для ввода большой буквы на клавиатуре нажимается малая буква, и наоборот. Старое обозначение заносится в накопитель обозначений точек, и будет использовано при последующих вводе либо коррекции обозначения (если не сбросить накопитель с помощью клавиши "Delete").

Эти операции можно применять и в других режимах (но не в режиме ввода новых точек, так как тогда каждая попытка выделить точку будет приводить к созданию новой точки).

2. Режим ввода новых точек. Вход в режим ввода новых точек осуществляется при нажатии клавиши "Insert"либо выборе мышью пункта "Точка"меню в верхней части экрана. В прямоугольнике указателя режима появляется слово "точка". Для ввода новой точки следует выбрать курсором мыши ее позицию и нажать левую клавишу мыши. В качестве обозначения точки каждый раз выбирается первая не использованная большая буква. Однако, можно явно указывать обозначения точек, предварительно нажав последовательно некоторые буквенные клавиши. Тогда последующие нажатия левой клавиши мыши на выбранных позициях приведут к использованию для точек обозначений из данной последовательности. Так как обычно точки обозначаются большими буквами, предусмотрено автоматическое преобразование вводимых с клавиатуры малых букв в большие, а больших - в малые. Для сброса накопителя обозначения новых точек нажимается клавиша "Delete".

Обозначения точек можно сопровождать индексами - для этого после нажатия буквенной клавиши нажимаются одна или несколько цифровых клавиш, образующих индекс.

Если нужно удалить ранее введенную точку, то к ней подводится курсор мыши и нажимается правая клавиша мыши. При этом обозначение удаленной точки сохраняется в накопителе обозначений и будет использовано (если не сбросить этот накопитель) при вводе новой точки.

3. Режим сдвига точек. Переход в этот режим осуществляется нажатием клавиши "Ctr-c" либо выбором мышью пункта "Сдвиг" меню в верхней части экрана. Указатель режима прорисовывает слово "сдвиг". Далее можно изменить положение любой из точек, не прибегая к сравнительно медленному перемещению ее клавиша-

ми курсора. Для этого сначала точка выделяется мышью (нажатие левой клавиши после подведения к точке курсора мыши), затем курсор мыши подводится к новой позиции точки, и снова нажимается левая клавиша мыши.

4. Режим проведения отрезка. Переход в режим осуществляется при нажатии "Ctr-о"("о кир.) либо выборе мышью пункта "Отрезок"меню в верхней части экрана. Указатель режима прорисовывает слово "отрезок". Для проведения отрезка между двумя точками сначала выделяется первая из них, затем курсор мыши подводится ко второй и нажимается левая клавиша мыши. Если нужно не ввести, а наоборот, удалить ранее введенный отрезок, то после выделения первой точки отрезка на второй точке нажимается правая клавиша мыши.

5. Режим проведения прямой. Переход в режим осуществляется при нажатии клавиши "Ctr-п" либо выборе мышью пункта "Прямая" меню в верхней части экрана. Указатель режима прорисовывает слово "прямая". Режим аналогичен режиму проведения отрезков, однако здесь отрезок продолжается вплоть до границ рамки. Режим практически не используется.

6. Режим проведения перпендикуляра. Переход в режим осуществляется при нажатии клавиши "Ctr-т" либо выборе мышью пункта "Перпендикуляр" меню в верхней части экрана. Указатель режима прорисовывает слово "перпендикуляр". Возможны две операции: восставить перпендикуляр к прямой в заданной точке этой прямой, либо опустить из заданной точки перпендикуляр на заданную прямую.

В первом случае левой клавишей мыши выбираются две различные точки на прямой (вторая из них - та, через которую пройдет перпендикуляр). Чтобы доопределить направление перпендикуляра и ввести на нем еще одну точку, курсором мыши выбирается какая-либо из уже введенных ранее точек, и на ней нажимается правая клавиша мыши. Тогда перпендикуляр к прямой, проходящей через первые две точки, будет проведен из второй точки в направлении третьей точки, причем на нем будет введена ближайшая к третьей точке новая точка.

Во втором случае, как и в первом, сначала левой клавишей мыши выбираются две точки на прямой, к которой требуется провести перпендикуляр, и затем на третьей точке - из которой должен быть опущен перпендикуляр - нажимается левая клавиша мыши. При проведении перпендикуляра вводится новая точка - основание этого перпендикуляра.

7. Режим проведения окружности. Переход в режим осуществляется при нажатии клавиши "Ctr-к"("к кир.) либо выборе мышью пункта "Окружность"меню в верхней части экрана. Указатель режима прорисовывает слово "окружность". Для проведения новой окружности сначала левой клавишей мыши выбирается ее центр, затем берется какя-либо точка, через которую должна пройти окружность, и снова нажимается левая клавиша мыши. Чтобы удалить введенную ранее окружность, сначала левой клавишей мыши выбирается ее центр, затем на той точке окружности, с помощью которой она вводилась, нажимается правая клавиша мыши. Последняя операция возможна лишь в том случае, если ранее введенная окружность не перемещалась по экрану (такое перемещение происходит при перемещении ее центра); иначе для удаления окружности нужно удалить ее центр).

8. Режим параллельного перемещения всего изображения. Переход в режим осуществляется при нажатии клавиши "Home"либо выборе мышью пункта "Перемещение"меню в верхней части экрана. Указатель режима прорисовывает слово "перемещение". Для задания вектора перемещения сначала на некоторой позиции нажимается клавиша мыши, затем курсор мыши переносится на новую позицию, и снова нажимается клавиша мыши.

9. Режим изменения размеров рамки чертежа. Переход в режим осуществляется при выборе мышью пункта "Рамка" меню в верхней части экрана. Возникает подменю, в котором перечисляются названия четырех сторон рамки ("правыйкрай", "левыйкрай", "верхнийкрай"и "нижнийкрай"). После выбора в нем нужной стороны указатель режима повторяет название выбранной стороны. Для выбора нового положения этой стороны курсор мыши перемещается на требуемую позицию и нажимается левая клавиша мыши. Режим обычно используется для увеличения зоны чертежа, под которым размещены другие изображения. Это увеличение относится только к периоду редактирования; по завершении работы геометрического редактора нижняя граница чертежа выбирается автоматически по крайним снизу его точкам.

Глава 4

Общая схема функционирования решателя

4.1 Сканирование задачи

При обычном программировании для решения задач заданного типа разрабатывается один общий алгоритм, который гарантирует получение за конечное число шагов ответа в случае любой конкретной задачи этого типа. Такой алгоритм представляется в виде нескольких взаимодействующих между собой блоков, обеспечивающих циклический процесс постепенного упрощения параметров решаемой задачи (в частности, уменьшения оставшегося объема перебора для процедур "переборного"типа) - вплоть до получения окончательного ответа. Каждый из блоков несет здесь вполне определенную функциональную нагрузку, как правило, основанную на том или ином теоретическом утверждении, связанном с обрабатываемыми алгоритмом объектами. Его можно рассматривать как прием для достижения определенной текущей подцели. В свою очередь, отдельный блок сам определяется схемой своих подблоков, и т.д. Здесь возникает иерархия уровней, в которой на каждом уровне несколько приемов (как правило, не более одного - двух десятков) оказываются связаны в вычислительный цикл, так что после выполнения очередного приема известно, к какому приему следует переходить дальше.

К сожалению, такого рода четкую программу действий удается составить лишь для достаточно узких классов задач. Хорошо известны результаты об алгоритмической неразрешимости различных общих проблем, и даже в случае наличия алгоритма он часто оказывается практически неприемлемым по своей трудоемкости. Для преодоления возникающих здесь трудностей приходится резко увеличивать количество используемых приемов, чтобы охватить алгоритмическими возможностями если не все многообразие задач предметной области, то хотя бы возможно больший класс задач, встречающихся в реальных ситуациях. Эти приемы создаются уже не путем "теоретического"проектирования алгоритма, а извлекаются из последовательности конкретных обучающих задач и складываются в одну общую базу приемов. Количество приемов, используемых для решения задач в предметной области, обычно существенно больше чем в алгоритмической процедуре указанного выше "блочного"типа - сотни и тысячи вместо десятков. Последовательность их работы заранее не фиксирована, и после срабатывания очередного приема необходим цикл поиска следующего применяемого приема. Разумеется, при увеличении числа приемов, работающих в общем процессе, существенно усложняется регулировка взаимодействия между ними, и возникает необходимость в длительной эмпирической оптимизации их решающих

правил.

Для организации цикла поиска очередного применяемого приема в решателе используется процедура сканирования задачи, которую можно представлять как своего рода модель внутреннего логического зрения. Для большинства приемов активизация их при рассмотрении задачи начинается с обнаружения в логических структурах данных некоторого логического символа, заранее выбранного для этого приема. В качестве такого "ключевого" логического символа берется обычно некоторый логический символ, появление которого необходимо для возможности применения рассматриваемого приема, причем при нескольких возможных выборах предпочтение отдается наиболее редко встречающемуся логическому символу (для уменьшения потерь времени при попытках применения приема). Указанное закрепление за приемами логических символов предопределяет организацию всей базы приемов решателя по принципу энциклопедии: она распадается на группы приемов, "принадлежащих" соответствующим символам, причем каждая группа реализуется в виде отдельной алгоритмической процедуры A_f - программы логического символа f. В особых случаях прием не удается связать с каким-либо конкретным логическим символом, появление которого является необходимым для срабатывания. Такие приемы (их совсем немного) распределены по четырем логическим символам - названиям "доказать", "описать", "преобразовать", "исследовать" типов задач, при решении которых возможно их применение.

Текущая ситуация, возникающая в процессе работы решателя, описывается последовательностью задач $Z_1, Z_2, \ldots, Z_N$, где Z_{i+1} - вспомогательная задача, введенная при решении задачи Z_i $(i = 1, \ldots, N-1)$. В этой последовательности (далее называем ее цепью задач) задача Z_1 является фиктивной - она создается автоматически при запуске логической системы и используется для организации интерфейса. Эта задача (далее называемая исходной задачей) имеет тип "исследовать"и единственную однобуквенную посылку "вход". При просмотре посылок данной задачи решатель обращается к программе логического символа "вход", которая и является программой интерфейса логической системы. С помощью интерфейса можно ввести некоторую задачу Z_2, которая далее и решается, порождая вспомогательные задачи $Z_3, \ldots, Z_N$. Будем называть задачу Z_N текущей задачей; Z_2 - корневой задачей. Исходная задача, кроме организации интерфейса, выполняет функции "доски объявлений" - различные процедуры решателя могут обмениваться между собой сообщениями, размещая их в списке комментариев к посылкам этой задачи.

Каждая задача Z_i $(i = 1, \ldots, N)$ характеризуется натуральным числом M_i, называемым ее максимальным уровнем и определяющим уровень средств, отведенных для ее решения (по исчерпании этих средств выдается отказ), целым неотрицательным числом m_i, $m_i \leq M_i$, называемым текущим уровнем этой задачи и определяющим уровень средств, среди которых в текущий момент ведется поиск очередного преобразования задачи Z_i, а также вспомогательной информацией, необходимой для возобновления прерванной процедуры решения задачи Z_{i-1} по окончании решения задачи Z_i.

Текущий уровень задачи является одним из входных параметров, получаемых приемами; он учитывается решающими правилами приемов и позволяет организовать необходимые приоритеты в их применении: при меньших значениях этого уровня срабатывают приемы с большим приоритетом. В процессе обучения решающие правила приемов корректируются таким образом, чтобы на каждом шаге выбирался прием, наиболее целесообразный с точки зрения обучающего систему эксперта. При первоначальном обращении к задаче Z_i ее текущий уровень равен 0.

Изменение текущей ситуации происходит в следующем рабочем цикле решателя:

1) Происходит обращение к программе логического символа f - типа задачи Z_N. Если при этом не срабатывает ни один из приемов, либо сработавшие приемы изменили лишь комментарии задач и ни один из них не указал явно на необходимость повторного рассмотрения задачи (в таких случаях говорим, что процедура не внесла существенных изменений в текущую ситуацию), то переход к пункту 3, иначе - к пункту 2.

2) Если в результате срабатывания приема определился ответ на задачу Z_N либо был выдан отказ на нее, то возобновляется прерванный ранее прием решения задачи Z_{N-1}, в процессе реализации которого возникла задача Z_N. При $N = 2$ в этом случае выдается ответ либо отказ на решаемую задачу и возвращение в программу интерфейса решателя; при $N = 1$ - происходит выход из логической системы. При отсутствии ответа либо отказа на задачу Z_N текущий уровень этой задачи заменяется на 0, и переход к пункту 1. Это означает, что при наличии существенных изменений, внесенных приемом, решатель повторяет цикл анализа текущей ситуации с самого начала.

3) Осуществляется последовательный просмотр всех условий и посылок F задачи Z_N, вес v которых равен текущему уровню m_N этой задачи либо равен m_N+1. Сначала просматриваются условия, затем посылки; порядок просмотра условий (посылок) - слева направо по соответствующим спискам задачи Z_N. Если $v = m_N$, то происходит однократный просмотр всех вхождений логических символов φ в F (слева направо); если же $v = m_{N+1}$, то - серия таких просмотров, в процессе которых значение текущего уровня задачи Z_N полагается последовательно равным $0, 1, \ldots, m_N$. Для каждого рассматриваемого вхождения логического символа φ в F осуществляется обращение к программе логического символа φ (исходные данные этой программы содержат полную информацию о координатах вхождения символа φ в задачу Z_N). Если процедура не внесла существенных изменений в текущую ситуацию, то переход к рассмотрению очередного вхождения логического символа, иначе - к пункту 2. Если просмотр терма F закончился безрезультатно, то вес его увеличивается на 1 (новые либо видоизмененные посылки и условия задачи получают вес 0). Если просмотр всех условий и посылок задачи Z_N закончился безрезультатно, то текущий уровень задачи Z_N увеличивается на 1. Если в результате он становится больше, чем максимальный уровень M_N, то на задачу Z_N выдается отказ, иначе - переход к пункту 1.

Использование весов посылок и условий позволяет сузить область просмотра при поиске очередного приема, исключая из нее посылки и условия, имеющие большой вес (они уже были достаточно хорошо рассмотрены ранее, и срабатывание связанного с ними приема маловероятно). В результате происходит локализация рассмотрения задачи, направляемого в первую очередь на новые либо видоизмененные посылки и условия; рассмотрение же всей задачи в целом имеет место, как правило, лишь на начальном этапе ее решения. По мере повышения текущего уровня m_N задачи Z_N в просмотр вовлекаются ранее отложенные посылки и условия F, вес v которых больше m_N; это происходит при $v = m_N + 1$ (см. пункт 3), причем предварительно осуществляется поиск приема, срабатывающего при рассмотрении F для меньших, чем m_N, значений текущего уровня. Переключение внимания при рассмотрении задачи может происходить также в результате срабатывания приемов, уменьшающих веса тех или иных условий и посылок.

4.2 Запуск решения задачи и его пошаговый просмотр

Запуск решения задачи происходит из просмотра списка задач некоторого концевого раздела задачника. Войдя в этот список и создав в нем новую задачу либо выбрав для решения одну из ранее имевшихся задач, следует прежде всего добиться, чтобы верхняя горизонтальная линия данной задачи была прорисована на экране, а верхняя линия предыдущей задачи - не была видна на экране. Альтернативный способ - выделение нужной задачи нажатием правой кнопки мыши на ее поле (чтобы задача оказалась выделена, ее верхняя горизонтальная линия опять же должна быть видна на экране), после чего можно произвольно перемещаться по списку задач и выполнять запуск выделенной задачи из любой его точки.

Если требуется получить ответ задачи без отображения процесса решения, то нажимается клавиша "о" (кир.). Если на задачу будет получен ответ, то он будет прорисован в верхней части экрана, с указанием времени решения (в секундах либо минутах), а также с указанием трудоемкости, измеряемой в числе шагов работы интерпретатора языка ЛОС (этот язык нижнего уровня, используемый для записи приемов, описывается в последующих разделах книги). Ответ и трудоемкость сохраняются в файлах задачника и впоследствии будут прорисовываться непосредственно под условием задачи (время решения не сохраняется). Для исключения их из файлов следует выделить задачу (правой кнопкой мыши) и нажать "Ctr-F4". Если ответ на задачу не получен, то происходит перерисовка ее текста с указанием под ним времени, затраченного на решение.

Если нужно отображать процесс решения по шагам, то нажимается клавиша "р" (кир.). Каждый последующий шаг обеспечивается нажатием "Enter". Если в некоторый момент нужно оборвать пошаговое отображение решения и далее решать задачу до получения ответа, то нажимается клавиша "0"(ноль). Если нужно вообще прервать решени, то нажимается клавиша "Esc", возвращающая к тексту задачи в задачнике.

При показе очередного шага решения в верхней части экрана прорисовывается описание текущего действия. Над этим описанием отображается вся цепь задач (кроме фиктивной исходной задачи)- сначала идет выбранная из задачника задача (возможно, измененная в процессе решения по сравнению с ее первоначальной версией, оставшейся в задачнике), затем вспомогательная задача, к которой произошло обращение от первой задачи, и т.д. Под последней задачей этой цепи задач и размещается описание текущего действия. При просмотре всех этих записей применяется та же прокрутка, что и при просмотре списка задач в задачнике.

Отображаемые на экране задачи приводятся с теми же сокращениями, что и в задачнике (для включения либо выключения полного просмотра нажимается клавиша "ы"). Кроме того, в просматриваемой цепи задач могут встречаться "замаскированные"под задачи обращения к вспомогательным процедурам (пакетным операторам, см. описание языка ГЕНОЛОГ), не являющиеся задачами. Такие вспомогательные операторы введены из соображений оптимизации: каждый из них содержит в себе сравнительно небольшое число приемов, и поиск нужного приема в нем ускоряется на порядок по сравнению с поиском по всей базе приемов. Типы этих операторов аналогичны типам задач: проверочный оператор является аналогом задачи на доказательство, нормализатор - аналогом задачи на преобразование, синтезатор - аналогом задачи на описание и анализатор - аналогом задачи на исследование.

Под кадром, содержащим описание текущего действия, могут быть размещены несколько кадров со вспомогательными задачами - эти задачи решались в процессе выполнения данного действия. Можно повторно запустить решение любой из них - выделив ее либо разместив ее верхнюю линию в верхней части экрана и нажав "Enter". Эту процедуру можно применять любое число раз и внутри пошагового просмотра решения вспомогательных задач - таким образом создается подобие гипертекста решения. Для возвращения на предыдущий уровень данного гипертекста нажимается клавиша "End".

Комментарии к очередному действию решателя размещаются в скобках после описания этого действия. Вспомогательные задачи, сопровождающие текущее действие, также снабжаются комментарием, размещаемым в скобках перед описанием задачи. Иногда эти комментарии специально подготовлены для объяснения примененного приема, иногда они просто представляют собой подзаголовок того раздела оглавления базы приемов, в котором размещен примененный прием.

Если комментарий недостаточен для понимания выполненного действия или вообще непонятен (такое может случиться, если комментарий просто является подзаголовком концевого пункта оглавления приемов - некоторые из этих подзаголовков понятны только в контексте заголовков внешних разделов), то можно перейти к просмотру описания сработавшего приема. Конечно, здесь понадобится знакомство с языками, на которых задаются приемы решателя - ЛОСом и ГЕНОЛО́Гом (им посвящены последующие разделы книги). Для этого нажимается клавиша "б", переводящая в просмотр приема (если прием реализован на языке ГЕНОЛОГ) либо (если прием реализован на языке ЛОС) переводящая в тот концевой пункт оглавления программ, который связан с последней пройденной перед реализацией приема контрольной точкой. В обоих случаях можно также просмотреть ЛОС-программу примененного приема, нажав клавишу "ф" - она переводит в отладчик ЛОСа. Для возвращения в просмотр текущего действия решателя из просмотра описания приема на ГЕНОЛОГе следует нажать клавишу "End". Для возвращения из просмотра ЛОС-программы приема следует нажать клавишу "з".

При просмотре текущего шага решения можно посмотреть исходную задачу в задачнике, не прерывая процесса решения. Для этого достаточно нажать клавишу "Home"; возвращение к просмотру текущего шага решения из задачника - по нажатии "End".

Если при пошаговом просмотре возник промежуточный результат, представляющий самостоятельный интерес (даже в случае, когда ответ на решаемую задачу так и не будет получен), то можно выделить этот результат (целую формулу или ее часть, одну или несколько) - так же, как это делалось в задачнике, перейти в задачник (через "Home") и зарегистрировать выделенную формулу в любой из задач или введя новый бланк задачи. Как и обычно, для этого следует войти в формульный редактор и нажать "Insert" - "N", где N - номер выделенного элемента среди всех выделенных элементов; $1 \leq N \leq 9$. Далее можно нажать "End"и продолжить просмотр шагов решения.

Если возникла необходимость прервать процесс пошаговой трассировки решения некоторой задачи из цепи задач и перейти к очередному шагу для ее надзадачи, то следует выделить (правой кнопкой мыши) данную надзадачу и нажать "Enter". Тогда следующий отображаемый на экране шаг будет относиться уже к выбранной надзадаче.

Если требуется прервать затянувшийся шаг решения задачи, то нажимается клавиша "Break". В результате появится текст текущего выполняемого фрагмента ЛОС-

программы, в котором текущий (еще не выполненный) оператор выделен малиновым цветом. Далее возможен анализ текущей ситуации с помощью средств отладчика ЛОСа (см. последующие разделы) либо возвращение в главное меню при нажатии клавиши Esc. Можно также просмотреть текущую цепь задач (нажатием клавиши "з") либо (до нажатия "з"либо вернувшись из просмотра цепи задач в ЛОС-программу нажатием "ф") запустить процесс пошаговой трассировки на уровне той задачи, которая при прерывании оказалась текущей. Это делается нажатием клавиш "пробел" и "Enter".

Иногда пошаговый просмотр решения оказывается неудобен из-за того, что на некотором этапе начинается решение трудоемкой вспомогательной задачи, обращение к которой не было вынесено в самостоятельный шаг. Разумеется, по завершении этого решения и отображении срабатывания приема, в рамках которого оно происходило, решение вспомогательной задачи можно повторить для ознакомления с подробностями. Однако, длительная пауза из-за неотображаемого процесса решения может оказаться нежелательной. В таких случаях можно повторно запустить пошаговый просмотр решения, изменив его режим непосредственно перед появлением паузы. Для выбора нужного режима трассировки следует нажать клавишу "т" (это делается либо до запуска решения, из задачника, либо уже в начатом процессе трассировки). После нажатия появляется диалог установки режима.

Для ввода либо отмены условия на трассировку, определяемого в одном из пунктов диалога, следует переместить курсор мыши внутрь прямоугольника этого пункта и нажать левую клавишу мыши (наличие плюса справа от прямоугольника означает, что условие включено, иначе - выключено). После выбора необходимой комбинации условий, курсор мыши перемещается в прямоугольник "Ввести"и снова нажимается левая клавиша мыши.

Для преодоления указанных выше пауз можно воспользоваться пунктом "ручной выбор входа в подпроцесс". Если этот пункт активирован, то каждая попытка обращения решателя к вспомогательной задаче (включая наиболее крупные пакетные операторы) будет вводиться на экран. Чтобы продолжить решение без входа в пошаговую трассировку этой вспомогательной задачи, нажимается "Enter" ;чтобы войти в нее, нажимается "Ctr-Enter". Заметим, что в этом режиме текущее действие решателя не сопровождается выдачей списка вспомогательных задач, решенных для его выполнения - сообщения об обращениях к этим задачам уже выдавались на экран до соуществления действия, и была возможность просмотреть их решение по шагам. Недостатком режима с ручным входом в подпроцессы является чрезмерное количество выдаваемых на экран сообщений о попытках решения вспомогательных задач, бóльшая часть которых оказывается неудачной. Действительно ценные шаги тонут в этом потоке сообщений. Поэтому данный режим является лишь техническим, в обычных ситуациях отключенным.

Отладочные режимы трассировки описываются в разделе, посвященном отладчику ЛОСа. С помощью этих режимов можно, в частности, обеспечить прерывание процесса решения при попытке применения заданного приема, что часто используется при оптимизации его решающих правил.

Возможен серийный запуск решения задач из задачника. Такой запуск бывает необходим для контроля за изменениями в поведении решателя при его обучении. Простейшая форма этого запуска - выбор в оглавлении задачника нужного подраздела и нажатие клавиши "Ctr-з" либо клавиши "Ctr-э". В первом случае из цикла решения будут исключены все задачи, которые при предыдущем запуске решателем не были решены; во втором случае будут решаться все задачи подряд.

При серийном решении последовательно решаются сначала все задачи из текущего пункта просматриваемого меню (выделенного в момент запуска) оглавления задачника, затем - все задачи из следующего пункта этого меню, и т.д. до конца подраздела. На экране при этом последовательно сменяются формулировки решаемых задач. Если решатель слишком долго решает какую-либо задачу (возможно, "залипает"на ней из-за плохих решающих правил), то последовательное нажатие клавиш "Break" и "Esc" переводит в решение следующей задачи. Если до конца решения всех задач подраздела произойдет "зависание"программы и понадобится ее перезапуск, то после перезапуска автоматически восстанавливается прерванный цикл решения задач - вплоть до достижения конца меню.

Для обрыва серийного режима следует нажать клавишу "Break" (на экране появится фрагмент ЛОС-программы, в котором произошло прерывание, и установится пошаговый режим отладочной трассировки), и далее - нажать клавишу "Ctr-з". Лишь после этого нажатие клавиши "Esc"выведет в главное меню.

По окончании серийного запуска обновляется статистика о результатах решения задач подраздела. Такая статистика позволяет находить "особые точки"в задачнике (например, выявлять задачи, которые перестали решаться или решение которых замедлилось после внесенных в приемы изменений). Для просмотра статистики следует войти в меню нужного подраздела и нажать клавишу "з". После небольшой паузы, необходимой для просмотра всех задач подраздела, на экране возникает таблица, в которой указываются следующие сведения:

а) Пять наибольших величин замедления в решении задач по сравнению с предыдущим запуском. Эти величины измеряются в числе шагов интерпретатора ЛОСа, как и величины трудоемкости решения задач, сохраняемые в задачнике - отсюда легко извлекается процент замедления. Если нужно просмотреть имеющие самые большие значения замедления задачи, то нажимается клавиша "з", переводящая в просмотр первой из этих задач. Переход к очередной задаче (отобранные для просмотра задачи упорядочены по убыванию замедлений) осуществляется нажатием клавиши "ш". Для просмотра в таком режиме отбираются не более 40 задач подраздела (это же ограничение распространяется на другие приводимые ниже случаи отбора серий задач).

б) Число нерешенных задач раздела. Для просмотра этих задач нажимается клавиша "н". Чтобы перейти к просмотру очередной нерешенной задачи, следует нажать "ш".

в) Число утерянных решений задач подраздела (только по сравнению с предпоследним циклом решения). Для просмотра серии таких задач сначала нажимается "у", и далее - через нажатия "ш".

г) Пять наибольших величин ускорения в решении задач. Просмотр задач с ускорившимся решением - через нажатие клавиши "с",и далее к каждой следующей задаче - через нажатие "ш".

д) Суммарное замедление в решении задач подраздела (отрицательная его величина означает суммарное ускорение).

е) Число изменившихся по сравнению с предпоследним циклом решения ответов на задачи подраздела. Для просмотра серии соответствующих задач сначала нажимается клавиша "и", и далее - через нажатия "ш".

ж) Число сомнительных ответов. Для просмотра серии соответствующих задач - нажатие "о", и далее - через "ш".

з) Число задач, замедлившихся более чем на 10000 шагов интерпретатора ЛОСа, и число задач, замедлившихся более чем на 100000.

и) Можно просмотреть список всех ответов на задачи подраздела. Для этого нажимается клавиша "О". Если на выделенном ответе нажать клавишу "курсор вправо", то происходит переход к просмотру условия соответствующей задачи.

к) Пять наибольших величин трудоемкости решения задач подраздела (в числе шагов интерпретатора). Для просмотра задач в порядке убывания трудоемкости (не более 40 задач) - нажатие "т", и далее - через "ш".

Серийный запуск позволяет косвенно оценивать "холостой ход" приема - суммарную трудоемкость попыток его применения, не приводивших к срабатыванию приема. Фактически здесь оценивается относительная трудоемкость попыток применения приема на одно его срабатывание. Чтобы получить такую статистику, требуется запустить серийное решение не через "Ctr-з", а через "Ctr-х"(кир.). Тогда в указанной выше таблице статистики будут отображены данные о пяти наихудших случаях такой относительной трудоемкости ("холостого хода"). Эти данные суть пары (суммарная трудоемкость попыток применить прием - число срабатываний приема), для которых отношение первого элемента ко второму (если второй равен 0, то вместо него берется 1) наибольшее. Возможен просмотр серии наихудших приемов (по убыванию указанной характеристики) - через нажатие клавиши "х"(кир.). Переход к каждому очередному приему - через "ш". При просмотре приема повторный вызов на экран указанной пары чисел - через "Х"(кир.); пара (A_1, A_2) прорисовывается в виде двух термов - "пассив(A_1)"и "актив(A_2)".

4.3 Диалоговые задачи

Для большего единообразия интерфейса системы, различные процедуры диалогового характера (построение графиков и моделей, игровые процессы и т.п.) реализуются в виде фиктивных задач, причем запуск этих процедур происходит так же, как запуск настоящих задач - путем нажатия клавиши "о". Перечисленные выше режимы трассировки при этом обычно не используются. Ниже перечисляются интерфейсы диалога для задач таких типов.

4.3.1 Построение графиков

Задача на построение графика оформляется как задача на преобразование, условием которой служит то выражение, для которого нужно построить график. Целевая установка ее вводится, как указано выше, через оглавление типов целевых установок. После нажатия клавиши "о"под условием возникает информационная полоса, в которой содержатся либо тексты "переменная", "от", "до", $x_1, \ldots, x_n$ (если выражение имеет более одной переменной и $x_1, \ldots, x_n$ - все его переменные), либо тексты "переменная x", "от", "до", x - если выражение имеет всего одну переменную. Эти тексты представляют собой заготовки указателей варьируемой переменной, нижней и верхней границ диапазона ее изменения, а также список переменных - для различных операций, требующих выбора переменной.

Если переменных более одной, то под информационной полосой прорисовывается текст "переменная:", после которого размещается курсор формульного редактора. Этим редактором требуется ввести варьируемую переменную и нажать "Enter" (в случае единственной переменной выбор варьируемой переменной производится автоматически, и в информационной полосе она уже указана). Далее под полосой появляется сначала текст "от:", после которого формульным редактором вводится число

- нижняя граница диапазона изменения варьируемой переменной, а затем "до:" - верхняя граница этого диапазона. Введенные значения границ диапазона переносятся в информационную полосу - после соответствующих указателей "от",
"до". Если переменных больше одной, то после указания границ диапазона под полосой последовательно прорисовываются оставшиеся переменные, для каждой из которых формульным редактором вводится ее численное значение. После ввода значения последней из переменных под информационной полосой прорисовывается график. Он ограничен двумя координатными шкалами - вертикальной (размещенной слева) и горизонтальной (размещенной снизу). Масштабы на этих шкалах изначально выбираются автоматически (обычно это делается так, чтобы наибольшее и наименьшее значения на рассматриваемом диапазоне занимали крайние верхнюю и нижнюю точки области экрана, отведенной для графика). После того, как график прорисован, с ним можно выполнять следующие операции:

1) Клавишами "курсор вправо - курсор влево"можно перемещать промежуток значений варьируемой переменной с сохранением масштаба. Клавишами "курсор вверх - курсор вниз"можно сдвигать график вверх либо вниз, если интересующие участки его оказались вне экрана (как и горизонтальное перемещение, это происходит с сохранением масштаба).

2) Если нужно изменить масштаб по вертикали, то прежде всего можно использовать клавиши "в"и "н". После нажатия первой из них под информационной полосой появляется окно диалога "Верхняя граница значений:", в котором формульным редактором вводится наибольшее прорисовываемое на экране значение. Аналогично, клавиша "н"активизирует окно "Нижняя граница значений:".Чтобы восстановить исходный (автоматически выбранный масштаб), нажимается клавиша "а".

3) Чтобы изменить наибольшее либо наименьшее из отображаемых на экране значений, можно также воспользоваться мышью. Ее курсор подводится к верхней либо нижней части экрана слева от вертикальной координатной оси. Нажатие левой кнопки приведет к уменьшению диапазона просматриваемых значений на 20 процентов с выбранной (верхней либо нижней) стороны, нажатие правой кнопки - к его увеличению на 20 процентов.

4) Для выбора новой переменной, по которой строится график, нужно подвести курсор мыши к надписи "переменная"в информационной полосе и нажать левую кнопку. Далее формульным редактором вводится переменная.

5) Для выбора диапазона значений переменной нужно подвести курсор мыши либо к надписи "от", либо к надписи "до", нажать левую кнопку и далее ввести формульным редактором начало либо конец диапазона значений переменной. Можно увеличивать цифры для границ диапазона непосредственно мышью. Для этого нужно подвести курсор мыши к нужной цифре и нажать либо левую клавишу (уменьшение данного разряда на 1), либо правую клавишу (увеличение его на 1).

6) Для указания значения параметра выражения, отличного от варьируемой переменной, следует подвести курсор мыши к обозначению этого параметра в информационной полосе и нажать левую кнопку. Далее формульным редактором вводится значение параметра. Как и границы диапазона, значение параметра можно менять непосредственно мышью.

7) Можно выделить для просмотра любой прямоугольник внутри отображаемой области графика. Для этого следует активизировать пункт меню "Рамка", и далее сначала выбрать нажатием левой клавиши мыши левый верхний угол прямоугольника, а затем нажатием правой клавиши мыши - правый нижний его угол. Для выделения границ прямоугольника прорисовываются синие линии. Повторными нажатиями

указанных клавиш мыши можно корректировать их положение. Далее следует снова активизировать пункт "Рамка область внутри выбранного прямоугольника отобразится на всем экране. Для отмены просмотра прямоугольника (во время выбора его границ или после выхода в его просмотр) активизируется пункт меню "Отмена"(либо нажимается клавиша пробела). Если последовательно выбиралось несколько подпрямоугольников, то каждое нажатие "Отмена"возвращает к предыдущей картинке в цепочке.

8) Для уточнения координат точки на графике можно перейти в режим, в котором от выбранной точки к вертикальной и горизонтальной шкалам ведут отрезки прямых, выделенные голубым цветом. Одновременно в углу графика прорисовываются значения абсциссы и ординаты точки. Для перехода в данный режим следует сначала нажать левую кнопку мыши на пункте меню "Точка", после чего нажать левую кнопку мыши на выбранной для просмотра точке графика. После того, как режим просмотра абсциссы и ординаты текущйе точки уже установлен, можно менять положение этой точки - либо используя клавиши "курсор вправо" - "курсор влево"(для сдвига на один пиксел), либо нажимая левую кнопку мыши в нужной точке. Для выхода из режима повторно нажимается левая кнопка мыши на пункте меню "Точка".

9) Выход из работы с графиком - "Esc".

4.4 Упражнения по вводу и решению задач

Ввести перечисленные ниже задачи в соответствующие разделы задачника и просмотреть процесс их решения.

4.4.1 Элементарная алгебра

1. Упростить выражение:

$$\sqrt{\frac{4}{x}+\frac{1}{4x^{-1}}-2}+\sqrt{\frac{1}{4x^{-1}}+\frac{2^{-2}}{x}+\frac{1}{2}}$$

2. Решить уравнение:

$$\frac{x^2}{x+2}+1=\frac{4}{x+2}$$

3. Решить неравенство:

$$\frac{\sqrt{x^2+x-6}+3x+13}{x+5}>1$$

4. Решить систему уравнений:

$$\begin{cases}(x+y)(x^2-y^2) &= 16\\ (x-y)(x^2+y^2) &= 40.\end{cases}$$

5. Решить уравнение:

$$(tgx)^{\cos^2 x}=(ctgx)^{\sin x}$$

6. Решить неравенство:

$$|x-2|^{\log_4(x+2)-\log_2 x} < 1$$

7. Для всех a решить неравенство $ax^2+(a+1)x+1>0$.

8. При каких a неравенство $\sin^6 x+\cos^6 x+a\sin x\cos x \geq 0$ выполнено для всех значений x ?

9. Найти все a, при которых из неравенства $x^2-a(1+a^2)x+a^4<0$ следует неравенство $x^2+4x+3>0$.

10. При каких a система

$$\begin{cases} x^2+y^2 &= 2(1+a) \\ (x+y)^2 &= 14 \end{cases}$$

имеет ровно два решения ?

Указания

1. Находясь в главном меню, выбрать пункт "Оглавление задачника" (клавиша "з" либо левая кнопка мыши в прямоугольнике указанного пункта). Используя клавиши курсора, найти в корневом меню оглавления раздел "Элементарная алгебра" (возможно, сначала понадобится несколько раз нажать "курсор влево"). Войти в этот раздел, далее войти в подраздел "Упрощение выражений", затем - в любой из подразделов "Упрощение иррациональных выражений - 1,2,3". В действительности выбор раздела несущественен; решатель будет работать вне зависимости от того, в каком разделе находится задача, и классификация задач по разделам нужна лишь для упрощения поиска нужной задачи вручную. Используя "PageDown" либо "курсор вниз", вывести на экран последний пункт выбранного концевого раздела (все его пункты - номера с тремя штрихами). Затем нажать клавишу "к" (кир.) для создания бланка новой задачи. Далее нажимается "ц" и в оглавлении типов целевых установок выбираются разделы "Преобразовать выражение" - "Упростить выражение в области допустимых значений", причем после выбора последнего пункта нажимается "курсор вправо". Экран расчищается, и в верхней его части возникает текст "Упростить в о.д.з. выражение:". Далее нажимается "Enter", и формульным редактором вводится упрощаемое выражение. В процессе набора можно получать справки о клавиатуре формульного редактора, нажимая F1. Точнее, нажатие F1 переводит в общее оглавление справочника по системе, и из его корневого меню нужно перейти в раздел "Формульный редактор". Далее полезно прочитать раздел "общие сведения"; для получения информации о вводе конкретных символов - переходить в соответствующие подразделы. Чтобы вернуться в набор формулы, достаточно нажать "End". По завершении набора упрощаемого выражения нажимается "Enter". В данном примере задача оказывается полностью введенной, и для решения ее без пошаговой трассировки нажимается "о" (кир.), а для входа в пошаговую трассировку (она отображает лишь верхний уровень процесса решения - срабатывания приемов; такую трассировку, в отличие от отладочной трассировки, называем далее семантической) нажимается "р". Каждый очередной шаг трассировки - нажатие "Enter".

Если в некоторый момент под кадром, поясняющим текущее действие, оказываются расположены другие кадры, то эти последние суть кадры обращений к вспомогательным задачам, решавшимся для выполнения текущего действия. Можно выбрать любой из них для детального просмотра решения вспомогательной задачи. При этом верхняя отделяющая линия кадра должна быть в точности верхней границей экрана; такое выравнивание обеспечивается клавишами "Ctr-курсор вверх либо вниз". Вход в решение вспомогательной задачи - нажатие "Enter". По завершении трассировки решения вспомогательной задачи нажатие "Enter"возвращает на предыдущий уровень (такое возвращение можно обеспечить в процессе трассировки нажатием "End").

2. Действия аналогичны предыдущим, но выбирается раздел "Элементарная алгебра" - "Решение уравнений" - "Рациональные уравнения 1,2". Нажимаются "к","ц", и выбирается раздел оглавления целевых установок "Найти значения неизвестных" - "Получить полное явное описание значений неизвестных". На экране появляется текст "Найти", под которым размещен курсор формульного редактора. Этим редактором вводятся неизвестные задачи (отделенные запятой) - в данном примере единственная переменная x. После нажатия "Enter"ввод целевой установки завершается. Для ввода уравнения снова нажимается "Enter", и далее происходит набор уравнения.

3. Действия аналогичны предыдущим, но выбирается подраздел для неравенств.

4. Для вводы системы уравнений и (или) неравенств сначала вводится целевая установка (аналогично предыдущему, но число неизвестных может быть более одной). Затем по отдельности вводятся условия - уравнения и неравенства; ввод каждого нового условия начинается с "Enter" и завершается "Enter". После ввода последнего условия задача готова к запуску решателя.

5. Аналогично задаче 2; тригонометрические операции вводятся последовательным набором двух (в особых случаях трех) первых латинских букв обозначения этих операций: синус - s,i; косинус - c,o; тагенс - t,g; котангенс - c,t, и т.д. Заметим, что степень тригонометрической операции набирается нестандартным образом: сначала набирается вся операция, затем - курсор вверх, затем - показатель степени и "Enter". Тригонометрическая операция относится к наименьшему расположенному после нее осмысленному выражению (суммы и произведения под такой операцией следует заключать в скобки), и степень оказывается относящейся не к аргументу операции, а ко всей операции.

6. Аналогично задаче 3; вертикальные отрезки модуля вводятся с помощью клавиш "Ctr-м" (лат.), для ввода логарифма последовательно вводятся буквы l,o, после чего набирается основание логарифма. После набора основания нажимается "Enter", и набирается выражение под логарифмом (суммы и произведения следует заключать в скобки).

7. Аналогично предыдущей задаче; параметр a не включается в число неизвестных задачи, и никак более не выделяется.

8. Неизвестной задачи служит переменная a. Условие ее набирается в виде

$$\forall_x(x - \text{число} \to \sin x^6 + \cos x^6 + a \sin x \cos x \geq 0).$$

Для ввода "x – число" сначала вводится x, затем нажимается /, затем "ч".

9. Аналогично предыдущему; условие набирается в виде:

$$\forall_x(x^2 - a(1+a^2)x + a^4 < 0 \rightarrow x^2 + 4x + 3 > 0.$$

Заметим, что понятие "следует" здесь трактуется таким образом, что если левая от стрелки часть ложная (в частности, неравенство слева вообще не имеет решений), то вся импликация истинна. Это приводит к тому, что концевые точки указанного в ответе промежутка (для них левое неравенство не имеет решений) отнесены к ответу.

10. Условие задачи состоит в том, что множество пар (x, y), удовлетворяющих уравнениям, имеет мощность 2. Оно записывается в виде

$$card(set_{xy}(x^2 + y^2 = 2(1+a)\ \&\ (x+y)^2 = 14)) = 2.$$

4.4.2 Планиметрия

1. Основание равнобедренного треугольника равно a, угол при вершине равен b. Найдите биссектрису, проведенную к боковой стороне.

2. В трапеции $ABCD$ с основаниями AD и BC имеем $AD = 3$, $BC = 1$. Точка P лежит на стороне AB, а точка Q - на стороне CD, причем отрезок PQ параллелен основаниям и проходит через точку пересечения диагоналей трапеции. Найти длину отрезка PQ.

3. Известно, что $ABCD$ - ромб и радиусы окружностей, описанных около треугольников ABC и ABD соответственно, равны R и r. Найти площадь ромба $ABCD$.

4. В параллелограмме $ABCD$ биссектриса угла BAD пересекает сторону CD в точке M, причем $DM/MC = 2$. Известно, что угол CAM равен a. Найти угол BAD.

5. Окружность проходит через вершины A и C треугольника ABC, пересекает сторону AB в точке D и сторону BC в точке E. Известно, что $AD = 5, AC = 2\sqrt{7}, BE = 4, BD/CE = 3/2$. Найти угол CDB.

Указания

1. Вычислительные задачи по геометрии вводятся в следующем порядке: сначала набираются посылки задачи, перечисляющие известные свойства чертежа; затем вводится целевая установка задачи; затем указываются неизвестные величины, которые должны быть вычислены. Перед набором посылок можно ввести чертеж, однако чертеж может быть создан системой и автоматически (по окончании набора задачи нажимается "Ctr-ч").

   При наборе посылок следует обязательно обозначить все точки, участвующие в задаче, буквами (обычно большими; можно использовать натуральные индексы). Ссылаться на плоские фигуры (треугольники, многоугольники, окружности, прямые, и т.д.) можно только через их "базисные" точки. В нашем случае обозначим вершины треугольника буквами A, B, C и введем первую посылку "$\triangle(ABC)$". Для ввода посылки нажимается "Enter", затем последовательно

нажимаются клавиши "т","р"(обычно используются первые две буквы вводимого понятия). Это приводит к прорисовке символа $\triangle$ с расположенной после него открывающей скобкой. Далее последовательно нажимаются A,B,C и закрывающая скобка, после чего - завершающее набор формулы "Enter". Никакие разделители между буквами не ставятся. Заметим, что если в обозначении треугольника либо другой фигуры используется буква с индексом, то после такой буквы, перед набором следующей, обязательно нужно нажать на клавишу "умножение" (звездочка).

После ввода первой посылки (она указывает только на то, что три точки A, B, C образуют вершины некоторого треугольника, то есть не лежат на одной прямой) нужно ввести посылку, определяющую, что треугольник равнобедренный. Выберем в качестве основания треугольника вершины A, C и введем посылку $l(AB) = l(BC)$. Для обозначения расстояния дважды нажимается клавиша "l", что приводит к появлению рукописной латинской буквы l с идущей после нее открывающей скобкой. Затем (как в обозначении треугольника) подряд вводятся буквы для точек, между которыми рассматривается расстояние, и ставится закрывающая скобка.

Далее вводятся: посылка $l(AC) = a$, обозначающая длину основания треугольника через a, и посылка $\angle(ABC) = b$ - для величины угла при основании. Чтобы получить обозначение угла, последовательно нажимаются клавиши "у","г". Далее - как для обозначения треугольника. Как и обычно, вершина угла размещается в обозначении угла посередине.

Для ввода основания D биссектрисы треугольника ABC, проведенной из угла BAC, можно использовать посылку "Биссектреуг($BACD$)". Другой способ (более громоздкий, но не использующий специального обозначения "Биссектреуг") - пара посылок "$D \in$ прямая(BC)", "биссектриса($BACD$)". В первом случае нажимаем "И","Т", и далее - как в случае обозначения треугольника, но для четырех букв. Во втором случае сначала вводим D, затем нажимаем пробел, b, e (лат.; появляется символ $\in$). Далее нажимаем "п","р" (кир.; появляется слово "прямая" с открывающей скобкой), вводим буквы B, C и закрывающую скобку. Для ввода "биссектриса(...)" нажимаем клавиши "и","т".

На этом ввод посылок завершен. Для ввода целевой установки нажимаем клавишу "ц", переводящую в оглавление типов целевых установок. Из корневого меню этого оглавления переходим к пункту "Найти значения неизвестных" - "Выразить значения неизвестных через заданные параметры". Заметим, что применявшийся в элементарной алгебре пункт "Получить полное явное описание значений неизвестных" в планиметрических задачах на вычисление НЕ следует использовать. Это объясняется принципиальным различием понятий "известная" и "неизвестная" в задачах из двух этих разделов. В элементарной алгебре все переменные из списка посылок по умолчанию считались известными и могли входить в ответ задачи. В геометрической задаче на вычисление посылки содержат обозначения точек $A, B, C, \ldots$, которые не должны появляться в ответе. Соответственно различаются и типы выбираемых целевых установок. После выбора указанного типа целевой установки нажимается "курсор вправо". Далее сначала вводится буква (или несколько отделенных запятыми букв) для неизвестной (неизвестных) и нажимается "Enter". Затем вводятся разделенные запятыми буквы для известных числовых параметров, через которые должны быть выражены неизвестные (если таких параметров вообще нет, сразу

нажимается "Enter", иначе оно нажимается после ввода параметров). В нашем примере обозначим неизвестную длину биссектрисы через x; параметры суть a, b.

После ввода целевой установки вводим равенства, связывающие неизвестные (переменные) с теми выражениями, которые они обозначают и которые нужно вычислить. В ответ войдет сама неизвестная, а не обозначаемое ею выражение.

Как уже говорилось выше, после ввода задачи по планиметрии можно ввести чертеж - либо попробовать сделать это автоматически (нажатием "Ctr-ч"), либо ввести чертеж вручную (нажать "ч" и далее воспользоваться геометрическим редактором; чертеж появится перед списком посылок, в начале задачи). Можно также скорректировать вручную чертеж, созданный автоматически (вход в редактирование уже созданного чертежа - снова через "ч"; удаление чертежа - выделить его левой кнопкой мыши и нажать "Ctr-Del").

2. Напомним, что в решателе предусмотрены два варианта обозначения трапеции: "трапеция($ABCD$)" и "Трапеция($ABCD$)" - первый из них для случая трапеции с большим основанием AD и острыми углами при основании; второй - для случая, когда углы при основании не обязательно острые. В обоих случаях указанные записи предствляют собой даже не обозначения трапеции, а лишь утверждения о том, что точки A, B, C, D являются вершинами соответствующей трапеции. Сама трапеция (как и любой другой четырехугольник) обозначается посредством выражения "фигура($ABCD$)". В нашем примере годится любой из указанных способов записи. Например, введем посылку "трапеция($ABCD$)". Затем вводятся посылки $l(AD) = 3, l(BC) = 1$. Принадлежность точки P стороне AB, а точки Q - стороне CD, записываются в виде $P \in$ отрезок(AB), $Q \in$ отрезок(CD). Параллельность отрезка PQ основаниям трапеции записывается как прямая(PQ)$\|$прямая(AD). Чтобы сформулировать условие о том, что отрезок PQ проходит через точку пересечения диагоналей трапеции, нужно ввести обозначение для этой точки. Например, обозначим ее через M. То, что M является точкой пересечения диагоналей, записываем как $M \in$ прямая(AC), $M \in$ прямая(BD). Далее добавляем посылку $M \in$ отрезок(PQ). На этом ввод посылок завершается. Как и в предыдущей задаче, выбираем целевую установку и вводим единственное условие $x = l(PQ)$ для неизвестной x.

3. Начинаем со ввода посылки "ромб($ABCD$)". Так как в задаче речь идет об описанных окружностях, надо ввести обозначения для этих окружностей, то есть обозначить центр окружности и выбрать какую-либо точку на окружности (в планиметрии ссылки на окружности могут быть только такими). Например, обозначим центры через M, N. В случае описанной окружности, которая должна проходить через вершины треугольника, в качестве второй точки можно взять какую-либо вершину треугольника (например, A). Тогда добавятся посылки "окружность(MA) описана около фигура(ABC)"; "окружность(NA) описана около фигура(ABD)". Заметим, что хотя эти тексты кажутся достаточно длинными, набираются они весьма малым числом нажатий клавиш: сначала нажимаем "о","к" (кир.) - появляется слово "окружность" с открывающей скобкой. Затем вводим буквы M, A и закрывающую скобку. Далее нажимаем "Ctr-ф" - появляются слова "описана около". Наконец, нажимаем "ф","и" - появляется слово "фигура"с открывающей скобкой. В заключение вводим A,B, C и закрывающую скобку. Радиусы окружностей указываем с помощью посылок

$l(MA) = R, l(NA) = r$. При вводе целевой установки указываем, что значение неизвестной x должно быть выражено через R, r. Наконец, набираем условие $x = S(\text{фигура}(ABCD))$. Заметим, что знак площади S здесь вводится двукратным нажатием малой латинской буквы s; если его ввести нажатием клавиши большой латинской S, то задача окажется набранной неверно и не будет решена (вместо площади окажется введенным значение какой-то неопределенной функции S).

4. Вводятся посылки "параллелограмм$(ABCD)$", "биссектриса$(BADM)$", $M \in$ отрезок(CD), $l(DM)/l(MC) = 2$, $\angle(CAM) = a$. Затем выбирается целевая установка "Выразить x через a", и вводится условие $x = \angle(BAD)$.

5. Вводятся посылки $\triangle(ABC)$, $C \in$ окружность(PA), $D \in$ окружность(PA), $D \in$ отрезок(AB), $E \in$ окружность(PA), $E \in$ отрезок(BC), $l(AD) = 5, l(AC) = 2\sqrt{7}, l(BE) = 4, l(BD)/l(CE) = 3/2$. Затем вводятся целевая установка и условие $x = \angle(CDB)$.

4.4.3 Математический анализ

1. Вычислить производную функции

$$\frac{e^{-x^2}\arcsin(e^{-x^2})}{\sqrt{1-e^{-2x^2}}} + \frac{1}{2}\ln(1-e^{-2x^2})$$

2. Вычислить предел функции

$$(2e^{\frac{x}{x+1}} - 1)^{\frac{x^2+1}{x}}$$

при $x \to 0$.

3. Исследовать поведение функции $y = (x-5)\sqrt[3]{x^2}$.

4. Найти точки экстремума для функции $y = x^2 - ln(x^2)$.

5. Исследовать на непрерывность функцию $y = \text{arctg}(1/x + 1/x - 1 + 1/x - 2)$.

6. Найти неопределенный интеграл

$$\int \frac{2\sin x - \cos x + 3}{3\sin x + \cos x + 1} dx$$

7. Вычислить определенный интеграл

$$\int_0^{ln2} \sqrt{e^x - 1} dx$$

8. Вычислить двойной интеграл от $\sqrt{|x-y^2|}$ по области $(x,y) : |y| \le 1, 0 \le x \le 2$.

9. Найти площадь области, заключенной между параболой $y^2 = \frac{b^2}{a}x$ и прямой $y = \frac{b}{a}x$; $0 < a, 0 < b$.

10. Найти объем тела $x^2 \le ay \le bx, x^2 + y^2 \le hz \le 2x^2 + 2y^2$; $0 < a, 0 < b, 0 < h$.

11. Найти площадь поверхности $z = xy, x^2 + y^2 \leq R^2$.

12. При каких значениях параметра p сходится ряд

$$\sum_{i=1}^{\infty} \frac{1}{i^p} \sin \frac{\pi}{i}?$$

13. Разложить в ряд Тейлора по переменной x в точке 0 функцию

$$y = (x - \operatorname{tg} x) \cos x$$

14. Разложить в ряд Фурье на отрезке $[-\pi, \pi]$ функцию $y = (\pi^2 - x^2)^2$.

15. Найти сумму ряда

$$\sum_{n=1}^{\infty} \frac{3n^2 + 3n + 1}{n^3(n+1)^3}.$$

Указания

1. Прежде всего нужно войти в оглавление целевых установок и выбрать пункт "Упростить выражение в области допустимых знчений". Затем набирается производная указанного выражения. Она вводится как дробь вида $\frac{dA}{dx}$; A - дифференцируемое выражение. После символа d нужно ставить знак умножения; дифференцируемое выражение заключается в скобки. В принципе, переменную d можно использовать и внутри выражения A, и даже взять ее в качестве x. Если нужно найти значение производной в некоторой точке t, то вместо dx набирается $d(x = t)$, причем и здесь после d ставится знак умножения.

2. Снова вводится установка "Упростить выражение в области допустимых значений". Затем набирается условие: сначала нажимаются клавиши l, i - появляется символ lim, справа внизу от которого размещен курсор. Набираем $x \to 0$ (стрелка вводится клавишей "курсор вправо") и нажимаем Enter - курсор перемещается вверх в ту же строку, в которой расположен символ lim. Здесь набираем выражение под знаком предела. Это выражение обязательно нужно заключить в скобки, иначе знак предела будет отнесен к минимальному осмысленному его началу. В нашем примере, если не заключить все выражение в скобки, то степень окажется вне предела и ответ будет другим.

3. Выбирается целевая установка "Исследовать поведение функции", в которой указывается обозначение функции - переменная y. Затем вводится условие $y = \lambda_x((x-5)\sqrt[3]{x^2}, x - \text{число})$.

4. Выбираем переменные, которые будут обозначать, соответственно, точку экстремума, значение функции в этой точке, и тип экстремума (максимум либо минимум). Например, пусть это будут переменные u, v, w. Вводим целевую установку "Найти полное явное описание значений неизвестных" в которой указываем выбранные неизвестные. Затем вводим условие

$$Extr(\lambda_x(x^2 - \ln(x^2), x - \text{число}), u, v, w).$$

Аналогично вводятся задачи на поиск множества точек минимума либо максимума некоторой функции внутри заданной области, но число неизвестных

здесь будет равно двум (искомое множество точек и значение функции в этих точках). Вместо $Extr$ используются символы Min, Max, причем после функции должна быть указана область, по которой берется минимум или максимум.

5. Отличие от общего исследования поведения функции - только в том, что выбирается целевая установка "Исследовать функцию на непрерывность".

6. Выбирается целевая установка "Упростить выражение в области допустимых значений". Затем набирается неопределенный интеграл: нажимается "Ctr-j"и вводится подынтегральное выражение, которое умножается справа на произведение dx.

7. Аналогично неопределенному интегралу, но нажимается "Ctr-i". Тогда курсор сначала оказывается под знаком интеграла, где набирается нижний предел. После нажатия "Enter" курсор переводится вверх, где набирается верхний предел. Еще одно нажатие "Enter" - и набирается подынтегральное выражение, умножаемое на dx.

8. При вычислении двойных интегралов область интегрирования обычно задается в списке посылок. Выбираем для нее обозначение - например, переменную P, и введем посылку $P = set_{xy}(|y| \le 1 \ \& \ 0 \le x \ \& \ x \le 2)$. Затем вводим целевую установку "Упростить выражение в области допустимых значений". Затем набираем двойной интеграл - нажимаем "Ctr-2"; под интегралом вводим обозначение области P, нажимаем "Enter", и далее набираем подынтегральное выражение, умноженное на произведение $dxdy$.

9. Плоскую область определяем в списке посылок, обозначив ее вспомогательной переменной (например, P). Эту область задаем, используя ссылку на прямоугольную систему координат, которую тоже обозначаем вспомогательной переменной (например, K). После этого в нашем примере вводим посылки

$$P = \text{точки}(\text{областьграницы}(set_{xy}(y^2 = \frac{b^2}{a}x) \cup set_{xy}(y = \frac{b}{a}x)), K),$$

$0 < a, 0 < b$, "прямкоорд(K)". Заметим, что операция "областьграницы" применяется к теоретико-множественному объединению пар координат точек кривых, ограничивающих область, а значением этой операции служит множество пар координат точек области. Для перехода от пар координат к точкам плоскости используется операция "точки". Далее выбирается целевая установка "Упростить выражение в области допустимых значений" и вводится условие $S(P)$. Как и в случае геометрических задач, символ площади S вводится двойным нажатием клавиши s.

10. В случае нахождения объемов применяется аналогичная конструкция, но множество троек координат трехмерной области задается непосредственно, с помощью неравенств. В нашем примере вводим посылки

$$P = \text{точки}(set_{xyz}(x^2 \le ay \ \& \ ay \le bx \ \& \ x^2 + y^2 \le hz \ \& \ hz \le 2x^2 + 2y^2), K),$$

$0 < a, 0 < b, 0 < h$, "прямкоорд(K)". Целевая установка - та же, что и в предыдущем примере; условие имеет вид "объем(P)" (двукратное нажатие клавиши "O", кир.).

11. Посылки вводятся аналогично предыдущей задаче, причем вместо множества троек координат точек трехмерной области задается множество троек координат точек поверхности. Условие имеет вид $S(P)$.

12. Целевая установка при наличии параметров ряда - "Получить полное явное описание значений неизвестных" (при их отсутствии - "Проверить истинность утверждения"); неизвестная - параметр ряда p. Условие имеет вид утверждения о сходимости последовательности частичных сумм ряда:

$$\text{сходится}(\lambda_n(\sum_{i=1}^{n}(\frac{1}{i^p}\sin\frac{\pi}{i}), n - \text{натуральное}))$$

13. Выбирается установка "Разложить в ряд Тейлора", в которой указываются переменная разложения и точка разложения. Условием задачи служит выражение $(x - \operatorname{tg} x)\cos x$.

14. Аналогично предыдущему - с установкой "Разложить в ряд Фурье".

15. Установка - "Упростить выражение в области допустимых значений". Условие задачи набирается непосредственно в виде бесконечной суммы.

4.4.4 Аналитическая геометрия и линейная алгебра

1. Даны три вектора $a(4,1,5)$, $b(0,5,2)$ и $c(-6,2,3)$. Найти вектор x, удовлетворяющий системе уравнений $(x,a) = 18, (x,b) = 1, (x,c) = 1$. Система координат прямоугольная.

2. Даны координаты двух вершин треугольника $A(-1,3), B(2,5)$ и точки пересечения его высот $H(1,4)$ в прямоугольной системе координат. Найти координаты третьей вершины треугольника и составить уравнения его сторон.

3. Составить уравнения плоскостей, проходящих через прямую $\frac{x-1}{3} = \frac{y-1}{5} = \frac{z+2}{4}$ и равноудаленных от точек $A(1,2,5)$ и $B(3,0,-1)$.

4. Составить уравнение касательной к параболе $y^2 = -8x$, отрезок которой между точкой касания и директрисой делится осью Oy пополам. Система координат прямоугольная.

5. Найти уравнение плоскости, пересекающей эллипсоид $x^2 + 2y^2 + 4z^2 = 9$ по эллипсу, центр которого находится в точке $C(3,2,1)$. Система координат прямоугольная.

6. Поверхность задана уравнением $6xy - 8y^2 - z^2 + 60y + 2z + 89 = 0$ в прямоугольной системе координат. Найти каноническую систему координат и каноническое уравнение этой поверхности. Определить тип поверхности.

7. Решить матричное уравнение:

$$\begin{pmatrix} 2 & -3 & 1 \\ 4 & -5 & 2 \\ 5 & -7 & 3 \end{pmatrix} \cdot X \cdot \begin{pmatrix} 9 & 7 & 6 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -2 \\ 18 & 12 & 9 \\ 23 & 15 & 11 \end{pmatrix}$$

8. Вычислить определитель

$$\begin{vmatrix} \sin a & \cos a & \sin(a+d) \\ \sin b & \cos b & \sin(b+d) \\ \sin c & \cos c & \sin(c+d) \end{vmatrix}$$

9. Найти собственные значения и собственные векторы линейного преобразования, заданного матрицей:

$$\begin{pmatrix} 3 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 3 & 0 & 5 & -3 \\ 4 & -1 & 3 & -1 \end{pmatrix}$$

10. Найти каноническую жорданову форму для матрицы:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Указания

1. В задачах по аналитической геометрии обычно приходится вводить обозначение для используемой системы координат. Обозначим ее в нашем примере через K и введем посылку "прямкоорд(K)". Напомним, что в решателе системы координат на плоскости отождествляются с тройками точек общего положения (первая точка - начало системы координат, две последние - концы координатных векторов), а в пространстве - с четверками таких точек. Если в задаче имеются ссылки на координатные оси либо плоскости, то приходится явным образом вводить такие тройки либо четверки точек. В нашем примере этих ссылок нет, поэтому далее вводим посылки, определяющие координаты векторов a, b, c относительно K: "коорд(a, K) = (4, 1, 5)"; "коорд(b, K) = (0, 5, 2)"; "коорд(c, K)
= (−6, 2, 3)". Затем вводим посылки "скалумнож(a, x) = 18", "скалумнож(b, x) = 1", "скалумнож(c, x) = 1". Выбираем целевую установку "Выразить значения неизвестных через заданные параметры" и вводим переменную y для неизвестной. Наконец, вводим условие задачи "y = коорд(x, K)".

2. Вводим обозначение K для прямоугольной системы координат и заносим посылки "прямкоорд(K)", $\triangle(ABC)$, "коорд(A, K) = (−1, 3)", "коорд(B, K) = (2, 5)". Чтобы определить H как точку пересечения высот, проводим две высоты - из вершины A и из вершины B. Это можно сделать, добавив две посылки "Высота($ABCM$)" и "Высота($BCAN$)"; M, N - основания высот. Далее добавляем посылки, связанные с точкой H: $H \in$ прямая(AM); $H \in$ прямая(BN); "коорд(H, K) = (1, 4)". Выбираем в качестве неизвестных переменные x, y, z, v - для уравнений сторон треугольника и координат его третьей вершины. Наконец, создаем целевую установку (так же, как в предыдущей задаче) и вводим условия: x = коорд(прямая(AB), K), y = коорд(прямая(AC), K), z =

коорд(прямая
$(BC), K)$,v = коорд(C, K).

3. В этой задаче система координат не предполагается прямоугольной. Поэтому выбираем обозначающую ее переменную K, но никаких специальных посылок для K не вводим. Так как в условии задачи говорится о прямой, а прямые в решателе обозначаются только с помощью пары точек, выбираем переменные P, Q для обозначения этих точек. После этого можно ввести посылку, задающую уравнение прямой: "коорд(прямая$(PQ), K) = set_{xyz}$(пропорцнаборы$((x-1, y-1, z+2), (3, 5, 4)))$". Заметим, что использование отношения "пропорцнаборы" пропорциональности двух числовых наборов при задании канонического уравнения прямой в пространстве является для решателя стандартом. Чтобы обозначить плоскость, выбираем переменные C, D, E для трех точек общего положения на этой плоскости. Затем вводим посылку, выражающую включение прямой PQ в искомую плоскость: "прямая$(PQ) \subseteq$ плоскость(CDE)". Вводим посылки, определяющие координаты указанных в задаче точек A, B: "коорд$(A, K) = (1, 2, 5)$", "коорд$(B, K) = (3, 0, -1)$". Наконец, указываем на равноудаленность плоскости CDE от точек A, B: "расстдоплоскости$(A,$ плоскость$(CDE)) =$ расстдоплоскости$(B,$ плоскость$(CDE))$". Затем вводим целевую установку (так же, как в предыдущих задачах), указывая в ней неизвестную x для координат плоскости CDE. Завершаем набор задачи условием $x =$ коорд(плоскость$(CDE), K)$.

4. Так как система координат прямоугольная, вводим посылку "прямкоорд(K)". Обозначаем параболу через E и указываем ее уравнение: "коорд$(E, K) = set_{xy}$ $(y^2 = -8x)$". Выбираем обозначение "прямая(AB)"для касательной и вводим посылку "прямая(AB) − касательная к E". Для точки касания можно было бы выбрать специальное обозначение, однако без ограничения общности можно считать, что этой точкой служит точка A. Поэтому вводим посылку $A \in E$. Для директрисы вводим обозначение "прямая(FG)". Так как в решателе упоминание о директрисе связано с обязательным упоминанием фокуса, которому она соответствует, то выбираем обзначение C для фокуса и заносим посылки "фокус$(C,$
$E)$", "директриса(прямая(FG),$C, E)$". В качестве точки пересечения касательной с директрисой можно без ограничения общности выбрать точку B; соответствующая посылка имеет вид $B \in$ прямая(FG). Для обозначения середины отрезка AB выбираем переменную D и заносим посылки $D \in$ отрезок(AB), $l(AD) = l(BD)$. Чтобы указать на принадлежность точки D оси ординат, вводим явное обозначение для тройки точек, образующих систему координат K : $K = (M, N, P)$. Теперь принадлежность точки D оси ординат записывается в виде $D \in$ прямая(MP). Чтобы предотвратить недопустимый в задаче случай совпадения прямой AB с осью ординат, добавляем посылку $\neg(A \in$ прямая$(MP))$. Далее вводим целевую установку (как в предыдущей задаче) с упоминанием неизвестной x и добавляем условие "$x =$ коорд(прямая$(AB), K)$".

5. Вводим посылки "прямкоорд(K)" и "коорд$(E, K) = set_{xyz}(x^2 + 2y^2 + 4z^2 = 9)$"; E - обозначение эллипсоида из условия задачи. Обозначаем искомую плоскость "плоскость(ABC)", а сечение ею эллипсоида - F, после чего вводим посылки $E \cap$ плоскость$(ABC) = F$ и "эллипс(F)". Обозначив центр эллипса через D,

добавляем посылки "центр(D, F)" и "коорд$(D, K) = (3, 2, 1)$". Далее вводим целевую установку с неизвестной x и условие "$x =$ коорд(плоскость$(ABC), K)$".

6. Вводим посылку "прямкоорд(K)" и выбираем целевую установку "Исследовать свойства поверхности, заданной своим уравнением". В этой установке обозначаем исследуемую поверхность через E, и вводим условие задачи: "коорд(E, K) $= set_{xyz}(6xy - 8y^2 - z^2 + 60y + 2z + 89 = 0)$".

7. Для решения матричного уравнения используем обычную целевую установку уравнений "Получить полное явное описание значений неизвестных". Для набора матрицы нажимаем клавиши "м","а" (кир.) - появляется левая скобка матрицы. Затем вводится первая строка, причем после каждого элемента нажимается "Enter". Чтобы перейти к следующей строке, после набора последнего элемента вместо "Enter" нажимается "Page Down". После набор последней строки нажимается "End" - возникает правая скобка матрицы. Для отката в случае ошибочных действий при наборе матрицы используется "Backspace". После набора первой матрицы вводится обычный символ умножения (клавиша "звездочка"), затем X, затем снова умножение, далее - вторая матрица, и после знака равенства - третья. Преобразование умножения чисел в матричное умножение предпринимается автоматически при запуске решения задачи.

8. Для вычисления определителя выбирается целевая установка "Упростить выражение в области допустимых значений". При наборе условия сначала нажимаются клавиши "d","e"(лат.) - возникает символ det. Затем водится матрица.

9. Выбираем обозначение для матрицы - переменную A. Затем вводим посылку - равенство с переменной A в левой части и матрицей в правой. Выбираем в качестве неизвестных переменные x, y, z; x - собственное значение, y - его кратность, z - собственный вектор, отвечающий данному собственному значению. Затем выбираем целевую установку "Получить полное явное описание значений неизвестных" для указанных неизвестных и вводим условия "собствзначение(A, x, y)", "собствектор(A, x, z)". В ответе для z приводится условие принадлежности множеству линейных комбинации базиса собственных векторов, соответствующих собственному значению x.

10. Выбираем целевую установку "Получить полное явное описание значений неизвестных" для неизвестных x (жорданова форма) и y (матрица, преобразующая к жордановой форме, т.е. такая, что произведение обратной к ней матрицы, исходной и снова матрицы y, равно x). Затем вводим условие "жордформа(A,x, y)", где A - исходная матрица.

4.4.5 Дифференциальные уравнения

1. Решить уравнение
$$2(y - 2xy - x^2\sqrt{y}) + x^2y' = 0$$

2. Решить уравнение
$$xyy'' + xy'^2 - yy' = 0$$

3. Найти решения уравнения $y = xy' - 2y'^2$, проходящие через точку (4,2).

4. Найти решение системы дифференциальных уравнений:

$$\begin{cases} \frac{dx}{dt} &= x - y + 4\cos(2t) \\ \frac{dy}{dt} &= 3x - 2y + 8\cos(2t) + 5\sin(2t). \end{cases}$$

Указания

1. Выбираем целевую установку "Решить функциональные уравнения" (эта установка берется и для остальных задач на решение дифференциальных уравнений). В качестве неизвестной указываем уже не переменную y, а выражение $y(x)$. Затем вводим условие задачи - уравнение, в котором везде вместо функции y записывается ее значение $y(x)$ в точке x, а вместо производной y' - $\frac{dy(x)}{dx}$. Напомним, что по всем произвольным постоянным, возникающим при интегрировании уравнения, в ответе навешивается квантор существования.

2. Задача вводится аналогично предыдущей; производная второго порядка набирается в виде $\frac{d^2y(x)}{dx^2}$. Символ дифференцирования d здесь рассматривается как переменная - то есть после него или его степени нужно нажимать клавишу "звездочка" для умножения.

3. После ввода дифференциального уравнения добавляем условие $y(4) = 2$.

4. В целевой установке указываем две неизвестных функции - $x(t), y(t)$. Уравнения системы набираются аналогично предыдущим задачам; вместо x, y в них используются записи $x(t), y(t)$.

4.5 Анализ траекторий решения задач при обучении решателя

Пополнение базы приемов решателя происходит при ручном обучении только за счет анализа траекторий решения задач. Создавать какие-либо приемы из общих соображений, без примерки на задачах, не рекомендуется. Велика вероятность того, что такой прием не будет использоваться решателем - либо из-за того, что он окажется избыточным и его срабатывание предвосхитят другие приемы, либо из-за того, что действия других приемов уведут задачу из области его применимости. Обучающий материал для решателей дают задачники, а не учебники.

Таким образом, чтобы научиться создавать решатели, прежде всего нужно овладеть техникой разбиения траектории решения задачи на последовательность применений приемов. Отсюда возникают исходные неформальные версии описаний приемов, которые затем уже записываются на ГЕНОЛОГе или на ЛОСе. Разумеется, для одной и той же задачи могут быть предложены сильно отличающиеся друг от друга способы решения, а для одного и того же решения - различные версии объясняющих его приемов. Наиболее простые и эффективные варианты не всегда удается "угадать" по единственному примеру, так что оптимизация решателя предполагает более или менее регулярные откаты для модернизации уже созданных групп приемов. Обычно эти откаты имеют локальный характер, а использование компилятора ГЕНОЛОГа делает их в техническом отношении не слишком дорогостоящими.

Рассмотрим несколько примеров анализа траекторий решения задач, в которых не будем предполагать наличия каких-либо ранее созданных приемов. Все эти задачи

взяты из задачника решателя, и в качестве упражнения можно рекомендовать проследить по шагам его действия. Приводимые ниже рассуждения дают лишь первое, весьма приблизительное представление о разбиении решений на приемы. Реальное обучение решателя требует учета гораздо большего числа технических подробностей. Лучшее представление об этом можно будет составить по второй книге монографии, при знакомстве с уже созданными приемами.

Начнем с простейшего примера - решения уравнения

$$\frac{6a+7b}{6a} - \frac{3bx}{2a^2} = 1 - \frac{bx}{a^2-ab}.$$

Первый шаг, который представляется естественным в этой задаче - группировка в левой части всех членов с неизвестной x, а в правой - всех известных членов. Этот шаг сразу подсказывает прием: если обе части числового уравнения имеют неизвестные либо известные слагаемые, то выполняется указанная выше перегруппировка членов. Уровень срабатывания приема можно взять совсем маленьким, например, равным 1. Для усиления стандартизации вводим еще один прием, переводящий неизвестную часть равенства влево, а известную - вправо. Уровень его пусть тоже будет равен 1. Каких-либо соображений об ограничении применения данных приемов не возникает. Однако, следует учитывать, что теперь на уровнях выше первого все уравнения будут иметь известные слагаемые только в правой части. Поэтому, например, шаблон для усмотрения квадратных уравнений должен иметь вид $ax^2+bx=c$, вместо привычного $ax^2+bx+c=0$.

Итак, получаем уравнение:

$$\frac{bx}{a^2-ab} - \frac{3bx}{2a^2} = 1 - \frac{6a+7b}{6a}.$$

Следующий шаг - сложить дробные слагаемые в левой части уравнения. Формулируем соответствующий прием: "если левая часть уравнения имеет дробное слагаемое, то обращаемся к вспомогательной задаче на преобразование ее к виду дроби, после чего выполняем замену". Каких-либо оснований откладывать это действие не видно, так что уровень срабатывания приема снова выбираем равным 1. Целесообразность применения данного приема уже не столь очевидна, как в предыдущем случае. Сложение дробных выражений может привести к очень громоздкому результату, и задача будет заведена в тупик. С другой стороны, сразу привести условия, отделяющие допустимые применения от недопустимых, мы не можем. Чтобы возникли какие-то подсказки на этот счет, нужно все-таки ввести прием таким, как есть, и подождать появления задачи, в которой он будет мешать. Тогда и начнется накопление серии дополнительных эвристических ограничений, которые обеспечат должное управление приемом.

В нашем случае эта работа уже проделана - можно заглянуть в решатель и посмотреть, какие ограничения возникли. Прежде всего, оказалось, что для систем уравнений уровень срабатывания данного приема лучше положить равным не 1, а 3. Если нужно не складывать неизвестные дробные выражения, а обозначить их новыми неизвестными и решить полученную вспомогательную систему, то прием, выполняющий эти действия, успеет сработать. Если уравнение содержит неизвестную, явно выраженную с помощью еще одного уравнения через другие неизвестные, то прием блокируется - лучше (вообще говоря) сначала подставить найденное значение. Блокировка происходит также при наличии в уравнении неизвестного логарифма от

суммы с дробным слагаемым. Она заставляет решатель сначала преобразовать указанный логарифм. Наконец, в случае единственной неизвестной применение приема откладывается (как и в случае систем) до уровня 3 при наличии неизвестных логарифмов по разным основаниям. Перечисленные условия относятся к сравнительно редким ситуациям, и таким образом применение рассматриваемого приема оказалось "почти всегда" оправданным.

Введенный нами прием обратился к вспомогательной задаче на сложение дробных выражений. Поэтому временно прерываем анализ цепочки преобразований основной задачи и переходим к рассмотрению выражения

$$\frac{bx}{a^2-ab}-\frac{3bx}{2a^2}.$$

План наших действий очевиден - сначала нужно разложить на множители знаменатели, а затем выполнить сложение дробей. Оформим эти действия в виде приемов.

Разложение на множители числителей и знаменателей можно считать преобразованием общей стандартизации, предшествующим применению других приемов, относящихся к дробям. Однако, на этапе завершающего редактирования упрощаемого выражения может понадобиться обратное преобразование - иногда раскрывание скобок приводит к получению более компактной записи. Момент перехода к завершающему редактированию ответа можно помечать вводом специального комментария задачи, и тогда первый прием (разложение на множители) будет срабатывать при отсутствии данного комментария, а второй (попытка упрощающего раскрывания скобок) - при его наличии. Точкой применения приема обращения к разложению на множители можно считать сумму - основание степени, являющейся сомножителем знаменателя (соответственно, числителя) дроби, допуская вырожденный случай степени с показателем единица. В нашем примере единственной такой точкой является выражение a^2-ab.

Прием, применяемый здесь для разложения на множители, состоит в вынесении за скобку общего множителя всех слагаемых. Для его программирования понадобится вспомогательная процедура, находящая наибольший общий делитель двух одночленов.

После разложения на множители знаменателя получаем выражение

$$\frac{bx}{a(a-b)}-\frac{3bx}{2a^2}.$$

Прием, выполняющий сложение дробных выражений, сначала находит общий множитель числителей и общий множитель знаменателей. Они сразу выносятся за скобки, после чего применяется обычное преобразование: числители и знаменатели перемножаются "крест накрест" и складываются. Перед тем, как составить результирующую дробь, предпринимается попытка разложить эту сумму на множители. Последнее действие может показаться излишним - ведь уже имеется прием, который пытается разложить на множители числитель. Однако, тогда понадобятся два цикла сканирования задачи вместо одного - система как бы "забудет" о том, что только что сложила дроби, и лишь после нового цикла рассмотрения задачи натолкнется на указанный числитель. Циклы сканирования задачи в решателе обычно составляют главную часть вычислительного времени. Поэтому лучше объединять в одном и том же приеме как основное действие, так и все сопровождающие его дополнительные - это дает весьма ощутимое ускорение.

В нашем случае задача решается с целью сложить дроби, и каких-либо особых эвристических решающих правил не требуется. Достаточно ввести в прием проверку наличия данной цели. Впрочем, можно ввести ограничение, требующее при сложении нескольких дробных выражений начинать с самых коротких. Иногда это упрощает выкладки.

Возвращаемся к цепочке преобразований уравнения, которое после сложения дробных выражений в левой части приобретает вид

$$\frac{bx(3b-a)}{2(a-b)a^2} = 1 - \frac{6a+7b}{6a}.$$

Это уравнение имеет в левой части дробь; для устранения ее естественно домножить обе части уравнения на знаменатель. Однако, если сформулировать прием подобным образом, то он иногда будет приводить к излишним вычислительным затратам. Целесообразно до исключения знаменателя левой части сложить дробные выражения в правой, известной части. Тогда можно будет сначала вынести за скобку общие множители числителей и знаменателей, и лишь затем избавляться от знаменателей. Результатом преобразований станет дизъюнкция, объединяющая в себе уравнение с исключенными знаменателями и частные случаи равенства нулю общих множителей числителей:

$$b = 0 \vee 3x(3b-a) = -7a(a-b).$$

Как и прием сложения дробных выражений, данный прием исключения знаменателей объединяет сразу несколько независимых преобразований, ускоряя тем самым процесс вычислений.

Следующий шаг решения задачи - разбор случаев. Прием, используемый для этого, последовательно рассматривает уравнения, соответствующие подслучаям, и затем объединяет полученные ответы связкой "или". Ответ каждого подслучая упрощается отдельно, однако дизъюнкция ответов, вообще говоря, допускает дальнейшее упрощение - какие-то серии корней или целые промежутки могут склеиваться. Поэтому прием должен обращаться к вспомогательной задаче на упрощение полученной дизъюнкции. Более того, этот же прием должен сразу выдать ответ задачи, иначе дизъюнкция, которая им получена, снова вызовет разбор случаев, и система зациклится.

В нашем примере первый подслучай - условие $b = 0$. Оно не содержит неизвестных, и может быть сразу выдано в качестве ответа. Это действие, хотя и очень простое, тоже требует специального приема. Кроме усмотрения того, что условия задачи не содержат неизвестных, данный прием должен обратиться к вспомогательной задаче на упрощение их конъюнкции, и результат упрощения выдать в качестве ответа.

Второй подслучай - уравнение $3x(3b-a) = -7a(a-b)$. Согласно условиям на область допустимых значений (хотя мы и опустили их рассмотрение, но приведенные выше приемы должны были сопровождать все преобразования коррекцией таких условий), правая его часть отлична от нуля. Поэтому уравнение эквивалентно соотношению

$$x = -\frac{7a(a-b)}{3(3b-a)}.$$

Формулировка приема, выполняющего данный переход, очевидна.

Далее следует подстановка найденного значения неизвестной в сопровождающие условия на область допустимых значений и упрощение этих условий. Эти действия

требуют специального приема, выполняющего обращение к задаче на редактирование ответа. Полученный ответ

$$a \neq 0, a - b \neq 0, 3b - a \neq 0, x = \frac{7a(b-a)}{3(3b-a)}$$

возвращается приему разбора случаев, объединяющему его с ответом $a \neq 0, b = 0$ первого подслучая и выдающему окончательный ответ.

Разобранный пример оказался совсем простым с точки зрения управления преобразованиями - каждое действие было практически однозначным. Рассмотрим несколько более сложный случай - решение системы уравнений. Будем решать систему:

$$\begin{cases} x^2 + y^2 &= z^2 \\ xy + yz + xz &= 47 \\ (z-x)(z-y) &= 2. \end{cases}$$

Поначалу каких-то соображений однозначного характера о ее преобразованиях не возникает. Однако, есть соображения о том, с чего начинать анализ ситуации. Например, можно попробовать раскрыть скобки в третьем уравнении и посмотреть, не станут ли очевидными следующие шаги. Это - тоже преобразование, однако не основной задачи на описание, а сопровождающей ее задачи на исследование, в которой накапливаются общие следствия посылок и условий. Здесь мы имеем сразу два приема: первый принимает решение о переходе к выводу следствий в задаче на исследование, второй - выводит из третьего уравнения следствие, получаемое раскрыванием скобок. Впрочем, в действительности эти приемы в решателе переставлены местами - раскрывание скобок выполняется сразу же. Случаи, когда оно может повредить, отслеживаются приемами, срабатывающими на меньших уровнях. Создавая прием для раскрывания неизвестных скобок в уравнениях, мы не должны сразу же предусматривать какие-то ограничения на его применение. Если они необходимы, то проявятся позднее, при анализе других задач. Так или иначе, получаем в списке посылок задачи на исследование первые два уравнения, сопровождаемые уравнением $xy - xz - yz + z^2 = 2$. Теперь становится видно, что при сложении второго и третьего уравнений уничтожаются сразу два неизвестных слагаемых в левой части. Получается уравнение $2xy + z^2 = 49$ - следствие двух исходных уравнений. Прием, выполняющий это действие, можно несколько обобщить - чтобы он искал линейную комбинацию двух уравнений, позволяющую устранить сразу два неизвестных слагаемых. Следующий естественный шаг - сложить первое уравнение с полученным следствием, чтобы исключить неизвестную z. Формулировка соответствующего приема несложна. После сложения уравнений имеем следствие $x^2 + y^2 + 2xy = 49$. Очередной шаг очевиден - представить левую часть как полный квадрат: $(x + y)^2 = 49$. Прием, выполняющий это действие, обращается к вспомогательной задаче на разложение левой части уравнения на множители. Если такое разложение удается, то есть вероятность, что полученное уравнение будет полезно для дальнейшего (например, чтобы поделить два уравнения, сократив неизвестный множитель). Разумеется, нужно принять меры, чтобы прием не пытался разложить на множители левую часть уравнения, полученного после раскрывания скобок, и обратно. Для этого можно использовать специальные комментарии к уравнениям, блокирующие "обратный ход". Следующий шаг - решение простейшего степенного уравнения. После него появляется дизъюнкция $x + y = 7 \ \vee \ x + y = -7$. Обычно получение дизъюнкции в задаче на исследование означает, что следует вернуться в исходную задачу на описание и

предпринять разбор случаев. В нашей ситуации нужно, кроме того, учесть, что найденная дизъюнкция эквивалентна, при наличии первых двух уравнений, третьему. Это позволяет при разборе случаев исключить третье уравнение. Таким образом, цикл вывода следствий из уравнений завершился, и далее решаем две независимых системы. Ограничимся рассмотрением первой из них -

$$\begin{cases} x+y &= -7 \\ x^2+y^2 &= z^2 \\ xy+yz+xz &= 47. \end{cases}$$

Очередной шаг состоит в преобразовании первого уравнения к виду $x = -(y+7)$. Прием, реализующий это шаг, можно было бы представить как обращение к вспомогательной задаче, разрешающей одно из уравнений системы для исключения неизвестной. На первый взгляд, такое обобщение нашего шага может показаться естественным. Однако, при рассмотрении последующих примеров выяснилось бы, что данный прием почти всегда вреден - если произвольно выбрать какое-то уравнение системы и разрешить (пусть даже успешно) относительно одной из неизвестных, то чаще всего возникает такое усложнение системы, после которого решить ее становится весьма затруднительно. Сравнивая случаи, где выражение одной неизвестной через другие полезно, со случаями, где оно вредно, можно было бы создать несколько приемов, применяемых в очень специальных ситуациях. Для нашей задачи вводим прием, выражающий неизвестную из уравнения, если оно линейно по всем своим неизвестным.

Далее подставляем найденное выражение для x через y и переходим к решению системы относительно y, z. Это делается отдельным приемом, который должен создать вспомогательную задачу для двух неизвестных и обратиться к ее решению. Завершающая обработка ответа допускает разные варианты. Можно получить ответ на вспомогательную задачу, объединить его с уравнением $x = -(y+7)$ и упростить результат. Однако, если при решении вспомогательной задачи происходил разбор случаев и ответ на нее имеет вид дизъюнкции, то при упрощении снова понадобится разбор случаев. Поэтому в решателе реализован другой способ - уравнение $x = -(y+7)$ передается вспомогательной задаче через ее технические структуры данных и извлекается оттуда при редактировании ответа для каждого подслучая. Тогда после решения вспомогательной задачи какой-либо дополнительной обработки ответа не потребуется.

После перехода к неизвестным y, z выполняются простые преобразования, связанные с общей стандартизацией вида уравнений и с раскрыванием скобок. Они реализуются простыми приемами, на которых можно сейчас не останавливаться. В результате возникает система $14y+2y^2-z^2 = -49, 7y+7z+y^2 = -47$. Легко заметить, что члены с неизвестной y в обоих уравнениях пропорциональны, и после вычитания из первого уравнения удвоенного второго остается уравнение с единственной неизвестной z. Прием, усматривающий возможность получить уравнение с единственной неизвестной за счет линейной комбинации уравнений, практически не требует каких-либо дополнительных ограничений на целесообразность срабатывания. В результате получаем систему $14y + 2y^2 - z^2 = -49, -14z - z^2 = 45$. Дальнейшие действия очевидны, и мы их разбирать не будем.

Разобранные примеры продемонстрировали два режима работы решателя - эквивалентные преобразования уравнения в первом случае и вывод следствий для получения дополнительной информации о неизвестных во втором. Первый режим, априори, требует большой осмотрительности при выборе каждого действия, так как неудач-

ное преобразование может завести задачу в тупик. В действительности, однако, для многих областей наблюдается явление устойчивости: если в целом следовать некоторым несложным представлениям о том, какие преобразования упрощают задачу, то выбор конкретного порядка их выполнения несущественен - в любом случае за разумное число шагов получается один и тот же ответ. Это и позволяет в таких областях "избавляться от перебора". Второй режим позволяет исключать перебор в его классическом виде рассмотрения дерева задач, сводя поиск решения к рассмотрению расширяющегося списка посылок одной и той же задачи. Так как приемы, выводящие следствия, снабжены решающими правилами, отсекающими малоперспективные (например, чрезмерно громоздкие) с точки зрения эксперта следствия, то через определенное время наступает полное исчерпание разумных следствий, и процесс обрывается. Процедура решения задачи, вместо перебора, приобретает здесь характер "логического замыкания" исходных данных.

Рассмотрим пример, в котором возникает еще одна разновидность "логического режима". Будем решать задачу на доказательство неравенства

$$0 \leq a^4 - 2a^3b + 2a^2b^2 - 2ab^3 + b^4.$$

В таких задачах часто помогает прием, использующий неравенство для среднего арифметического и среднего геометрического. Удобнее переформулировать его в виде приема, выделяющего в оцениваемой сумме квадрат суммы или квадрат разности. В нашей задаче можно заметить, что слагаемые a^2b^2, b^4 и $-2ab^3$ представляют собой квадрат разности величин ab, b^2. Поэтому, если доказать неравенство $0 \leq a^2b^2 + a^4 - 2a^3b$, полученное отбрасыванием данных слагаемых, то задача будет решена. Применяя к последнему неравенству тот же прием, получаем искомое доказательство.

Заметим, что выбор группы слагаемых, образующих квадрат суммы или разности, выполняется неоднозначно. Например, на первом шаге можно было бы выделить группу $2a^2b^2, a^4, b^4$ и сразу завести задачу в тупик. По этой причине может показаться, что прием выделения оцениваемой группы слагаемых требует какого-то сильного управления, без чего возникнет трудоемкий перебор. Однако, данный прием обычно сильно упрощает правую часть неравенства, так что длина цепочек неравенств, возникающих при доказательстве, невелика. Кроме того, часто появляются тупиковые ситуации, в которых прием неприменим. Поэтому реальный перебор оказывается невелик, и особых проблем с принятием решения не возникает. Здесь возникает режим "ограниченного перебора", у которого ограничения обусловлены не какими-то специальными решающими правилами, а просто тем фактом, что сложность задачи при каждом шаге сильно уменьшается. Этот режим используется в решателе для разных разделов (например, вычисление пределов и интегрирование), и обычно дает очень быстрое получение ответа.

Перейдем к примерам из других разделов. Рассмотрим сначала простую геометрическую задачу на вычисление. В параллелограмме $ABCD$ из точки N пересечения диагоналей проведены перпендикуляры NF, NE к сторонам AB,AD. Длины этих перпендикуляров равны, соответственно, p, m. Угол BAD равен a. Найти длины x, y диагоналей AC, BD и площадь параллелограмма z.

Схема решения геометрических задач на вычисление напоминает схему решения систем уравнений - происходит вывод следствий из посылок и условий до тех пор, пока не возникают либо равенства для значений неизвестных, либо уравнения, из которых значения неизвестных извлекаются чисто алгебраическими методами. В начале процесса вывода порождаются совсем простые утверждения, которые можно

рассматривать как логическое представление чертежа задачи. В нашем примере таковыми являются утверждения о параллельности прямых AB, CD и AD, BC. При решении геометрической задачи последовательность рассуждений допускает множество вариаций. Итоговая картина при этом обычно оказывается не очень чувствительной к ее конкретной версии. Приведем для нашего примера одну из таких возможных последовательностей.

Прежде всего, замечаем, что в задаче нужно найти площадь параллелограмма и что к стороне AD проведен перпендикуляр NE. Для получения высоты, длина которой участвует в формуле площади, продолжаем перпендикуляр до пересечения с противоположной стороной в точке G, и одновременно выписываем соотношение $z = l(AD)l(EG)$. Эти действия оформляем в виде отдельного приема. Основаниями для его применения служат указанные выше признаки - упоминание в задаче о площади параллелограмма и "недоведенный" до конца перпендикуляр к одной из сторон.

Продолжая общий анализ чертежа, выводим из равенства $\angle(BAD) = a$ соотношения для четырех остальных углов параллелограмма. Основанием для применения этого приема является упоминанение в задаче хотя бы одного из углов параллелограмма.

Аналогичным образом, выводим равенство длин противоположных сторон параллелограмма $l(BC)$ и $l(AD)$. Основанием применения приема служит упоминание в задаче одной из этих длин.

Так как выделена точка E пересечения диагоналей параллелограмма, замечаем, что диагонали делятся в ней пополам: $l(CN) = l(AN)$, $l(DN) = l(BN)$. Еще один прием отмечает, что точка E не просто лежит на прямой BD, но является точкой отрезка BD. Основанием для его срабатывания служит наличие в посылках задачи равенства $l(DN) = l(BN)$. Этот же прием устанавливает, что точка N лежит на отрезке AC.

Так как точка N находится на отрезке AC, причем длина отрезка и расстояния от N до его концов упоминаются в задаче, то выводим соотношение равенства длины отрезка сумме длин подотрезков: $x = 2l(AN)$. Аналогично, выводим $2l(BN) = y$.

Далее выводится условие принадлежности точки N отрезку EG - проведенной высоте параллелограмма. Основаниями являются принадлежность точки N диагонали и параллельность сторон. Отсюда, аналогично предыдущему, получается следствие $l(EG) = 2m$ - прием о равенстве длины отрезка сумме длин подотрезков срабатывает уже в третий раз.

Так как угол BAD и высота $l(EG)$ теперь известны, можно выписать тригонометрическое соотношение $\sin(a)l(AB) = 2m$, в котором единственный неизвестный числовой параметр - длина отрезка AB. Выписывание таких соотношений (возможно, не на самых малых уровнях сканирования задачи) представляется разумным, так как постепенно доопределяются новые параметры чертежа.

После того, как предыдущее соотношение занесено в посылки задачи, оно преобразуется к виду $l(AB) = 2m/\sin(a)$. Здесь работает прием, разрешающий линейное соотношение относительно числового параметра, определенного через ссылки на точки.

Как только оказалась введена в рассмотрение длина стороны AB, срабатывает уже упоминавшийся выше прием, выписывающий равенство длин сторон AB,CD.

Далее вводится высота параллелограмма CH, проведенная к продолжению стороны AB. Основанием для этого действия служит то, что, во-первых, длины отрезков AN,NC пропорциональны, а длина p высоты FN - известна. Отсюда прием выводит следствие $l(CH) = 2p$. Во-вторых, основанием для срабатывания является наличие

известного угла CBA, который мог бы позволить выразить впоследствии через $l(CH)$ какие-то новые параметры чертежа.

Условие перпендикулярности прямых CH и AB преобразуется в условие параллельности прямых CH и FN - чтобы в дальнейшем иметь дело с классами параллельных прямых, выбирая в каждом таком классе единственного представителя.

Снова применяется прием, выписывающий тригонометрическое соотношение

$$\sin(a)l(AD) = 2p.$$

Однако, это другой прием, срабатывающий на меньшем уровне, чем рассмотренный выше. Оснований для его срабатывания больше - параметр $l(AD)$ к этому моменту уже встречается в задаче. Введенное соотношение сразу преобразуется к виду $l(AD) = 2p/\sin(a)$.

Теперь начинает срабатывать прием, подставляющий найденное значение $l(AD)$ во все посылки, где этот параметр встречается. В частности, таким образом из посылки $z = 2ml(AD)$ получаем фрагмент ответа $z = 4mp/\sin(a)$. Накопление информации о параметрах чертежа "само собой" привело к нахождению искомой площади. Разумеется, существенную роль при этом сыграли приоритеты на получение соотношений, устанавливающих связь данных и искомых параметров. Вывод следствий имеет характер "встречного распространения" цепочек зависимостей - от известных и от неизвестных величин.

К текущему моменту определились длины сторон параллелограмма и его углы. Поэтому для нахождения неизвестных диагоналей и завершения решения остается воспользоваться теоремой косинусов. Основания для срабатывания приема - известны длины двух сторон треугольника и угол между ними, а длина третьей стороны связана с неизвестными задачи каким-либо соотношением.

Перечисленные выше действия в действительности совпадают с действиями решателя. Таким образом, видно, что даже после накопления большой базы приемов удается избежать чрезмерного количества выводимых следствий. Это достигается тенденцией вводить при обучении как можно более сильные ограничения на срабатывания, пропускающие лишь действительно разумные шаги.

Сразу заметим, что при обучении решателя геометрическим задачам было допущено одно отклонение от "человеческих" стандартов. Вместо того, чтобы в цикле предварительного анализа чертежа определять, какие его элементы можно выразить через другие, и лишь затем выписывать нужные соотношения, решатель вводит эти соотношения сразу. Однако, за исключением редких случаев, он никак их не преобразует, а использует лишь для установления самого факта взаимной выразимости параметров - как своего рода "граф" взаимосвязей. На быстродействии компьютерной системы это сказывается мало, но у пользователя, анализирующего ход решения по шагам, складывается впечатление об избыточной громоздкости накапливаемой информации. Впрочем, за счет дальнейшего усиления решающих правил, с данным явлением можно достаточно эффективно бороться. Иногда оно и вовсе незаметно.

Следующая задача - на качественное исследование поведения функции $f(x) = x^x$ с помощью пределов и производных. Она решается по схеме, аналогичной только что разобранному геометрическому примеру. Решение заключается в последовательном выводе следствий, характеризующих функцию f, и в отборе тех из них, которые целесообразно включить в итоговый список. Прежде всего применяется прием, находящий область определения функции. Он выписывает условия на область допустимых значений x и обращается к вспомогательной задаче на упрощение класса

таких значений. Выводится следствие $\mathrm{Dom}(f) = (0, \infty)$. Следующий шаг - вычисление производной. Получаются следствия $g(x) = (1 + \ln x)x^x$; "Производная(f, g)". Снова применяется прием, определяющий область определения введенной в рассмотрение функции g. После того, как явно указаны области определения f, g, вводится посылка, указывающая множество точек, где производная не определена либо не вычислена: $\mathrm{Dom}(f) \setminus \mathrm{Dom}(g) = \emptyset$. Следующий шаг - срабатывание приема, обращающегося к вспомогательной задаче для отыскания корней производной и вводящего следствие $\mathrm{roots}(g, \mathrm{Dom}(g)) = \{1/e\}$. Чтобы анализировать поведение функции в точках, где ее производная определена и отлична от нуля, эта область явно находится и разбивается на промежутки. Соответствующий прием обращается к вспомогательной задаче, осуществляющей такое разбиение, после чего появляется посылка "областьроста$(f, (0, 1/e) \cup (1/e, \infty))$", распадающаяся на утверждения "областьроста$(f, (0, 1/e))$", "областьроста$(f, (1/e, \infty))$". Используя информацию о промежутках монотонности и анализируя знак производной на их стыке (точка $1/e$), следующий прием выводит утверждения "возрастает$(f, (1/e, \infty))$", "убывает$(f, (0, 1/e))$". Далее применяется прием, анализирующий знаки функции f на интервале монотонности. Он выводит следствия о числе корней на интервале: $\mathrm{card}(\mathrm{roots}(f, (0, 1/e))) = 0$, $\mathrm{card}(\mathrm{roots}(f, (1/e, \infty))) = 0$, немедленно преобразуемые к виду $\mathrm{roots}(f, (0, 1/e)) = \emptyset$, $\mathrm{roots}(f, (1/e, \infty)) = \emptyset$. Следующий прием усматривает экстремум на стыке промежутков монотонности -

$$\mathrm{Extr}(f, \frac{1}{e}, \frac{1}{\exp \frac{1}{e}}, \min).$$

На этом поток вывода следствий исчерпывается, и выдается ответ, в который отбираются: соотношение, определяющее функцию f, информация об экстремуме, о промежутках монотонности, об области определения функции f и о числе ее корней на промежутках монотонности.

В этом примере управляющая компонента приемов была почти вырожденной - по существу, совокупность приемов составляла алгоритм исследования функции. Однако, разбиение такого алгоритма на отдельные почти не связанные друг с другом фрагменты - приемы - предоставляет несомненные удобства для последующего пополнения его все новыми и новыми элементами, ориентированными на различные специальные случаи.

Еще один пример, решаемый по схеме вывода следствий - из аналитической геометрии. Пусть стороны треугольника заданы уравнениями $7x+y-2 = 0, 5x+5y-4 = 0, 2x-2y+5 = 0$. Нужно найти координаты точки внутри треугольника, равноудаленной от первых двух прямых и отстоящей от третьей на расстояние $\frac{3\sqrt{2}}{4}$. В аналитической геометрии, а в особенности в таких разделах, как теория вероятностей и физика, обычной математической символики для полной формулировки задачи оказывается уже недостаточно. Поэтому пошаговому анализу решения здесь предшествует анализ возможных вариантов логической формализации условия и отбор наиболее удобных вариантов. В нашем примере обозначения возьмем из логического языка решателя. Пусть вершины треугольника обозначены A, B, C, а прямоугольная система координат, относительно которой берутся уравнения - K. Сами уравнения можно записать тогда, например, в следующем виде: "коорд(прямая(AB),K) = $\mathrm{set}_{xy}(x-$число $\&$ $y-$число $\&$ $7x+y-2 = 0)$"; "коорд(прямая(AC),K) = $\mathrm{set}_{xy}(x-$число $\&$ $y-$число $\&$ $5x+5-4 = 0)$"; "коорд(прямая(BC),K) = $\mathrm{set}_{xy}(x-$число $\&$ $y-$число $\&$ $2x-2y+5 = 0)$". Искомую точку обозначим D; условие ее принадлежности треугольнику запишем в виде $D \in$ фигура(ABC). Равноудаленность точки от первых двух прямых представляем записью "расстдопрямой(D,прямая(AB))= расстдопрямой(D,прямая(AC))". Указы-

ваем расстояние до третьей прямой: "расстдопрямой(D,прямая(BC)) = $\frac{3\sqrt{2}}{4}$". Перечисленные утверждения образуют список посылок задачи, т.е. то, что дано. Условием ее служит равенство $z =$ коорд(D, K), причем переменная z является неизвестной задачи. По постановке задачи, в выражение для z не должны входить обозначения A, B, C, D, K, хотя они и появляются в посылках задачи, а следовательно, формально являются "известными". Это специально оговаривается в целевой установке задачи.

Процесс решения задачи выглядит как вывод следствий из объединенного списка посылок и условий. Приведем цепочку выводов, реализуемую решателем. Хотя она, быть может, и не оптимальна, однако для новичка вполне допустима.

Прежде всего, вводятся координаты точек A, B, C: "коорд(A,K)" = (a, b); "коорд(B,K)" = (c, d); "коорд(C,K)" = (e, f). Основанием для такого действия служит то, что точки встречаются в обозначениях прямых, для которых известны уравнения. Следующий шаг - ввод координатного набора для неизвестной z: $z = (g, h)$. Основанием является то, что в задаче рассматривается расстояние от точки D до прямой, уравнение которой известно. Для всех введенных новых параметров $a, b, \ldots, h$ регистрируются посылки, указывающие, что параметры - числовые. Подставляя координаты точки A в уравнение прямой AB, получаем соотношение $b + 7a - 2 = 0$. Его преобразуем к виду $b = 2 - 7a$ и подставляем b во все остальные посылки задачи. Аналогичным образом, выводим соотношение $d + 7c - 2 = 0$ и преобразуем к виду $d = 2 - 7c$. Подставляя координаты точки A в уравнение прямой AC, находим $a = 1/5$. Все эти действия выполняются простыми приемами, срабатывающими без ограничений. Подставляя значение a в уравнение для b, находим $b = 3/5$. Для точек B, C выполняем аналогичные действия; в итоге получаем $c = -1/16$, $d = 2 + 7/16$, $e = -17/20$, $f = 33/20$.

Определив координаты точек A, B, C (быть может, они для дальнейшего и не понадобятся), решатель переходит к выводу соотношений для искомых координат g, h. Так как уравнение прямой AC известно, можно воспользоваться формулой для расстояния от точки до прямой и получить соотношение

$$50(\text{расстдопрямой}(D, \text{прямая}(AC)))^2 = 50gh + 25g^2 + 25h^2 - 40g - 40h + 16.$$

Основанием для этого действия служит то, что указанное расстояние уже упоминается в задаче, а уравнение прямой известно. Выражаем квадрат расстояния:

$$(\text{расстдопрямой}(D, \text{прямая}(AC)))^2 = \frac{50gh + 25g^2 + 25h^2 - 40g - 40h + 16}{50}.$$

Аналогичным образом, для расстояния от D до прямой AB получаем:

$$(\text{расстдопрямой}(D, \text{прямая}(AB)))^2 = \frac{14gh + 49g^2 + h^2 - 28g - 4h + 4}{50}.$$

С учетом посылки задачи, указывающей равенство данных расстояний, выводим соотношение

$$\frac{50gh + 25g^2 + 25h^2 - 40g - 40h + 16}{50} = \frac{14gh + 49g^2 + h^2 - 28g - 4h + 4}{50}.$$

Далее переходим к расстоянию от D до прямой BC:

$$(\text{расстдопрямой}(D, \text{прямая}(BC)))^2 = \frac{-8gh + 4g^2 + 4h^2 + 20g - 20h + 25}{8}.$$

В это равенство подставляем значение расстояния, данное в формулировке задачи:

$$\frac{9}{8} = \frac{-8gh + 4g^2 + 4h^2 + 20g - 20h + 25}{8}.$$

Начинается цепочка общей стандартизации последних уравнений. После устранения знаменателей и сокращения они преобразуются к виду: $5g - 2gh - 5h + g^2 + h^2 = -4$, $g + 3h + 2g^2 - 3gh - 2h^2 = 1$.

Теперь начинается учет условия принадлежности точки D треугольнику. Здесь-то и используются найденные ранее координаты вершин. Условие расположения точки D по ту же сторону от прямой AB, что и точка C, дает неравенство $h + 7g - 2 \leq 0$. В случае прямой AC получаем $0 \leq 5g + 5h - 4$. В случае прямой BC - $0 \leq 2g - 2h + 5$.

Далее решатель усматривает систему из двух приведенных выше уравнений относительно числовых параметров g, h и решает ее вне общего контекста, учитывая также неравенства для g, h. В результате получается $g = 0, h = 1$, $z = (0, 1)$, и задача решена.

Процесс вывода следствий похож на решение задачи по элементарной геометрии, однако управление приемами здесь существенно проще.

Следующий пример - задача по теории вероятностей. Формулировка ее такова. Радиолокационная станция ведет наблюдение за объектом, который может применять или не применять помехи. Если объект применяет помехи, то за один цикл обзора станция обнаруживает его с вероятностью p_1; если не применяет - с вероятностью p_2. Вероятность того, что во время цикла будут применены помехи, равна p и не зависит от того, как и когда применялись помехи в остальных циклах. Найти вероятность того, что объект будет обнаружен хотя бы один раз за n циклов обзора.

Хотя обучение логической системы пониманию текстов и начато, но объем предстоящей здесь работы чрезвычайно велик, и пока логическую формализацию задач текстового характера придется выполнять вручную. В любом случае, определение логического языка должно будет предшествовать созданию системы, понимающей тексты, ибо переводить их придется именно на этот язык. В нашем примере введем обозначение $A(i)$ для случайных событий, состоящих в применении помех на i-м цикле, а также обозначение $B(i)$ для случайных событий, состоящих в обнаружении объекта на i-м цикле. Обозначим также C вероятностное пространство. Тогда формулировка задачи на логическом языке может быть представлена в следующем виде:

"незавсобытия(A,C)" (независимость событий семейства A);

$l(A) = n$ (число событий в семействе A);

$\forall_i(i \in \{1, \ldots, n\} \rightarrow$ вероятность$(A(i), C) = p)$ (вероятность события $A(i)$);

"незавсобытия(B,C)";

$l(B) = n$;

$\forall_i(i \in \{1, \ldots, n\} \rightarrow$ услвероятн$(B(i), A(i), C) = p_1)$ (вероятность события $B(i)$ при условии события $A(i)$);

$\forall_i(i \in \{1, \ldots, n\} \rightarrow$ услвероятн$(B(i),$ элемсобытия$(C) \setminus A(i), C) = p_2)$ (вероятность события $B(i)$ при условии отсутствия события $A(i)$).

Смысл логических символов, используемых в этих утверждениях, понятен из их названия. После перечисления посылок задачи формулируем ее условие. Очевидным образом, оно состоит в определении вероятности объединения событий семейства B:

$$x = \text{вероятность}(\bigcup_{i=1}^{n} B(i), C).$$

Здесь x - неизвестная.

Первый шаг в решении - выразить вероятность объединения событий $B(i)$ через вероятности отдельных $B(i)$, используя их независимость. Прием преобразует исходное условие к виду

$$x = 1 - \prod_{i=1}^{n}(1 - \text{вероятность}(B(i), C)).$$

Каких-либо ограничений на срабатывание приема не имеется - он применяется всегда, как только усматривается объединение независимых событий. Разумеется, это не исключает возможности появления таких ограничений при продолжении обучения.

Дальнейшее решение происходит в рамках вывода следствий из условия и посылок. Числовое выражение "вероятность($B(i)$,C)", встречающееся под знаком конечного произведения, можно попытаться выразить через числовые переменные. Прием, выполняющий эту попытку, обращается к решению вспомогательной задачи. Посылки ее получаются присоединением к посылкам текущей задачи ограничений на переменную i, т.е. утверждений: i – целое, $1 \le i$, $i \le n$. Условие имеет вид "вероятность($B(i)$,C) = a", где a - неизвестная вспомогательной задачи. Используя имеющиеся кванторные посылки, получаем следствия "услвероятн($B(i)$,$A(i)$,C) = p_1", "услвероятн($B(i)$, элемсобытия(C)\$A(i)$,$C$) = p_2". Основанием применения приемов вывода здесь является то, что получаемые утверждения содержат значения функциональных переменных A, B в точке i. Далее применяем формулу полной вероятности, связывающую a с найденными условными вероятностями:

$$p_1\text{вероятность}(A(i), C) + p_2\text{вероятность}(\text{элемсобытия}(C) \setminus A(i), C) = a.$$

Основанием для применения приема служит упоминание в задаче вероятности "вероятность($B(i)$,C)" и обеих условных вероятностей. В найденном утверждении вероятность дополнения преобразуется к виду $1-$вероятность$(A(i), C)$. Затем приводятся подобные члены, и получается "$p_2 + (p_1 - p_2)$вероятность$(A(i), C) = a$". Наконец, из кванторной посылки выводится следствие "вероятность($A(i)$,C) = p". Полученное значение вероятности подставляется в предыдущее уравнение, и возникает равенство $a = pp_1 + p_2(1 - p)$, определяющее значение неизвестной a. По завершении решения вспомогательной задачи возвращаемся в основную задачу, подставляя вместо "вероятность$(B(i), C)$" выражение $p_2 + pp_1 - pp_2$. Под конечным произведением теперь находится выражение, не зависящее от индекса i. Поэтому данное произведение заменяется на

$$(1 - (p_2 + pp_1 - pp_2))^n.$$

Отсюда получаем ответ $x = 1 - (1 + pp_2 - p_2 - pp_1)^n$.

Разобранный пример достаточно типичен для задач по теории вероятности - сначала выполняется цепочка эквивалентных преобразований условия задачи на описание, и лишь затем, если без этого ответ получить не удалось, используется вывод следствий.

Следующий пример - задача по элементарной физике. Формулировка ее, разумеется, текстовая: "С вершины наклонной плоскости высотой 5 м и углом наклона к горизонту 45 градусов начинает соскальзывать тело. Определите скорость тела в конце спуска, если коэффициент трения тела о плоскость 0.19".

Логическую формализацию рассуждений в физике приходится осуществлять в ситуации, когда многое принимается по умолчанию. Например, если специально не оговаривается в задаче, трение не учитывается; нити считаются невесомыми и нерастяжимыми; сопротивление воздуха не учитывается; спуск по наклонной плоскости

происходит вдоль линии с наибольшим уклоном, и т.п. Чтобы извлекать эту, опущенную в постановке задачи информацию, приходится вводить приемы, восстанавливающие ее согласно принятым условностям. В процессе обучения часто складывается ситуация, когда приемы, введенные с ориентацией на какие-то априорные предположения, приходится корректировать, оговаривая новые и новые особые случаи, где эти предположения некорректны. Все это здесь можно считать нормой.

Предлагаемая логическая формализация физических задач никоим образом не претендует на завершенность. Ее следует рассматривать как результат проработки конкретного обучающего материала из элементарной физики, оказавшийся достаточным для работы с этим материалом. Видимо, первые же попытки выйти за рамки элементарной физики потребуют радикального пересмотра всего созданного здесь логического языка. Однако, эту работу можно будет проводить уже не на пустом месте, а с учетом всех плюсов и минусов данной версии.

Уже на этапе выбора базисных предикатов, необходимых для задания положения и скорости тел, возникает некоторое осложнение. Можно было бы, например, обозначать положение тела (для простоты - материальной точки) A в момент t, используя запись вида "Место(A,t)". Однако, в задачах часто встречается ситуация, когда рассматриваемое тело, кроме реального движения, участвует в одном или нескольких воображаемых процессах движения. Например, про пешехода говорится: "если бы он шел на 2 км/час быстрее, то пришел бы на 10 минут раньше". Ясно, что здесь для задания положения пешехода недостаточно указания момента времени. В логической записи нужно вводить еще один параметр, доопределяющий контекст - в каком "мире" происходит движение пешехода (реальном, воображаемом и т.п.). Такой дополнительный параметр пришлось бы вводить для указания любых физических характеристик объекта - скорости, ускорения, силы воздействия и т.п. Все это, безусловно, утяжелило бы логические конструкции. Более рационально пойти по другому пути. Можно объединить пару "объект - контекст его рассмотрения" в понятие процесса, и таким образом получить более экономную запись, сохранив число параметров. Однако, тогда придется основные физические характеристики - положение в пространстве, скорость, ускорение, силу и т.д. связывать уже не с объектом, а с процессом. Впрочем, такая точка зрения становится вполне естественной, если понимать под процессом просто функцию, определенную на некотором временном промежутке и принимающую в качестве значения совокупность качественных и количественных характеристик, полностью определяющих текущее состояние рассматриваемого объекта. Отнесение различных упоминаемых в задаче процессов к общему контексту происходит за счет использования связей между ними, явно указанных в задаче. Если таких связей нет, процессы считаются происходящими в различных контекстах ("мирах", и т.п.).

Согласно вышесказанному, в качестве одного из базисных предикатов для задания движущихся объектов выбран предикат "движение(a b c)". Он означает, что a есть процесс движения материальной точки b вдоль траектории c. Последняя представляет собой какую-либо ориентированную кривую в пространстве, возможно, имеющую самоналожения.

В нашем примере речь идет не просто о движении, а о движении по наклонной плоскости. Такая разновидность движения настолько часто встречается в задачах по элементарной физике, что для нее разумно ввести специальный предикат "движ-наклплоск(a b c d)". Здесь a - процесс движения тела по наклонной плоскости, совершающей процесс движения b; c - прямоугольная система координат в пространстве, определяющая ориентацию наклонной плоскости; d - угол ее наклона. Для опреде-

ленности считаем, что наклонная плоскость параллельна оси OY и наклонена под углом d к направлению оси абсцисс. Угол d изменяется от 0 до $\pi/2$. Можно было бы пойти по другому пути - ввести предикат для движения одного тела по поверхности другого, предикат для характеризации объекта "наклонная плоскость", а также функциональный символ для определения угла. Однако, в этом случае каждый раз пришлось бы дополнительно уточнять, что движение происходит по линии наибольшего уклона, и все вместе взятое оказалось бы существенно более громоздким, чем предикат "движнаклплоск(...)". Несколько неудобно то, что этот предикат позволяет вводить только наклонные плоскости с уклоном "справа налево". Необходимости рассматривать в задачах другие наклонные плоскости пока не возникало; если понадобится, то нетрудно перейти к использованию двух различных предикатов "движлевнаклплоск" и "движправнаклплоск".

Возвращаясь к рассматриваемой задаче, вводим первые две ее посылки - "движнаклплоск(a b K $\pi/4$)", "движение(a c d)". По умолчанию, в задаче присутствует поверхность Земли, по отношению к которой и берется угол наклона. Систему координат, у которой ось OZ направлена вертикально вверх по отношению к данной поверхности, выделяем посылкой "поверхнземли(K)". Чтобы указать на неподвижность наклонной плоскости (опять же предполагаемую по умолчанию), вводим посылку "неподв(b P)". Здесь P - сама наклонная плоскость (как физическое тело). Условие о том, что высота наклонной плоскости равна 5 м, представим как соотношение для разности z - координат тела в начале и конце процесса движения. Будем обозначать положение в момент t материальной точки, совершающей процесс движения a, через "Место(a,t)". i-ю координату точки (либо вектора) A в системе координат K обозначаем посредством "крд(A,K,i)". Тогда посылка задачи записывается в виде: "крд(Место(a, исхмомент(a)),K,3)$-$ крд(Место(a, послмомент(a)),K,3) = 5м". Условие неподвижности тела в начальный момент дает посылку "Скорость(a, K,исхмомент(a)) = вектор0". Здесь посредством "Скорость(A,B,t)" обозначается вектор скорости в момент t объекта, находящегося в процессе движения A, относительно объекта, находящегося в процессе движения B. Допускается в качестве B использовать также ссылку на систему координат. Чтобы задать коэффициент трения, введем в рассмотрение силовое воздействие f наклонной плоскости на движущееся тело. Согласно общему принципу, будем считать это воздействие тоже процессом. Тогда добавляется посылка "воздействие(f,a,b)". Коэффициент трения, характеризующий это воздействие, осредняется на весь период движения, и задается посылкой вида "коэффтрения(f Период(f)) = 0.19".

Для искомой скорости тела в конце спуска вводим единственное условие задачи: $x =$ длина(Скорость($a, K,$ послмомент(a))).

Как видно из этого примера, для физических задач значительную часть процесса обучения составляет выбор логической формализации. Иногда такая формализация даже не требует пополнения ранее введенных понятий, оставаясь при этом достаточно трудоемкой и кропотливой работой. Переходим далее к анализу собственно траектории решения и приводим (с небольшими упрощениями) последовательность действий, реализуемых решателем.

Сначала применяются приемы, выполняющие простейшую стандартизацию посылок и их пополнение. Во-первых, все последние моменты процессов выражаются через исходные моменты и длительность. Во-вторых, выписываются принимаемые по умолчанию соотношения о равенстве периодов взаимосвязанных процессов (в нашем случае - a и b). В-третьих, добавляется посылка о том, что K - прямоугольная система координат. Для неподвижной наклонной плоскости вводится фиктивный

процесс движения "движение(b,P,e)".

Далее вводится в рассмотрение сила притяжения, действующая на тело a: "воздействие(g,a,K)", "притяжение(g)". В качестве воздействующего объекта условно принимается связанная с поверхностью Земли система координат K. Срабатывает прием, замечающий, что сила притяжения около поверхности Земли постоянна: "констсила(g)". Выводится соотношение равенства периодов процесса движения a и силового воздействия f на тело со стороны наклонной плоскости.

Так как угол наклона известен, выписывается соотношение, связывающее искомую величину скорости тела с вертикальной составляющей этой скорости:

$$|\text{крд}(\text{Скорость}(a, K, \text{исхмомент}(a) + \text{длительность}(a)), K, 3)| = x/\sqrt{2}.$$

Это соотношение может оказаться полезным, так как последующий анализ силовых воздействий даст вертикальную и горизонтальную проекции ускорений.

Чтобы исключить из посылок как можно большее число атомарных числовых характеристик, выражаем далее z - координату исходной позиции тела через z - координату заключительной позиции: "крд(Место(a,исхмомент(a)), K, 3) = 5м + крд(Место(a,исхмомент(a)+длительность(a)), K, 3)". Так как в задаче упоминаются скорости тела в начале и конце спуска, а действующие на тело силы постоянны, то выписываем соотношение, связывающее z - компоненты данных скоростей, длительность спуска и ускорение: "крд(Скорость(a,K, исхмомент(a)+ длительность(a)), K, 3) = длительность(a)крд(Ускорение(a, K,Период(a)), K, 3)".

На этом цикл вывода простейших следствий завершается, и начинается составление списка всех сил, действующих на тело. Предполагается, что к моменту срабатывания данного приема (уровень его равен 3) все такие силы уже явно обозначены в посылках задачи. Прием извлекает ссылки на них из утверждений "воздействие(...)" и добавляет посылку "Силы(a,$\{g, f\}$)". Из постоянства этих сил выводится утверждение "Равносукоренное(a)".

Для равноускоренного движения выписывается соотношение, выражающее z - координату в конце движения через z - координату в начале, ускорение и длительность (исходная скорость равна 0). После несложной стандартизации это соотношение принимает вид:

$$(\text{длительность}(a))^2 \text{крд}(\text{Ускорение}(a, K, \text{Период}(a)), K, 3) = -10\text{м}.$$

С целью уменьшения числа атомарных числовых параметров, присутствующих в посылках задачи, преобразуем последнее соотношение так, чтобы получилось явное выражение для z - компоненты ускорения:

$$\text{крд}(\text{Ускорение}(a, K, \text{Период}(a)), K, 3) = -\frac{10\text{м}}{(\text{длительность}(a))^2}.$$

Найденное выражение подставляется в ранее полученные соотношения. После упрощения равенства с неизвестной x возникает равенство:

$$\frac{10\text{м}}{\text{длительность}(a)} = \frac{x}{\sqrt{2}}.$$

Из него получаем выражение для длительности спуска:

$$\text{длительность}(a) = \frac{10\sqrt{(2)}\text{м}}{x}.$$

Подстановка последнего в выражение для z - компоненты ускорения дает:

$$\text{крд}(\text{Ускорение}(a, K, \text{Период}(a)), K, 3) = -\frac{x^2}{20\text{м}}.$$

После указанной цепочки чисто алгебраических преобразований решатель снова выписывает физическое соотношение. На этот раз применяется прием, выражающий x- и z- координаты силы трения через величину силы тяжести, коэффициент трения и угол наклона плоскости. Ввиду того, что ситуация с движением тела по наклонной плоскости под действием только сил тяжести и воздействия со стороны наклонной плоскости является стандартной, для нее заготовлены формулы, сразу дающие указанное выражение. Основанием для применения приема является упоминание в задаче коэффициента трения. В результате возникают две новые посылки:

$$\text{крд}(\text{силатрения}(f, \text{Период}(a)), K, 1) = \frac{19\text{длина}(\text{сила}(g, \text{Период}(a)))}{200},$$

$$\text{крд}(\text{силатрения}(f, \text{Период}(a)), K, 3) = \frac{19\text{длина}(\text{сила}(g, \text{Период}(a)))}{200}.$$

Следующее действие - подстановка в эти соотношения вместо величины силы притяжения "длина(сила(g,Период(a)))" выражения "масса(c)g" и подстановка вместо g приближенного значения $9.8\text{м}/\text{сек}^2$.

Так как сила трения уже появилась в посылках, вводится соотношение, представляющее силу f воздействия плоскости на тело в виде суммы силы трения и силы нормальной реакции:

$$\text{сила}(f, \text{Период}(a)) = \text{силатрения}(f, \text{Период}(a)) + \text{нормреакция}(f, \text{Период}(a)).$$

Далее, так как в посылках встречается z - координата силы трения, применяется прием, переходящий от векторного соотношения к соотношению для z - координат. После исключения знаменателя получается: $\text{сек}^2\text{крд}(\text{сила}(f, \text{Период}(a)), K, 3) - 0.931\text{масса}(c)\text{м} - \text{сек}^2\text{крд}(\text{нормреакция}(f, \text{Период}(a)), K, 3) = 0$. Для x - координаты силы трения, тоже встречающейся в посылках, выписывается аналогичное соотношение. Из него получается выражение x - координаты силы f через x - координату силы нормальной реакции.

Появление выражений для координат силы нормальной реакции влечет срабатывание приема, выписывающего тригонометрические соотношения, связывающие их с величиной этой силы:

$$\text{крд}(\text{нормреакция}(f, \text{Период}(a)), K, 1) = -\frac{\text{длина}(\text{нормреакция}(f, \text{Период}(a)))}{\sqrt{2}};$$

$$\text{крд}(\text{нормреакция}(f, \text{Период}(a)), K, 3) = -\frac{\text{длина}(\text{нормреакция}(f, \text{Период}(a)))}{\sqrt{2}}.$$

Так как на тело действуют только силы притяжения и воздействия со стороны наклонной плоскости, то применяется прием, связывающий координаты нормальной реакции с величиной силы притяжения:

$$\frac{\text{длина}(\text{нормреакция}(f, \text{Период}(a)))}{\sqrt{2}} = \frac{\text{длина}(\text{сила}(g, \text{Период}(a)))}{2}.$$

Далее идет цепочка подстановок найденных значений и упрощающих преобразований; попутно подставляется явное выражение для величины силы притяжения. В результате возникают выражения для координат нормальной реакции и силы f через массу тела:

$$\text{длина}(\text{нормреакция}(f, \text{Период}(a))) = \frac{9.8\text{масса}(c)\text{м}}{\sqrt{2}\text{сек}^2};$$

$$\text{крд}(\text{нормреакция}(f, \text{Период}(a)), K, 3) = \frac{4.9\text{масса}(c)\text{м}}{\text{сек}^2};$$

$$\text{крд}(\text{нормреакция}(f, \text{Период}(a)), K, 1) = \frac{4.9\text{масса}(c)\text{м}}{\text{сек}^2};$$

$$\text{крд}(\text{сила}(f, \text{Период}(a)), K, 1) = \frac{3.969\text{масса}(c)\text{м}}{\text{сек}^2};$$

$$\text{крд}(\text{сила}(f, \text{Период}(a)), K, 3) = \frac{5.831\text{масса}(c)\text{м}}{\text{сек}^2}.$$

Так как в задаче упоминается z- координата ускорения, причем найден список сил, действующих на тело, то выписывается второй закон Ньютона для z - координаты:

$$\text{крд}(\text{Сила}(a, \text{Период}(a)), K, 3) = -\frac{x^2\text{масса}(c)}{20\text{м}}.$$

В это соотношение подставляется выражение для z - компоненты суммы сил:

$$\frac{5.831\text{масса}(c)\text{м}}{\text{сек}^2} + \text{крд}(\text{сила}(g, \text{Период}(a)), K, 3) = -\frac{x^2\text{масса}(c)}{20\text{м}}.$$

После подстановки явного выражения для z - компоненты силы притяжения и упрощений получается уравнение

$$x^2\text{сек}^2 - 79.38\text{м}^2 = 0.$$

Из посылок задачи нетрудно усмотреть неотрицательность величины x (длина вектора), так что окончательно

$$x = \frac{63\text{м}}{5\sqrt{2}\text{сек}}.$$

Хотя приведенная цепочка действий решателя и выглядит несколько громоздко и беспорядочно, однако почти все выведенные утверждения использованы при получении ответа, так что в итоге избыточность оказывается не очень большой. Некоторым оправданием громоздкости служит явное указание всех подробностей рассуждения. Кроме того, нужно учитывать, что отображено не готовое решение, а процесс его поиска. Обычно при проработке новой задачи, которую решатель заведомо не может решить из-за наличия нового материала, предпринимается трассировка его действий вплоть до выдачи отказа. Анализ завершающего состояния задачи подсказывает, какие именно приемы лучше всего добавить для продолжения цепочки решения. Если после этого задача все еще не решается, то анализируется новая "тупиковая" ситуация, и т.д. При таком режиме обучения борьбе с избыточностью, проявляющейся в каких-либо бесполезных действиях решателя на промежуточных шагах, уделяется совсем мало внимания. После того, как решатель выдает правильный ответ, сразу берется очередная задача. Однако, как показывает только что разобранный пример, даже такая черновая версия, безусловно, требующая последующей существенной оптимизации и переработки, уже демонстрирует сравнительно эффективное поведение.

Завершим разбор примеров рассмотрением текстовой задачи. Формулировка ее такова: "Расстояние между A и B по железной дороге 66 км, а по водному пути - 80.5 км. Из A поезд выходит на 4 часа позже парохода и прибывает в B на 15 мин. раньше парохода. Определить средние скорости поезда и парохода, если первая больше второй на 30 км/час".

При решении задачи можно выделить следующие основные этапы:

1. Этап чтения решателем текста и морфологического разбора каждого слова фразы. Слово разбивается на фрагменты, число которых обычно не более трех. Сначала идет основная часть, включающая в себя корень и приставку; затем - суффикс, и затем - окончание. При морфологическом разборе указываются род, число, время, возможные падежи, часть речи, к которой относится слово, и т.п.

2. Этап синтаксического разбора фразы. Здесь устанавливается подчиненность слов, а для сложных фраз - также подчиненность подфраз.

3. Этап предварительного отождествления объектов. Еще до перехода от естественного языка к логическому предпринимается попытка усмотреть в различных местах фразы ссылки на один и тот же объект. Кроме того, выполняются попытки отождествить объекты, на которые ссылается фраза, с объектами предыдущих фраз, уже перенесенными в логическую структуру данных. Роль такой структуры данных играет список посылок вспомогательной задачи на исследование, имеющей цель "текстоваязадача".

4. Этап получения логического подстрочника текущей фразы. Такой подстрочник представляет собой некоторый список утверждений псевдологического языка, пока никак не связанных с логическим языком решателя. Его логические символы суть условные обозначения, выбранные для кодирования корневых частей слов при регистрации этих слов в словаре текстового анализа. Если обозначение вводится для нового понятия, то оно может быть использовано далее как полноправный элемент логического языка решателя. Если же логический символ, соответствующий слову, уже занят, логический язык придется пополнять другим символом, и на этапе семантического анализа потребуется соответствующая коррекция обозначений.

5. Этап предварительного семантического анализа. Логический подстрочник текущей фразы представляется как список посылок вспомогательной задачи на исследование, имеющей цель "анализфразы". Происходит первоочередная коррекция утверждений, которую можно делать в изолированном контексте одной фразы.

6. Основной этап семантического анализа. Логические фрагменты различных фраз текстовой задачи объединяются в списке посылок задачи на исследование, имеющей цель "текстоваязадача". Здесь и происходит основная часть работы по переводу утверждений на логический язык решателя.

7. Переход к решению задачи, сформулированной на логическом языке решателя. По итогам семантического анализа находится утверждение "найти(...)", которое и указывает на неизвестные вспомогательной задачи на описание. Для решения этой задачи привлекаются обычные возможности решателя.

8. Ответ, полученный при решении задачи на описание, переводится на естественный язык. Применяется процедура, создающая названия фигурирующих в ответе объектов на основе данных, сохранившихся в задаче семантического анализа.

Сейчас мы не будем затрагивать этапы с первого по пятый. Более подробно о них будет рассказано в разделах данной книги, связанных с текстовым анализом. Проследим наиболее интересный с логической точки зрения основной этап семантического анализа. Разумеется, те приемы, которые будут применяться на этом этапе, не дают логически корректных переходов в обычном математическом смысле. Однако, естественный язык содержит множество вполне однозначно расшифровываемых условностей, и фактически речь идет лишь о логике данной расшифровки. Приведем исходный список посылок задачи семантического анализа, составленной решателем по приведенному выше тексту. Трехместные отношения $P(A\ B\ C)$ в этом списке соответствуют упоминаемым в фразе действиям, процессам либо отношениям. Переменная A является обозначением самого действия (отношения); B - обозначением субъекта, и C - обозначением объекта. Все утверждения представляются в скобочной записи, так формульный редактор рассчитан на логический язык решателя, а не на язык логических подстрочников. По этой причине все переменные при текстовом анализе имеют вид "xi".

1. Фраза "Расстояние между A и B по железной дороге 66 км, а по водному пути - 80.5 км" преобразована в следующие два списка термов:

   "расст(x2)"; "между(x2 перечень(набор(x3 x4)))"; равно(x2 умножение(66 км))"; "по(x2 x5)"; "железный(x5)"; "дорога(x5)";

   "расст(x7)"; "между(x7 перечень(набор(x3 x4)))"; "по(x7 x8)"; "чей(x8 x9)"; "вещество(x9 вода)"; "путь(x8)"; "составлять(x6 x7 умножение(80.5 км))".

2. Фраза "Из A поезд выходит на 4 часа позже парохода и прибывает в B на 15 мин. раньше парохода" преобразована в:

   "выходить(x10 x11 x12)"; "поезд(x11)"; "из(x10 x3)"; "выходить(x14 x13 x15)"; "пароход(x13)"; "из(x14 x3)"; "позже(x16 x10 x14)"; "на(x16 умножение(x17 4))"; "час(x17)";

   "прибывать(x20 x13 x21)"; "в(x20 x4)"; "прибывать(x18 x11 x19)"; "в(x18 x4)"; "раньше(x22 x18 x20)на(x22 умножение(15 мин))"; "время(x18 настоящее)".

3. Фраза "Определить средние скорости поезда и парохода, если первая больше второй на 30 км/час" преобразована в:

   "найти(x23 x24 перечень(набор(средний(скорость(x11))средний(скорость(x13)))))";

   "равно(плюс(минус(средний(скорость(x11))) средний(скорость(x13)))минус(умножение(дробь(км час)30)))".

Прежде всего срабатывает прием, связанный с единицей измерения "час". Если имеется посылка "час(x)", то предпринимается подстановка во все выражения "умножение(a x)"логического символа "час" вместо переменной x. В данном примере - утверждение "на(x16 умножение(x17 4))" заменяется на "на(x16 умножение(4 час))".

Логический символ "расст" введен для обозначения при текстовом анализе слова "расстояние", чтобы избежать блокировки большого числа приемов основного решателя, использующих логический символ "расстояние". Такие приемы имеются, например, в геометрии. Блокировка была бы нужна для предотвращения отказов интерпретатора ЛОСа, вызванных тем, что логический символ "расстояние" - двуместный, а в логическом подстрочнике слово "расстояние" возникает как одноместное отношение. В процессе семантического анализа символ "расст" подлежит исключению. В нашем примере исключение выполняет прием, заменяющий посылки "расст(х2)", "равно(х2 умножение(66 км))" на посылку "равно(длина(х2)умножение(66 км))".

В задачах по элементарной физике был введен трехместный предикат "движение(a b c)", означающий, что a есть процесс движения объекта b по траектории c. В действительности этот предикат был подсказан именно задачами на анализ текста, рассмотрение которых предшествовало в обучении решателя проработке задач по физике. К нему и будем сводить разнообразные варианты ссылок на процессы движения, встречающиеся в текстовых задачах. Остальные понятия логического подстрочника тоже постепенно будут заменяться понятиями, используемыми в физических задачах.

В нашем примере встречается утверждение "выходить(х14 х13 х15)". Оно означает, что х14 есть начальный момент процесса движения, выполняемого объектом х13. Решатель вводит две новых посылки - "движение(х1 х13 х25)" и "равно(х14 начало(х1))".

Процесс х14 выхода парохода х13 сопровождается указателем "из(х14 х3)". Это дает основание добавить новую посылку "равно(началопути(х25)х3)".

Для поезда х10 аналогичным образом выводятся утверждения "движение(х26 х11 х27)", "равно(х10 начало(х26))" и "равно(началопути(х27)х3)".

Посылки "составлять(х6 х7 умножение(80.5 км))", "расст(х7)", "между(х7 перечень(набор(х3 х4)))", "по(х7 х8)" означают, что длина пути х8 между х3,х4 равна 80.5 км. Они заменяются на "между(х8 перечень(набор(х3 х4)))", "равно(длина(х8)умножение(80.5 км))".

Аналогичный прием в случае поезда заменяет посылки "равно(длина(х2)умножение(66 км)), "по(х2 х5)", "между(х2 перечень(набор(х3 х4)))" на "между(х5 перечень(набор(х3 х4)))", "равно(длина(х5)умножение(66 км))".

Посылки "раньше(х22 х18 х20)", "на(х22 умножение(15 мин))" заменяются на одну посылку "равно(момент(х20)плюс(момент(х18)умножение(15 мин)))".

Наличие посылок "движение(х26 х11 х27)" и "прибывать(х18 х11 х19)" дает основание заменить во всех прочих посылках выражение "момент(х18)"на "послмомент(х26)" - последний момент процесса движения х26. Аналогичным образом решатель поступает с посылками "движение(х1 х13 х25)", "прибывать(х20 х13 х21)", заменяя повсюду "момент(х20)" на "послмомент(х1)".

Из посылок "движение(х1 х13 х25)". "прибывать(х20 х13 х21)" делается вывод "равно(х20 конец(х1))".

Далее рассматриваются посылки "движение(х1 х13 х25)", "равно(конец(х1)х20)", "в(х20 х4)". Из них извлекается утверждение "равно(конецпути(х25)х4)".

Аналогичным образом, для поезда выводятся утверждения "равно(х18 конец(х26))", "равно(конецпути(х27)х4)".

Так как в основном решателе логический символ "час" представляет собой единицу измерения и используется без скобок, то предпринимается замена посылки "час(х17)" на "равно(х17 час)".

В тексте задачи упоминаются два пути - железная дорога и водный путь, однако нигде явно не говорится, по какому из этих путей следует поезд, а по какому - пароход. Теперь наступает момент, когда необходимо сделать соответствующий выбор. Вообще говоря, этот выбор можно было бы подразбить на несколько элементарных шагов. Пока в решатель введен загрубленный прием, принимающий решение за один шаг. Он усматривает посылки "пароход(x13)", "движение(x1 x13 x25)", "равно(конецпути(x25)x4)", "равно(началопути(x25)x3)", "путь(x8)", "чей(x8 x9)", "вещество(x9 вода)", "между(x8 перечень(набор(x3 x4)))" и делает вывод "равно(x25 x8)". Для большей достоверности можно было бы ввести в прием фильтры, проверяющие, что в задаче не упоминается другой водный путь или какой-то его аналог (река, море, и т.п.).

Аналогичный прием усматривает посылки "поезд(x11)", "движение(x26 x11 x27)", "равно(конецпути(x27)x4)", "равно(началопути(x27)x3)", "дорога(x5)", "железный(x5)", "между(x5 перечень(набор(x3 x4)))" и делает вывод "равно(x27 x5)".

Требование "найти(x23 x24 перечень(набор(средний(скорость(x11))средний(скорость(x13)))))" преобразуется к виду, в котором искомые величины соединены связкой "и": "найти(x23 x24 и(средний(скорость(x11))средний(скорость(x13))))".

Так как единственным рассматриваемым в задаче процессом движения поезда x11 является процесс x26, то выражение "скорость(x11)" заменяется на "скорость(x26 x5)", т.е. на среднюю скорость процесса x26. После этого относящийся к данной скорости указатель "средний(...)" становится излишним и удаляется. Аналогичным образом, выражение "средний(скорость(x13))" за два шага преобразуется в "скорость(x1 x8)". Посылка, определяющая искомые величины, приобретает вид "найти(x23 x24 и(скорость(x26 x5)скорость(x1 x8)))". Аналогичные преобразования выражений "скорость(...)" выполняются в остальных посылках задачи.

Утверждения "позже(x16 x10 x14)" и "на(умножение(4 час))" заменяются на равенство "равно(момент(x10) плюс(момент(x14) умножение(4 час)))".

Так как имеются посылки "движение(x1 x13 x8)" и "выходить(x14 x13 x15)", то выражение "момент(x14)" заменяется на "исхмомент(x1)". Аналогичным образом, "момент(x10)" заменяется на "исхмомент(x26)".

На этом цепочка преобразований логического подстрочника завершается. Список утверждений, формулирующих задачу на логическом языке решателя, полностью укомплектован. Далее срабатывает прием, который по установке "найти(x23 x24 и(скорость(x1 x8)скорость(x26 x5)))" строит задачу на описание, имеющую своими неизвестными вспомогательные переменные x2,x6; условиями - равенства "равно(x6 скорость(x26 x5))" и "равно(x2 скорость(x1 x8))", а посылками - все посылки результата семантического анализа, за исключением установки "найти(...)" и ряда других посылок, остающихся за рамками собственного языка решателя. Для большей наглядности, перечислим эти посылки. Сначала приведем посылки, сформулированные на логическом языке решателя - для решения задачи понадобятся только они:

1. "равно(исхмомент(x26)плюс(умножение(4 час)исхмомент(x1)))";
2. "равно(конецпути(x5)x4)";
3. "равно(конец(x26)x18)";
4. "равно(конецпути(x8)x4)";
5. "равно(послмомент(x1)плюс(умножение(15 мин)послмомент(x26)))";

6. "равно(длина(x5)умножение(66 км))";
7. "равно(длина(x8) умножение(80.5 км))";
8. "равно(началопути(x5)x3)";
9. "движение(x26 x11 x5)";
10. "равно(началопути(x8)x3)";
11. "движение(x1 x13 x8)";
12. "равно(плюс(минус(скорость(x26 x5)) скорость(x1 x8)) минус(дробь(умножение(30 км)час)))";

Кроме того, сохраняются следующие посылки, сохранившиеся от этапа семантического анализа:

1. "равно(конец(x1)x20)";
2. "равно(конец(x26)x18)";
3. "между(x5 перечень(набор(x3 x4)))";
4. "между(x8 перечень(набор(x3 x4)))";
5. "равно(начало(x26)x10)";
6. "равно(начало(x1)x14)";
7. "прибывать(x20 x13 x21)";
8. "в(x20 x4)";
9. "в(x18 x4)";
10. "прибывать(x18 x11 x19)";
11. "выходить(x14 x13 x15)";
12. "пароход(x13)";
13. "из(x14 x3)";
14. "из(x10 x3)";
15. "поезд(x11)";
16. "выходить(x10 x11 x12)";
17. "путь(x8)";
18. "железный(x5)";
19. "дорога(x5)"

Эти посылки почти не замедляют решение, ибо приемов, связанных с их понятиями, в решателе нет. Поэтому средств для исключения таких посылок пока не введено.

Дальнейший ход решения задачи обеспечивается приемами по элементарной физике, наподобие разобранного выше примера. Находится ответ "равно(х2 умножение(14 дробь(км час)))", "равно(х6 умножение(44 дробь(км час)))", и далее включается процедура, предпринимающая попытку определить названия для х2 и х6. Здесь используется информация, сохранившаяся в списке посылок задачи семантического анализа. Фактически, процедура представляет собой небольшую базу приемов. Для х2 определяется название "скорость пархода"; для х6 - "скорость поезда". Окончательный текст ответа, выводимый на экран, имеет вид "Скорость поезда равна 44 км/час, скорость парохода равна 14 км/час".

База приемов, используемых логической системой для семантического анализа, пока крайне невелика. Эти приемы возникли всего лишь из полусотни проработанных текстовых задач. Однако, она позволяет рассмотреть траектории решения данных задач с хорошей степенью детализации и заставляет осознать подлинный объем работы, необходимой для обучения решателей пониманию естественного языка. Видимо, для ускорения этого процесса понадобятся какие-то дополнительные технологические средства. Пока процедура пополнения базы приемов семантического анализа состоит в поиске таких фрагментов текущего состояния логического подстрочника, которые достаточно убедительным образом транслируются на формальный язык. Степень убедительности не всегда оказывается достаточной; эти случаи нужно рассматривать как постановки задач для поиска более приемлемых версий. Видимо, во многих ситуациях полезными будут промежуточные понятия, не относящиеся к логическому языку и исключаемые по окончании семантического анализа. Цепочки вывода с применением таких понятий могут сделать расшифровку подстрочника более однозначной.

4.6 Сравнение логической системы с системами компьютерной математики

После того, как возникло предварительное представление о работе решателя, можно предпринять сравнение его с другими системами компьютерной математики, рассмотрев серию конкретных примеров. Однако, перед таким сравнением следует еще раз подчеркнуть принципиальное различие целей, которые ставились при разработке решателя и целей, для которых были созданы упомянутые коммерческие системы. Решатель представляет собой инструмент для изучения логических процессов путем их моделирования. Он необходим для получения ответа на главный вопрос - как научиться создавать саморазвивающиеся интеллектуальные системы. К сожалению, в этом отношении успехи сегодняшних технологий более чем скромны.

Можно вспомнить времена, когда прогресс в искусственном интеллекте связывался с прогрессом в шахматных программах; предполагалось, что по мере усиления их игры удастся обнаружить фундаментальные принципы разработки интеллектуальных систем. В наше время уже созданы исключительно мощные такие программы, способные обыгрывать чемпиона мира. Технология экспертных систем добилась впечатляющих результатов и во многих других областях. Во всех этих случаях мы имеем дело с чрезвычайно сложными узкоспециализированными алгоритмическими комплексами, зачастую разрабатывавшимися многими программистами, а иногда -

поколениями программистов. Даже если предположить, что такого рода экспертные системы будут созданы практических для всех областей интеллектуальной деятельности человека и окажутся по своей эффективности превосходящими его возможности (что пока далеко не так), можно ли будет считать, что эта галлерея экспертных систем и есть "настоящий" искусственный интеллект? Вероятно, нет, - ведь такая коллекция ничуть не ускорит научно-технический прогресс, а лишь зафиксирует его текущее состояние.

Создается впечатление, что курс на разработку высокоэффективных экспертных систем для конкретных задач загоняет проблему изучения явления саморазвития алгоритмических комплексов в тупик. Программисты сразу начинают решать все проблемы организации такого комплекса "своим умом"; отвлекаться на самоанализ им, в общем-то, некогда, да и технологии здесь никакой не создано, и когда дело доходит до завершающих этапов, вся та сложнейшая интеллектуальная работа, которой и нужно было бы научить саму проектируемую систему, остается лишь в головах ее создателей.

Каким мог бы оказаться выход из данного тупика? Видимо, исходить нужно из того факта, что программы и алгоритмы имеют своим источником теоретические знания - будем их называть, для краткости, теоремами. Поэтому ключи к проблеме саморазвития интеллектуальных систем следует искать прежде всего в пограничной зоне между теоремами и алгоритмами, используя для программирования таких систем алгоритмический язык, максимально приближенный к языку теорем.

Именно на таком языке, названном ГЕНОЛОГ'ом, и реализована база приемов предлагаемой логической системы. Этот язык, прежде всего, принципиально упрощает и ускоряет процесс обучения, делая возможным за разумные сроки осваивать сложные предметные области просто за счет количества созданных приемов. Например, так удалось выйти на сравнительно высокий уровень в вычислительных задачах по планиметрии. Разумеется, там, где имеются эффективные специальные вычислительные процедуры, использование решателя нецелесообразно - рассуждения на логическом уровне будут выполняться на порядки медленнее (хотя и существенно быстрее, чем они выполняются человеком). Однако, любые эффективные алгоритмические процедуры возникают, в конечном счете, из теорем, и для изучения этого процесса снова оказывается необходим ГЕНОЛОГ. Его компилятор предусматривает функционирование в двух режимах - как для создания эвристической алгоритмики логического уровня, так и для создания эффективных вычислительных процедур, использующих специализированные "технические" структуры данных. Использование такой компиляции делает данную систему, по крайней мере, в некоторой перспективе, вполне подходящей и для прикладных вычислений. Более того, возникает интересная новая возможность, недоступная обычным системам компьютерной математики - смешанный логико-вычислительный режим при исследовании решателем свойств каких-либо математических моделей.

Развитие в решателе способности решать не только символьные "вычислительные" задачи (системы компьютерной математики на большее и не претендуют), но и задачи "теоретического" характера позволило бы выйти в прикладном аспекте на качественно новый уровень. Ведь, по существу, сила современного математического аппарата заключается не в символьных вычислениях, а в том, что с его помощью решаются теоретические задачи, позволяющие создавать алгоритмические блоки для прикладных вычислительных процедур.

Уровень ГЕНОЛОГа не является предельно высоким при изучении истоков программирования. Логический вывод, приводящий к получению теорем, на которых

основаны приемы и алгоритмы, в значительной степени направляется потребностями расширения аппарата решения задач. В этом смысле он может рассматриваться как продолжение линии компиляции вверх от пограничного слоя между теоремами и программами. Видимо, одна из основных, если не самая важная, функция развиваемого решателя состоит именно в том, что он дает исходный материал, для прослеживания таких верхних линий компиляции. Более подробно об этом будет говориться в третьем томе данной монографии.

Но вернемся к основной цели раздела - сравнению на конкретных примерах возможностей логической системы и систем компьютерной математики. Будем рассматривать одну из наиболее мощных в плане символьных вычислений систем - Maple(v7). При сравнении нас будет интересовать не столько степень развития системы, сколько достоинства и недостатки, предопределяемые лежащими в ее основе принципами. Все рассматриваемые ниже задачи взяты из задачника решателя и, следовательно, им решаются.

4.6.1 Упрощение выражений и уравнения с одной неизвестной

Начнем с простейших примеров на упрощение алгебраических выражений. Из общих соображений, можно предположить, что здесь роль логики не слишком велика, и ответы сравниваемых систем не будут слишком сильно отличаться друг от друга. Действительно, предлагаем задачу на упрощение выражения:

$$\left(\frac{\sqrt{a}+\sqrt{b}}{\sqrt{a+b}}-\frac{\sqrt{a+b}}{\sqrt{a}+\sqrt{b}}\right)^{-2}-\left(\frac{\sqrt{a}-\sqrt{b}}{\sqrt{a+b}}-\frac{\sqrt{a+b}}{\sqrt{a}-\sqrt{b}}\right)^{-2}.$$

Решатель выдает ответ $(a+b)/\sqrt{ab}$; Maple - ответ $(a+b)/\sqrt{a}\sqrt{b}$. Ответ решателя чуть-чуть экономнее, однако различие несущественно. Преимущество решателя усматривается здесь лишь в том, что он позволяет просмотреть выкладки по шагам.

Предлагаем следующую, по виду похожую, задачу:

$$\left(\frac{\sqrt{a-b}}{\sqrt{a+b}+\sqrt{a-b}}+\frac{a-b}{\sqrt{a^2-b^2}-a+b}\right)\sqrt{\left(\frac{a^2}{b^2}-1\right)^{-1}}.$$

Решатель выдает ответ $\mathrm{sg}(b)$. Ответ системы Maple выглядит совсем по-другому:

$$-\frac{\left(\sqrt{a-b}\sqrt{a^2-b^2}+a\sqrt{a+b}-b\sqrt{a+b}\right)\sqrt{\frac{b^2}{a^2-b^2}}}{\left(\sqrt{a+b}+\sqrt{a-b}\right)\left(-\sqrt{a^2-b^2}+a-b\right)}.$$

Попытки применять функцию "simplify" с различными опциями приводят практически к такому же результату. Это уже несколько настораживает - ведь неотрицательность выражений под радикалами видна непосредственно из условий на область допустимых значений, и переход от $\sqrt{a^2-b^2}$ к $\sqrt{a-b}\sqrt{a+b}$, достаточный для получения сильных упрощений, не связан с каким-то особым логическим выводом.

Еще один несложный пример на упрощение, теперь уже связанный с логарифмами:

$$\sqrt{\log_a b+\log_b a+2}\,(\log_a b-\log_{ab} b)\,\sqrt{\log_a b}.$$

Решатель выдает ответ $(\log_a b)^2$. Ответ системы Maple почти такой же по сложности - $\ln b^2 / \ln a^2$. Однако, при уточнении выясняется, что преобразования делались лишь в предположениях $1 < a, 1 < b$, в то время как решатель рассмотрел всю область допустимых значений, существенно более широкую.

Рассматриваем далее два тригонометрических выражения -

$$\frac{\cos(-a+\pi/2)\cos(b+\pi/2) - \cos(-a+\pi)\cos(-b+2\pi)}{\sin(a+b+\pi/2)};$$

$$-\operatorname{tg}(-a+\pi/3)\operatorname{tg}(a+\pi/3)\operatorname{tg} a + \operatorname{tg}(3a) + 1.$$

В обоих случаях решатель выдает ответ 1. Maple в первом случае дает ответ:

$$\frac{\cos a \cos b - \sin a \sin b}{\cos(a+b)},$$

по каким - то причинам отказываясь усматривать равенство числителя и знаменателя. Во втором случа ответ Maple оказывается существенно более громоздким, чем исходное выражение - перейдя к синусам и косинусам, она не смогла найти каких-либо дальнейших упрощающих преобразований.

Из этой серии примеров видно, что Maple испытывает трудности трех типов. Во-первых, из-за отсутствия аккуратного логического описания контекста преобразований ("списка посылок" решателя), она оказывается во множестве случаев попросту неспособной определить, какие преобразования допустимы, а какие - нет. Это приводит к отказу от любых дальнейших преобразований и преждевременному обрыву задачи. Во-вторых, объем ее базы упрощающих приемов, видимо, не очень велик, и в сколь-нибудь нестандартных ситуациях выдается текущее "тупиковое" выражение. Наконец, из примера для дроби с косинусом суммы в знаменателе видна ее неспособность предпринимать эвристические попытки усмотрения сильных упрощений хотя бы на два шага вперед.

Хотя эти недостатки и не очень существенны для простейших задач, однако уже и здесь ответы часто выглядят неряшливо, а иногда и просто неприемлемо.

Переходим к рассмотрению уравнений с одной неизвестной. Для решения их системой Maple будем использовать функцию "solve". Рассмотрим сначала простейшее линейное уравнение с параметрами:

$$\frac{6a+7b}{6a} - \frac{3bx}{2a^2} = 1 - \frac{bx}{a^2-ba}.$$

Решатель выдает ответ $x = 7a(b-a)/3(3b-a)$, сопровождая его условиями на параметры: $a \neq 0, a-b \neq 0, 3b-a \neq 0$. Дополнительно в ответ включается вырожденный случай $b = 0, a \neq 0$, когда неизвестная x может принимать произвольное числовое значение. Система Maple ограничивается выдачей первого ответа, упуская из виду вырожденный случай и никак не комментируя ограничения на параметры.

Стандартные алгебраические уравнения - полиномиальные и сводимые к ним уравнения с радикалами Maple, разумеется, решает неплохо, если отвлечься от таких мелочей, как появление лишних корней и полное отсутствие сопровождающих ответ условий на значения параметров. Например, для уравнения

$$\frac{x+a+\sqrt{x^2-a^2}}{x+a-\sqrt{x^2-a^2}} = \frac{9(x+a)}{8a}$$

система Maple выдает правильные ответы $-29a/21, 5a/3$. Решатель в этом случае добавляет ограничение $0 < a$. Заметим, что из исходного уравнения оно не очевидно.

Несколько хуже обстоят дела с уравнениями, обращающимися в тождество на целых промежутках. Так, для уравнения

$$\sqrt{x+3-4\sqrt{x-1}}+\sqrt{x+8-6\sqrt{x-1}}=1$$

решатель дает ответ: x-число, $x \le 10, 5 \le x$. Maple прорисовывает в качестве ответа переменную x.

Чуть более сложное уравнение - и появляются "лишние" корни:

$$2\sqrt{a+x}+\sqrt{a-x}=\sqrt{\sqrt{x(a+x)}+a-x}.$$

Решатель дает ответ $x = -a, 0 \le a$, в то время как Maple, кроме значения $-a$, выдает также лишние значения 0, $64a/1025$.

Аналогичная картина для уравнения

$$\sqrt[3]{a+x^2}+2/\sqrt[3]{a+x^2}=3.$$

Правильный ответ, выдаваемый решателем, состоит из двух частей. При $1 < a, a \le 8$ имеются два корня $-\sqrt{-a+8}, \sqrt{-a+8}$. При $a \le 1$ к ним добавляются еще два корня $-\sqrt{-a+1}, \sqrt{-a+1}$. Maple просто выдает все четыре корня.

Впрочем, для многих уравнений рассматриваемого типа (как правило, без параметров) ответы Maple совершенно точны.

С логарифмическими и показательными уравнениями у Maple начинаются трудности. Например, несложное, в общем, уравнение

$$\log_{1-2x}(6x^2-5x+1)-\log_{1-3x}(4x^2-4x+1)=2$$

Maple "решает", выдавая ответ в виде чрезвычайно громоздкого выражения, содержащего так называемое "RootOf" от некоторого уравнения, которое на вид ничуть не проще исходного. Здесь Maple просто создает видимость того, что решает задачу. Решатель находит для указанного уравнения единственный корень $1/4$. Решая совершенно стандартное уравнение с параметром

$$\sqrt{\log_a\left(\sqrt[4]{ax}\right)+\log_x\left(\sqrt[4]{ax}\right)}+\sqrt{\log_a\left(\sqrt[4]{\frac{x}{a}}\right)+\log_x\left(\sqrt[4]{\frac{x}{a}}\right)}=a,$$

Maple предлагает ответ из двух десятков строк (здесь и далее имеется в виду формат экрана 640 на 480), в то время как решатель дает полный ответ

$$1 < a, x \in \{a^{1/a^2}, a^{a^2}\}.$$

Какие-то логарифмические уравнения Maple, все же, решает - например, для уравнения

$$\sqrt{\log_x\left(\sqrt{3x}\right)}\log_3 x=-1$$

находит корень $1/9$. В качестве примеров другого рода приведем два уравнения:

$$2\left|\log_{10}(ax)\right|\log_x 10=(4\log_{10}a-3)\log_{x^2}10-\frac{1}{2}\log_{10}x;$$

$$(x-3)^{3x^2-10x+3}=1.$$

В первом случае решатель дает ответ из двух частей. При $1000<a$ корнями являются числа $0.001, 0.1$; при $10\le a\le 1000$ - числа $0.1, 10^{-\sqrt{8\log_{10}a+1}+2}$. Во втором случае решатель находит ответ $x=4$. Первую задачу Maple оставляет без ответа; во втором случае она просто переписывает исходное уравнение под RootOf.

Тригонометрические уравнения обычно легко сводятся к полиномиальным, и с решением их Maple справляется неплохо. Правда, ответы выглядят неряшливо - серии корней измельчаются на множество подсерий. Например, для уравнения

$$\cos x=\left(\cos\frac{3x}{4}\right)^2$$

решатель находит три серии корней - $x=4\pi n$, $x=\pi(6n+1)/3$, $x=\pi(6n+5)/3$. Maple предлагает следующий ответ: $4\pi-8\pi m+8\pi n, 8\pi n, 4\pi-4\arccos(1/4+1/4\sqrt{2})-8\pi m+8m\arccos(1/4+1/4\sqrt{2})+8\pi n, 4\arccos(1/4+1/4\sqrt{2})-8m\arccos(1/4+1/4\sqrt{2})+8\pi n, 4\pi-4\arccos(1/4-1/4\sqrt{2})-8\pi m+8m\arccos(1/4-1/4\sqrt{2})+8\pi n, 4\arccos(1/4-1/4\sqrt{2})-8m\arccos(1/4-1/4\sqrt{2})+8\pi n$. Здесь и целочисленных параметров два вместо одного, и арккосинус не устранен. Разумеется, тригонометрические уравнения, требующие использования неравенств, Maple в общепринятом смысле не решает. Например, таковы уравнения $(\cos x)^7+(\sin x)^4=1$ или $\cos(\pi x)+x^2-6x+10=0$. Ответы выражены через RootOf, во втором случае - по существу просто переписано исходное уравнение. Показательные уравнения с тригонометрическими функциями вызывают у Maple примерно те же затруднения, что и обычные показательные уравнения. Например, для несложного уравнения

$$81^{(\sin x)^2}+81^{(\cos x)^2}=30$$

решатель находит две серии корней - $\pi(3n+1)/6$ и $\pi(3n+2)/6$. Maple предлагает в качестве ответа около десятка громоздких выражений, каждое из которых содержит мнимые единицы, радикалы и арктангенсы. Аналогичное уравнение

$$(\operatorname{tg}x)^{(\cos x)^2}=(\operatorname{ctg}x)^{\sin x}$$

Maple, по существу, вообще не решает. Для уравнений с параметром, даже если ответ и находится системой Maple, часто он оказывается неоправданно громоздким. Например, для уравнения

$$(k+1)\frac{\cos x\cos(2x-a)}{\cos(x-a)}=k\cos(2x)+1$$

решатель дает три серии корней, сопровождая каждую соответствующими ограничениями на значения параметра -

$$x=\frac{-\arcsin(k\sin a)+a-2\pi n-\pi}{2}, x=\frac{\arcsin(k\sin a)+a-2\pi n}{2}, x=\pi n.$$

Maple дает целых 8 выражений для серий, причем некоторые из них содержат до 16 вхождений тангенса. Для большей наглядности приведем одно из таких выражений:

$$\begin{aligned}&\operatorname{arctg}\left(-\sqrt{-\frac{2(\operatorname{tg}a)^2k-2-2(\operatorname{tg}a)^2-2\sqrt{1+(\operatorname{tg}a)^2-(\operatorname{tg}a)^2k^2}}{(\operatorname{tg}a)^2+1}}\times\right.\\&\times-\tfrac{1}{4}\frac{2(\operatorname{tg}a)^2k-2-2(\operatorname{tg}a)^2-2\sqrt{1+(\operatorname{tg}a)^2-(\operatorname{tg}a)^2k^2}}{(\operatorname{tg}a)^2+1}/\\&\left./\operatorname{tg}a\cdot-\tfrac{1}{2}\cdot\frac{2(\operatorname{tg}a)^2k-2-2(\operatorname{tg}a)^2-2\sqrt{1+(\operatorname{tg}a)^2-(\operatorname{tg}a)^2k^2}}{(\operatorname{tg}a)^2+1}\right).\end{aligned}$$

Приведенная выше картина, в общем, не удивительна. Уравнения элементарной математики, хотя и могут, путем грубого их сведения к чисто полиномиальным задачам и применения достаточно развитых в компьютерной алгебре алгоритмов нахождения корней полиномов, как-то быть доведены до ответа, но трудно ожидать, чтобы эти ответы имели сколь-нибудь обозримый вид. Чтобы получать действительно качественные ответы, необходимо такое разнообразие методов, которое может быть обеспечено только системой эвристического типа. Как показывают многочисленные примеры, во многих случаях ответ не достигается даже грубыми "силовыми" методами, причем это сплошь и рядом относится к уравнениям, решаемым вполне стандартными средствами элементарной математики. С другой стороны, логическая система успешно справляется с задачами стандартного типа, а при некотором доучивании оказывается способной решать и менее стандартные задачи.

В то время как для систем компьютерной математики умение решать полиномиальные задачи очень большой размерности является критическим - именно на нем и базируется вся их эффективность, для логических систем излишнее внимание к таким задачам представляется чем-то искусственным, ибо почти всегда удается обойтись вообще без них. В решатель были введены несколько приемов для разложения на множители целочисленных полиномов малых (до 7-й включительно) степеней, но случаи их использования - единичные, лишь для специально подобранных задач. Эффективная реализация на ГЕНОЛОГе, например, алгоритма Берлекэмпа, представляется интересной лишь как упражнение по развитию "вычислительной" компоненты компилятора этого языка.

Современные системы компьютерной математики, безусловно, представляют собой важный шаг в компьютерном освоении накопленного элементарной алгеброй умения решать задачи. Однако, сделан лишь первый шаг, и он создал ложное представление о том, что умение решать их, в приниципе, уже освоено такими системами. Как показывают хотя бы приведенные выше примеры, это далеко не так. Дело даже не в том, что системы Maple, Mathematica и т.п. неспособны объяснять ход решения по "шагам", разумным с точки зрения школьника. Сколь-нибудь качественные ответы они дают лишь для весьма узкого класса стандартных задач элементарной алгебры, а немного более сложные задачи, даже стандартные, не решают вовсе. В этом отношении логические системы имеют гораздо большие достижения и перспективы.

Задачи типа "найти все значения параметра, при которых уравнение имеет решения (или имеет заданное число корней)", и тому подобные задачи "с кванторами", часто встречающиеся в задачниках, логическая система решает (по крайней мере в стандартных ситуациях) без особого труда. Сравнение с Maple здесь не проводилось, так как в данной системе нет языка, на котором их можно сформулировать. По мере продвижения вперед в нашем сравнении двух подходов такая ситуация будет возникать все чаще.

Разумеется, пока мы проводим сравнение логической системы и систем компьютерной математики лишь в отношении сколь-нибудь "интеллектулаьных задач", не затрагивая численных процедур. Однако, и в этом отношении логическая система имеют хорошие перспективы. Они связаны с развитием исследований по компиляции вычислений непосредственно с уровня теорем. Некоторые уже имеющиеся ее вычислительные возможности мы обсудим ниже.

4.6.2 Системы уравнений и неравенств

Системы полиномиальных уравнений без труда решаются программами компьютерной алгебры, так как здесь существует хорошо развитый аппарат, основанный на алгоритме Бухбергера. Предлагаем несложную систему уравнений $xy + xz = x^2+2, xy+yz = y^2+3, xz+yz = z^2+4$ решателю и системе Maple. Решатель выдает ответ сразу в явном виде; Maple сначала выражает его через RootOf, и лишь после обращения к команде "allvalues" преобразует к явному виду. Точно такая же ситуация для системы уравнений с радикалами $\sqrt{x+\sqrt{y}} - \sqrt{x-\sqrt{y}} = 1, \sqrt{x^2-y} + \sqrt{x^2+y} = 1$. Решатель выдает ответ $x = 5/8, y = 3/8$ сразу; Maple сначала выражает y через RootOf, а явный ответ выдает только после ввода команды "allvalues". Впрочем, наличие мощной полиномиальной техники не спасает Maple от систем иррациональных уравнений, требующих искусственного приема. Например, система из двух уравнений

$$\begin{cases} \sqrt{\frac{20y}{x}} = \sqrt{x+y} + \sqrt{x-y} \\ \sqrt{\frac{18x}{y}} = \sqrt{x+y} - \sqrt{x-y} \end{cases}$$

Mapl'ом не решается. Логическая система, усматривая, что при перемножении уравнений неизвестные под радикалами в левых частях сокращаются, без труда получает ответ $x = 5, y = 4$. Если уравнение одно, а неизвестных две, но из экстремальных соображений они определяются однозначно (или в конечном числе вариантов), то Maple просто не понимает задачи - выражает одну неизвестную через другую и этим ограничивается. Например, так получается для уравнения $\sqrt{x+1}\sqrt{-y+1} + \sqrt{x}\sqrt{y} + x = \sqrt{2}(x+1)$. Решатель в этом случае находит единственное решение $x = \sqrt{2}+1, y = \sqrt{2}-1$, применяя аппарат отыскания экстремальных точек функции двух переменных. Впрочем, если система из нескольких уравнений с радикалами основана на экстремальных соображениях, но легко сводится к полиномиальной системе, то дальнейшее является для Maple очевидным - корни находятся, причем часто выдается большое количество дополнительных "решений" с RootOf-ами.

Системы уравнений с параметрами вызывают у Maple затруднения того же типа, что и одиночные уравнения. В ответах отсутствуют условия на параметры, при которых имеются корни; появляются "лишние" корни; выражения плохо упрощены. Например, система $x\sqrt{1-y^2} = a, y\sqrt{1-x^2} = b$ имеет при $|b^2 - a^2| \le 1$ ровно два решения. Maple выдает здесь 8 решений. Для простенькой системы

$$\begin{cases} \left(\frac{x}{a}\right)^b \left(\frac{y}{c}\right)^d = e \\ \left(\frac{x}{c}\right)^d \left(\frac{y}{a}\right)^b = f \end{cases}$$

нужно разбирать отдельно два случая: $b + d = 0$, $b + d \neq 0$, так как выражения в ответе будут различными. Maple этого не делает. Решатель находит для x выражение

$$(a^b c^d)^{\frac{1}{b+d}} \left(\frac{f^d}{e^b}\right)^{\frac{1}{d^2-b^2}},$$

в то время как Maple, даже после применения команды "simplify", то же самое записывает гораздо более громоздким образом

$$exp\left(\frac{-\ln(f)d + \ln(f)b + b\ln(e) - \ln\left(\frac{a}{c}\right)db + b^2\ln\left(\frac{a}{c}\right)}{-d^2 + db + b^2}\right)c.$$

Системы с логарифмическими и показательными уравнениями Maple начинает решать с перебоями, иногда не выдавая никакого ответа (даже с RootOf). Например, она не решает систему

$$\begin{cases} xy^{\log_x y} = y^{2/5} \\ \log_4 x \log_x(x-3y) = 1. \end{cases}$$

Систему

$$\begin{cases} (\log_{10} x)^2 = (\log_{10}(xy))^2 + (\log_{10} y)^2 \\ \log_{10} x \log_{10} y + (\log_{10}(x-y))^2 = 0 \end{cases}$$

Maple решает, хотя и присоединяет какое-то явно лишнее решение с RootOf. Похожую систему

$$\begin{cases} 2\log_{1-x}(-xy-2x+y+2) + \log_{2+y}(x^2-2x+1) = 6 \\ \log_{1-x}(y+5) - \log_{2+y}(x+4) = 1 \end{cases}$$

Maple снова не решает. Заметим, что все это - совершенно стандартные системы, легко решаемые "школьными" методами. На примере тригонометрической системы

$$\begin{cases} (\sin x)^3 = \frac{1}{2}\sin y \\ (\cos x)^3 = \frac{1}{2}\cos y \end{cases}$$

обнаруживается, что Maple может не только давать "лишние" корни, но и терять верные. Вместо четырех серий решений она здесь находит только две. Систему

$$\begin{cases} \frac{1}{\sqrt{\sin\left(\frac{\pi}{4}-x\right)}}(3\cos(2x) - 6\operatorname{ctg} y + 2) = 0 \\ 18(\sin x)^2 - 2\operatorname{tg} y - 3 = 0 \end{cases}$$

Maple не решает. Для системы

$$\begin{cases} \operatorname{ctg} x + \sin(2y) = \sin(2x) \\ 2\sin y \sin(x+y) = \cos x, \end{cases}$$

имеющей две серии корней, Maple представляет ответ в виде десяти серий. В ответах тригонометрических систем часто встречаются RootOf, хотя и легко устранимые, но все же требующие для этого обращений к специальным командам. Для системы

$$\begin{cases} \operatorname{tg} x + \operatorname{ctg} x = 2\sin\left(y - \frac{3}{4}\pi\right) \\ \operatorname{tg} y + \operatorname{ctg} y = 2\sin\left(x + \frac{\pi}{4}\right), \end{cases}$$

имеющей единственную серию решений $x = \pi/4 + 2\pi n, y = 5\pi/4 + 2\pi m$, Maple выдает пять серий решений. Выражение для y в одной из таких серий занимает пятнадцать строк. Тригонометрическая система с параметром

$$\begin{cases} \sin x \cos(2y) = a^2 + 1 \\ \cos x \sin(2y) = a \end{cases}$$

имеет решения только при $a = 0$. В этом случае она имеет две серии решений $x = \pi/2 + \pi m + 2\pi n, y = \pi m/2$ и $x = 3\pi/2 + \pi m + 2\pi n, y = \pi/2 + \pi m/2$. Maple игнорирует тот факт, что при $a \neq 0$ система решений не имеет, и предлагает ответ из 8 серий, занимающих 49 строк. Еще более "впечатляет" ответ Maple для системы

$$\begin{cases} \cos(x-2y) = a(\cos y)^3 \\ \sin(x-2y) = a(\cos y)^3. \end{cases}$$

Он занимает около 210 строк, в то время как ответ решателя помещается всего на 4 строках.

Переходя к неравенствам, обнаруживаем, что здесь легко провести границу между задачами, доступными и недоступными Maple. Неравенства без параметров Maple решает достаточно хорошо (хотя ответы временами выглядят неряшливо). Неравенств с параметрами, кроме крайне простых, Maple вообще не решает. Например, без ответа остается система из двух линейных неравенств $x - 3 \leq a(x-2), 8(a+1)x > 8ax + 9$. Как и в случае уравнений, задачи с кванторами для неравенств Maple не решает, и даже неясно, как такие задачи для нее формулировать.

Задачи на доказательство неравенств можно предлагать Maple, обращаясь к функции "verify", которая выдает "истина" или "ложь". Однако, даже очень простые неравенства приводят к отказу. Например, неравенство

$$1 \leq \frac{1}{a^2+1} + a^2.$$

Решатель обучен доказательству неравенств на обычном материале школьных задачников, и неплохо справляется со стандартными задачами такого рода.

Завершая сравнение Maple и решателя на материале элементарной алгебры, можно констатировать два факта. Во-первых, роль логических процессов для многих классов алгебраических задач не очень велика - задачи сводятся к полиномам, и применяются специальные алгоритмы для работы с последними. Это позволяет системе Maple решать или "почти решать" многие задачи. Во-вторых, остается впечатление, что логическая система справляется с задачами элементарной алгебры безусловно лучше, чем Maple. Число таких задач из стандартных задачников, решаемых логической системой, значительно больше, чем число решаемых Maple, а качество ответов - существенно выше. Разумеется, можно доводить вручную неряшливые ответы, отбрасывать неподходящие корни и проводить целые исследования, связывая найденные корни с параметрами, но лучше, все-таки, когда этим занимается сама компьютерная программа.

4.6.3 Элементарная геометрия

Логическая система располагает обширной базой приемов, предназначенной для решения школьных планиметрических задач "на вычисление". Она сформировалась в результате проработки около двух тысяч задач и насчитывает примерно четыре тысячи приемов. При обучении не делалось каких-либо исключений - задачи брались из задачников подряд, и для каждой, если она не решалась сразу, создавались недостающие приемы. Фильтры приемов формулировались из эвристических соображений, чтобы сохранить лишь минимально необходимые для получения ответа действия. Такая стратегия предопределяла итеративные приближения к границе, отделяющей полную блокировку инициативы системы от неконтролируемых лавинообразных срабатываний приемов. Где-то на этой границе и начало проявляться более-менее разумное поведение решателя, иногда вполне сопоставимое по своей целенаправленности с поведением школьника. Ход решения можно прослеживать по шагам, и даже получать на экране текст-формульные пояснения этих шагов. При обучении не ставилось задачи буквально воспроизводить поведение человека. Хотелось просто обеспечить наискорейшее получение ответа, так что в ряде случаев отдавалось предпочтение режимам, выгодным с точки зрения компьютерной реализации,

но мало приемлемым для человека. Вообще, предлагаемая версия "геометрии" является сугубо экспериментальной и обладает немалым числом недостатков. Анализ и исправление их могли бы привести к созданию новых, существенно более мощных версий.

Однако, даже и в данном варианте, система может делать довольно много. Удалось достичь сравнительно устойчивого решения ею стандартных задач средней степени сложности. Примером может служить следующая задача, которая была решена системой самостоятельно на одной из демонстраций: "В некотором треугольнике ABC биссектриса, медиана и высота делят угол A на четыре равные части. Найти этот угол". Система получила правильный ответ $\pi/2$ после 30-40 секунд. Анализ и коррекция ее действий позволили уменьшить время решения до 8 секунд. Единственное, что было изменено - несколько ослаблен фильтр приема, принимающий решение о выписывании соотношения для отрезков, на которые биссектриса делит противоположную сторону.

Кроме обучения решению планиметрических задач, рассматривалось также самостоятельное построение системой эскизов, соответствующих условию задачи. Такой решатель для "чертежей" был создан, и работает вполне стабильно. Он состоит из базы приемов, формирующих некоторую схему вычислений для итеративной оптимизации параметров чертежа, а также из интерпретатора, выполняющего вычисления по данной схеме. Последние осуществляются с помощью математического сопроцессора, так что весь цикл построения эскиза происходит быстро - обычно не более секунды. Если задача не имеет параметров, то построение эскиза (кроме особо сложных случаев) эквивалентно численному ее решению. С общей точки зрения, разработка процедуры построения чертежей оказалась интересной в первую очередь как пример использования логических процессов для создания программы вычислений. Возвращаясь к приведенному выше примеру планиметрической задачи, заметим, что система успешно справилась с построением чертежа и нарисовала прямоугольный треугольник.

Кроме вычислительных задач по планиметрии, были рассмотрены также планиметрические задачи на доказательство и на построение. Однако, эти задачи требуют самостоятельной проработки, а пока на малом количестве примеров были определены лишь общие особенности логических режимов их решения. Это же замечание относится к задачам по стереометрии. Здесь открыто широкое поле деятельности, освоение которого могло бы привести к значительному развитию геометрического решателя системы.

Примерно так понимается проработка элементарной геометрии в логической системе. Перейдем к рассмотрению того, как этот раздел представлен в системе Maple.

Для работы с геометрическими объектами Maple имеет так называемые пакеты расширений - "geometry" и "geom3d", обслуживающие, соответственно, двумерный и трехмерный случаи. Эти пакеты содержат большое количество функций, позволяющих определять различные геометрические фигуры и тела, находить их характеристики и рисовать на экране. Например, можно ввести в рассмотрение окружность, проходящую через 3 заданные точки A, B, C с заданными координатами, обозначив саму окружность $c1$, а ее центр $O1$. Для этого вводится команда "circle(c1, [point$(A,0,0)$, point$(B,2,0)$, point$(C,1,2)$],'centername' = O1)". Команда "radius(c1)" позволяет найти радиус данной окружности; команда "Equation(c1)" - получить ее уравнение; команда "draw(c1)" - нарисовать на экране, и т.п. Таким образом, Maple просто предоставляет пользователю некоторый язык программирования для работы с геометрическими объектами. Какой-либо геометрической функции, аналогичной

"алгебраической" функции "solve", у Maple не предусмотрено. Обычные школьные задачи по планиметрии (кроме совсем простых, сводящихся к применению функций указанного выше типа) нельзя даже сформулировать на языке системы Maple.

4.6.4 Аналитическая геометрия

В логической системе имеется база приемов по аналитической геометрии. Обучающим материалом для ее создания послужили задачи из классического сборника П.С.Моденова и А.С.Пархоменко. Всего было проработано около 770 задач, из которых извлечено примерно 2400 приемов. Рассматривались, в основном, задачи не теоретического характера, практически по всем основным разделам - векторной алгебре, системам координат, уравнениям прямой и плоскости, линиям и поверхностям второго порядка. В большинстве этих задач, как и в задачах по элементарной геометрии, требовалось восстановить неизвестные параметры какой-либо группы объектов по частичному их заданию. Поэтому стратегия обучения и логические режимы здесь оказались похожими. В качестве примера приведем задачу "Написать уравнение плоскости, параллельной плоскости Oyz и пересекающей однополостный гиперболоид

$$\frac{x^2}{9}+\frac{y^2}{4}-z^2=1$$

по гиперболе, действительная полуось которой равна 1". Хотя и не очень сложная, она все-таки не сводится к определению какой-то стандартной характеристики полностью заданной поверхности или кривой. С другой стороны, Maple, как и в случае элементарной геометрии, может предложить лишь набор функций для работы с полностью определенными кривыми и поверхностями. Эти функции даже не отделены от используемых в элементарной геометрии и содержатся в уже упоминавшихся выше пакетах "geometry" и "geom3d". К ним примыкает пакет "algcurves" для алгебраических кривых. Как и в предыдущем разделе, констатируем, что сколь-нибудь нетривиальные в логическом отношении, и притом вполне стандартные, задачи по аналитической геометрии нельзя даже сформулировать на языке системы Maple. Например, такой является приведенная выше задача.

4.6.5 Алгебра логики, алгебра множеств и комбинаторика

Решатель имеет приемы для решения несложных задач по алгебре логики и алгебре множеств. Они позволяют упрощать формулы и решать уравнения. Например, можно предложить решателю систему уравнений для множеств $x\cup(y\backslash a)=(x\backslash b)\cup(b\cap y),(x\backslash y)\cup(y\backslash x)=c$ и получить ответ $x=c\triangle y, y\cap(c\backslash(a\triangle b))=\emptyset, b\cap c\subseteq y, y-\text{set}$. Здесь допустимые значения неизвестной y определяются как множества, содержащие одно фиксированное множество и не пересекающиеся с другим. Неизвестная x однозначно определяется по y. Можно предложить решателю для упрощения выражение $a\setminus(b\cup c)\cup(a\cap d)\setminus b$ и получить ответ $a\setminus(b\cup(c\setminus d))$.

В Maple решение аналогичных типов задач не предусмотрено, хотя для операций алгебры множеств и имеются соответствующие функции. Эти функции, как и у геометрических пакетов, ориентированы лишь на выполнение действий с конкретными множествами. Maple отказывается упростить даже выражение $a\setminus(a\cap b)$.

В решателе создана также база приемов для задач по комбинаторике. Общее количество проработанных задач и созданных по ним приемов не очень велико - всего около сотни задач и двух сотен приемов. Однако, они уже позволяют сравнительно

стабильно решать стандартные комбинаторные задачи средней сложности. Часть обучающего материала предоставили школьные задачники, часть - задачники по теории вероятностей. Чтобы задача была понятна решателю, она должна быть переведена на теоретико-множественный язык. Например, имеется задача "Из вазы, где стоят 10 красных и 4 розовых гвоздики, выбирают один красный и два розовых цветка. Сколькими способами это можно сделать?". Для ее формализации вводим обозначения A, B множеств красных и розовых гвоздик. В посылки задачи заносим следующие данные об этих множествах: $\mathrm{card}A = 10; \mathrm{card}B = 4; A \cap B = \emptyset$. Затем вводим задачу на упрощение выражения $card(\mathrm{set}_x(x - \mathrm{set}\ \&\ x \subseteq A \cup B\ \&\ \mathrm{card}(x \cap A) = 1\ \&\ \mathrm{card}(x \cap B) = 2))$. При такой постановке решатель выдает ответ 60. Многие разделы математики и физика предлагают задачи только текстового характера. Чтобы работать с ними в логической системе, приходится много внимания уделять вопросам логической формализации. Здесь происходит встречное движение: с одной стороны, начато обучение решателя чтению и пониманию текстов естественного языка. С другой стороны, развивается тот формальный язык, на котором должен будет задаваться смысл этих текстов. Промежуточные между естественным и математическим языком этапы требуют работы с расширенным спектром понятий, таких, как встречающиеся в приведенной выше задаче понятия "гвоздика", "цветок", "красный" и т.п. По этой причине, предстоит развитие решателя, обеспечивающего преобразование задачи с полуформального логического языка промежуточного уровня на уровень завершающей математической формализации. Такую работу нужно будет проделать по многим уже рассматривавшимся в логической системе разделам - комбинаторике, геометрии, теории вероятностей, элементарной физике и т.п.

В системе Maple каких-либо средств для формулировки комбинаторных задач указанного выше типа не обнаружено.

Наконец, упомянем задачи на преобразование выражений, связанных с комбинаторными функциями. При обучении решателя в этой области были созданы около трех сотен приемов, позволивших решать стандартные школьные задачи. Maple тоже хорошо справляется с такими задачами. Особенно это видно на примерах, связанных с вычислением конечных сумм. Однако, для различных задач пользователю Maple приходится привлекать различные функции, в то время как у решателя везде работает целевая установка "упростить выражение". В некоторых случаях Maple, несмотря на использование своей мощной техники, все же не очень удачно преобразует сравнительно стандартные суммы, например:

$$\sum_{k=0}^{[n/4]} C_n^{4k},$$

$$\sum_{i=1}^{n} (\sin(3i+1))^2.$$

Ответы выдаются чрезмерно громоздкие, хотя обе суммы представимы несложными тригонометрическим выражениям. Вычисление кратных сумм предусмотрено в Maple только для случая, когда они сразу вводятся как повторные. Если область суммирования определяется какими-то более сложными условиями, то такую задачу нельзя даже сформулировать. Решателю же часто приходилось работать с такими суммами, например, в задачах по теории вероятностей.

4.6.6 Математический анализ

Задачи на символьное вычисление производных, пределов и интегралов в большинстве случаев решаются с помощью стандартных вычислительных процедур. Естественно ожидать, что Maple с ними будет легко справляться. Однако, в действительности задачник по математическому анализу значительно шире - кроме чисто вычислительных примеров, в нем встречается большое количество задач теоретического характера, а также задач, пусть и не претендующих на роль теоретических, но все же требующих определенных рассуждений. В этих случаях, столь же естественно, Maple оказывается бессилен. Продвижение в указанной области может дать только развитие логической системы.

Само по себе вычисление производных не составляет какого-либо труда ни для решателя, ни для Maple. Трудности начинаются на этапе упрощения результата дифференцирования; сравнение двух систем для задач последнего типа уже проводилось выше. В решателе, как и в Maple, предусмотрена возможность дифференцировать функции, заданные неявно.

Пределы без параметров вычисляются обеими системами хорошо; для громоздких выражений здесь у Maple имеется неоспоримое преимущество в скорости. Однако, в случае выражений с параметрами Maple начинает давать отказы, даже там, где получение ответа не связано с разбором случаев.

Приемы решателя для вычисления неопределенных и определенных интегралов были созданы при проработке задачника Б.П.Демидовича. Некоторой неожиданностью оказалось то, что Maple не берет части этих интегралов, например, интеграла

$$\int \frac{\arcsin x(1+x^2)}{x^2\sqrt{1+x^2}}dx.$$

Если подынтегральное выражение зависит от параметров, причем в вырожденных случаях результат интегрирования определяется особым выражением, то решатель разбирает все случаи, а Maple выдает лишь общий случай. Упрощение результатов интегрирования, даже после применения функции simplify, часто оставляет у Maple желать лучшего. Вычисляя интеграл

$$\int_0^{\pi/2} \frac{1}{a^2(\sin x)^2 + b^2(\cos x)^2}dx,$$

равный $\pi/2|ab|$, Maple дает громоздкий ответ на 7 строках.

В целом, анализ того, как Maple вычисляет производные, пределы и интегралы, не оставляет впечатления, что эта тема закрыта для дальнейшего совершенствования. Время от времени попадаются "неудобные" для Maple примеры, где ответ либо не находится, либо является неоправданно громоздким, неполным и т.п. Надо сказать, что приемы решателя, обеспечивающие аналогичные вычисления, собраны в нескольких крупных блоках (пакетах), функционирующих автономно, вне сканирования задачи. Это резко отличается от ситуации в других разделах и говорит о том, что роль логики при решении указанных задач не очень велика. Принципиальных преимуществ в данной области (кроме, быть может, задач с параметрами) решатель не имеет, хотя и не сильно уступает Maple. Впрочем, ход вычислений решателя ближе к привычному для человека, и его пошаговые пояснения могут в этом плане представлять определенную ценность.

Обучение решателя было продолжено на такие разделы, как вычисление двойных интегралов и нахождение с их помощью площадей, объемов и площадей поверхностей; вычисление длины кривой и криволинейных интегралов. Были проработаны

задачи из соответствующих разделов задачника Б.П.Демидовича. Логическая составляющая здесь уже становится заметной, так как кривые и области задаются с помощью систем утверждений. Видимо, по этой причине у Maple данные разделы развиты слабо или вообще не развиты. Например, вычисление кратных интегралов там предлагается вручную сводить к последовательному вычислению однократных.

Следующий раздел - нахождение экстремумов, максимумов и минимумов. Решатель применяет здесь весь свой аппарат решения уравнений и неравенств, подключая к нему возможности доказательства неравенств с помощью производной (например, для установления отсутствия корней производной на некотором промежутке). Хотя и несколько медленно, он, как правило, получает вполне аккуратные ответы. В них разобраны все подслучаи и отброшены лишние значения. Поведение Maple прокомментируем на примере использования функции "extrema". Предлагаем найти экстремумы функции $x(x-1)^2(x-2)^3$ и обнаруживаем, что кроме "настоящих" экстремумов в точках $1, (5-\sqrt{13})/6, (5+\sqrt{13})/6)$, выдается также точка перегиба 2. В качестве экстремумов синуса почему-то выдается единственная точка $\pi/2$. Для функции $\cos(2x)/2 + \cos(x)$ верно выдаются точки экстремума (с точностью до периода 2π), но почему-то пропущено одно из экстремальных значений для этих точек. Для функции

$$\ln(\frac{x^2-3x+2}{x^2+1})$$

еще хуже - в качестве одной из точек экстремума выдается значение $(1+\sqrt{10})/3$, расположенное вне области определения (выражение под логарифмом отрицательно). Каких-либо пояснений относительно того, где имеем точку локального максимума, а где - локального минимума, не выдается. Все это никак не укладывается в представление о математической аккуратности.

Для решателя проводилось также обучение задачам на общий "качественный" анализ поведения функции с помощью производных и пределов. По заданной функции предпринимаются попытки найти ее область определения, интервалы монотонности, экстремумы, определить число корней на участках монотонности, исследовать функцию на выпуклость-вогнутость. Рассматривались задачи на определение промежутков непрерывности с классификацией точек разрыва и на анализ равномерной непрерывности. Хотя у Maple и существует функция "discont", предназначенная для поиска точек разрыва, она не дает никакой их классификации. Например, для функции $x/\sin(x)$ просто выдается серия πn, хотя в этой серии, кроме точек разрыва второго рода, есть и точка устранимого разрыва 0.

Стандартные задачи на исследование сходимости рядов также были проработаны при обучении решателя. Например, можно предложить ему задачу на определение всех таких p, q, при которых сходится ряд

$$\sum_{i=1}^{\infty}\left(\prod_{j=0}^{i-1}(p+j)/(i!i^q)\right)$$

и получить ответ $p < q$. Система Maple не предусматривает решения задач такого типа. Разложение функций в ряды Тейлора и Фурье - еще один тип стандартных задач по математическому анализу. Функции системы Maple находят заданное конечное число членов разложения в степенные ряды. По каким-то причинам, ряды Фурье при этом не рассматриваются. Решатель, кроме того, обучался на примерах, где нужно получить разложение в виде бесконечной суммы. Например, при разложении

функции

$$\ln\left(\sqrt{\frac{x+1}{1-x}}\right)$$

в окрестности нуля он дает ответ

$$\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}.$$

Maple не выполняет таких разложений ни для степенных, ни для тригонометрических рядов. Впрочем, зато она неплохо решает обратную задачу - суммирование бесконечных рядов. Решатель также обучен этой задаче, в пределах стандартных задачников по математическому анализу.

4.6.7 Дифференциальные уравнения

Решатель имеет базу приемов для решения обыкновенных дифференциальных уравнений и систем таких уравнений. Обучение предпринималось по известному задачнику А.Ф.Филиппова. Всего было проработано около 550 задач, и по ним создано около 450 приемов. Хотя основные общие приемы и были заложены в решатель, его коллекция, видимо, не очень полна, и обучение можно продолжить по многочисленным справочникам, извлекая из них уже более специальные приемы. В системах компьютерной математики решению дифференциальных уравнений уделялось самое серьезное внимание, и в целом они располагают здесь весьма мощным аппаратом. Конечно, не на все задачи из стандартных задаников они дают "хорошие" ответы. Иногда получаются в десятки раз более громоздкие формулы. Другой, более неприятный недостаток - нередко теряются особые решения. Впрочем, это неизбежное следствие пренебрежения логикой. Данные замечания проиллюстрируем на единственном примере. В задаче

$$b\left(\frac{dy(x)}{dx}\right)^3+\frac{dy(x)}{dx}=y(x)$$

Maple и теряет особое решение $y(x)=0$, и получает крайне громоздкое выражение с квадратурами на семи строках, в то время как ответ в параметрической форме выглядит несложно: $y=t(bt^2+1); x=3bt^2/2+\ln(Ct)$. Несмотря на ограниченный объем обучающего матриала, решатель довольно стабильно решает стандартные примеры, значительно аккуратнее обращаясь с особыми случаями. При этом он немного уступает (хотя и не всегда) системе Maple по времени решения.

4.6.8 Теория вероятностей

Начато обучение решателя задачам по теории вероятностей. Источником обучающего материала послужила книга "Е.С.Вентцель, Л.А.Овчаров. Задачи и упражнения по теории вероятностей". Текстовые условия задач переводились на формальный логический язык вручную; одновременно происходило развитие языка. В настоящее время проработана примерно половина задачника - первые 5 глав. Сюда вошли: задачи на непосредственный подсчет вероятностей; теоремы сложения и умножения вероятностей; формула полной вероятности и формула Бейеса; повторение опытов; случайные величины (скалярные), законы их распределения и числовые характеристики. Первые задачи фактически продолжают комбинаторику. В качестве примера

задачи собственно по теории вероятностей рассмотрим следующую: "Прибор состоит из двух узлов. Работа каждого узла безусловно необходима для работы прибора в целом. Надежность (вероятность безотказной работы) в течение времени t первого узла равна p_1, второго p_2. Прибор испытывался в течение времени t, в результате чего обнаружено, что он вышел из строя (отказал). Найти вероятность того, что отказал только первый узел, а второй исправен". Чтобы сформулировать эту задачу на логическом языке, обозначим рассматриваемое вероятностное пространство C. События, соответствующие безотказной работе первого и второго узлов в течение времени t, обозначим A, B. Тогда имеем посылки: "вероятность(A,C)=p_1"; "вероятность(B,C)=p_2". К ним добавляем подразумеваемое по умолчанию условие статистической независимости событий A, B - "незавсобытия((A, B),C)". После этого остается добавить условие задачи "x = услвероятн($B \setminus A$, элемсобытия(C)\($A \cap B$),C)". Переменная x объявляется неизвестной задачи; параметры p_1, p_2 - известными, и далее решатель реализует ту же логическую схему действий, которая применялась для геометрических задач на вычисление. После несложных выкладок находится ответ

$$x = \frac{p_2(1-p_1)}{1-p_1p_2}.$$

Конечно, применение компьютерной системы для решения задач, подобных данной, требует от пользователя хорошего владения логическим "словарем" и определенного привыкания к базе приемов, его использующей. Возможно, эту работу можно будет упростить за счет каких-то вспомогательных средств интерфейсного порядка (схем, меню, подсказок и т.п.). Последнее могло бы оказаться даже более перспективным, чем обучение решателя пониманию самого текста задачи, ибо ввод текста тоже весьма трудоемок.

Приведем еще один пример задачи, доступной решателю: "Случайная величина X подчинена закону Лапласа $f(x) = a\exp(-\lambda|x|$ $(\lambda > 0)$. а) Найти коэффициент a; б) построить графики плотности распределения и функции распределения; в) найти математическое ожидание и дисперсию величины X". Формализация посылок задачи выглядит так: "случвеличина(X,A)" (A - вероятностное пространство); "плотнраспред(X,A) = $\lambda_x(a\exp(-n|x|), x -$ число"; "$0 < n$". Условия задачи суть: $k = a$; "$f =$ функраспред(X,A)"; "$m =$ матожидание(X,A)"; "$D =$ дисперсия(X,A)". Здесь неизвестными являются k, f, m, D. Ответы, полученные решателем, таковы: $D = 2/n^2, m = 0, k = n/2$,

$$f = \lambda_x\left(\left(-\frac{1}{2\exp(nx)} + 1\text{при}0 \le x,\text{иначе}\frac{\exp(nx)}{2}\right), x - \text{число}\right).$$

Система Maple имеет пакет для статистической обработки данных, однако задач, подобных приведенным выше, она не решает, и даже не предусматривает языка для их формулировки.

4.6.9 Линейная алгебра

Решатель имеет приемы для выполнений элементарных действий с матрицами, решения простейших матричных уравнений, нахождения ранга матрицы, вычисления определителей, нахождения собственных векторов и собственных значений, а также для приведения матрицы к жордановой форме. Все эти задачи имеют явно вычислительный характер (хотя и в символьном представлении) и обычно ставятся для

конкретных числовых матриц фиксированного порядка. Разумеется, Maple такие задачи тоже решает. Кроме того, была предпринята попытка обучить решатель вычислению определителей n-го порядка. Рассмотрено два-три десятка таких задач, из которых извлечены необходимые для их решения приемы. Даже формулировка этих задач оказалась достаточно громоздкой - для описания вида матрицы используются конструкции со вложенными условными выражениями. Однако, какого-либо принципиального тупика здесь не возникло, и при необходимости проработка таких задач может быть продолжена. Этого класса задач Maple не решает. Сложилось впечатление, что с точки зрения изучения логических процессов наиболее интересны не вычислительные задачи линейной алгебры, а теоретические задачи, проработка которых планируется в будущем. Однако, важным достижением представляется компиляция вычислительных процедур для перечисленных выше типов задач непосредственно с теоремного уровня. Это позволит перейти к такому развитию компилятора, которое обеспечит трансформацию тех же самых теорем уже не в символьные, а в численные процедуры.

4.6.10 Комплексный анализ

Для работы с комплексными числами в решателе введены обозначения операций и функций, отличающиеся от обозначений вещественнозначного случая. Таким образом, решение задач в вещественных числах выполняется независимо от решения их в числах комплексных. При отсутствии вещественных решений уравнения, про которое явно указано, что оно вещественнозначное, никаких комплексных значений выдаваться не будет. Чтобы задача была воспринята решателем как комплекснозначная, в посылках явно задается тип переменных "комплексное(...)". Формульный редактор прорисовывает вещественнозначные операции и их комплексные аналоги одинаково; различие имеется лишь во внутреннем логическом представлении.

Обучение решателя комплексному анализу выполнялось по сравнительно простым задачникам и имеет предварительный характер, хотя и охватывает основные классы задач. Кроме простейших задач на преобразование выражений и решение уравнений, рассматривались задачи, связанные с подмножествами комплексной плоскости: переход от геометрического описания к описанию с помощью комплексных чисел и обратно, нахождение образов и прообразов при отображениях, определение области аналитичности функции и нахождение аналитической функции, отображающей одну область на другую. Рассматривались также задачи на восстановление аналитической функции по ее вещественной либо мнимой компоненте, на вычисление интегралов, определение области сходимости ряда, и т.д. Maple решает задачи лишь некоторых из перечисленных типов. Например, она не решает задач на построение конформного отображения, переводящего одну область в другую, не вычисляет интегралов по замкнутому контуру или (для неаналитических функций) вдоль заданной кривой, не определяет области сходимости рядов и т.п. Видимо, это связано с тем, что перечисленные задачи требуют не только преобразования выражений, но и достаточно интенсивной работы с утверждениями, описывающими текущий контекст.

4.6.11 Элементарная физика

Естественным этапом развития логической системы, уже обученной решению стандартных математических задач из различных разделов, является попытка использовать этот потенциал в смежных областях, и в первую очередь - в физике. Спектр

возможных приложений компьютерной системы, способной самостоятельно создавать математические модели по качественным описаниям объектов и исследовать эти модели, значительно шире, чем у обычной системы компьютерной математики. Однако, такое расширение предполагает уже весьма интенсивное использование логических процессов. Maple и подобные ей системы не имеют каких-либо функций для решения школьных задач элементарной физики. Хотя существуют разнообразные компьютерные системы для выполнения физических расчетов, они тоже не решают проблемы логической формализации и моделирования решения задач хотя бы на уровне школьных задачников по физике. Недостатки этих систем аналогичны уже перечисленным выше недостаткам систем компьютерной математики.

Обучение логической системы решению задач по элементарной физике пока весьма далеко от завершения. Обучающий материал, в основном, берется из задачника "А.И.Черноуцан. Физика, задачи с ответами и решениями". Проработаны разделы "Кинематика", "Динамика", "Закон сохранения импульса", а также (частично) "Работа и энергия". Рассмотрены примерно 450 задач, по которым созданы более двух тысяч приемов. Основную трудность представляет выбор оптимальной, с точки зрения логических вычислений, формализации понятий, используемых при описании структур и процессов. Чтобы пояснить, как предлагаемая формализация связана с исходными текстами задач, эти тексты сохраняются и могут быть вызваны на экран при просмотре формализованной версии из задачника решателя.

В физических задачах многое подразумевается "по умолчанию". Переход от сокращенной формы представления задачи к полной ее форме пока выполняется вручную. Полная формулировка оказывается, как правило, достаточно громоздкой, что создает дополнительные трудности. Причиной отказа от решения задачи, даже подобной уже рассмотренным ранее, может быть случайный пропуск какой-либо мелкой подробности или неадекватное использование имеющихся понятий. Видимо, преодоление этой трудности потребует, с одной стороны, создания приемов, способных переходить от сокращенной формулировки задачи к полной, а с другой стороны, привлечения специальных средств интерфейса - схем, меню и т.п.

С точки зрения сложности эвристических решающих правил, используемых приемами, физические задачи особых трудностей пока не предоставили. Новизна задачи обычно проявляется не в каких-то новых особенностях прорабатываемых физических понятий, а в новых типах элементов той конструкции, которая рассматривается. Примером, иллюстрирующим эту ситуацию, может служить появление задачи про "струю воды, бьющую из шланга" в разделе, посвященном броску под углом к горизонту. Разумеется, при проработке такой задачи сразу возникает множество новых понятий (струя, поток, русло, течение, и т.п.) и множество связанных с ними приемов. Это обстоятельство, с одной стороны, придает процессу обучения определенную бессистемность, а с другой стороны, очень сильно замедляет обучение. Тем не менее, проработка в решателе полного курса школьной физики за сравнительно разумный срок представляется вполне реальной. Версия, которая при этом возникнет, будет, видимо, достаточно грубой, но она может послужить хорошим организующим началом для следующего, более глубокого цикла проработки тех же разделов.

4.6.12 Вычисления

Для задач некоторых классов удается выделить сравнительно небольшую группу приемов, связать их между собой в определенную схему, указывающую очередность срабатываний, и получить таким образом алгоритм, достаточный для решения любой

задачи класса. Применение его, в отличие от общего случая решения задач, обычно называется вычислением, хотя грань, отделяющая одно от другого, весьма расплывчата.

В принципе, вычисления можно выполнять непосредственно в логических структурах данных, используя приемы решателя. Однако, такой подход приводит к очень плохим результатам - по сравнению с обычными программами время получения ответа увеличивается в сотни или даже в тысячи раз. Поэтому логическая система должна использовать для вычислений традиционные "технические" структуры данных и приспособленные к ним программы. Она может брать эти программы из своей вычислительной библиотеки либо создавать их по мере надобности, опираясь на аппарат решения задач.

Таким образом, возникают два направления исследований. Первое заключается в развитии ГЕНОЛОГа и его компилятора, которое позволило бы задавать вычислительные процедуры непосредственно на уровне языка теорем. Запасы этих процедур и составили бы вычислительную библиотеку логической системы. Так как все равно источником любой программы служит некоторая совокупность знаний предметного уровня (в нашей терминологии - теорем), то данное направление означает лишь вполне естественную попытку автоматизации верхних уровней "обычного" программирования.

Второе направление состоит в обучении решателя стандартным приемам разработки алгоритмов по накопленному здесь достаточно обширному обучающему материалу (например, по монографиям, посвященным методам вычислений). Отличие от первого направления состоит в том, что передаваемые компилятору ГЕНОЛОГа "теоремы" (логическое описание схемы вычислений) берутся не из базы теорем, а из ответа задачи на программирование. При решении ее исходные условия, связывающие входные и выходные данные программы, задаются на обычном математическом языке. Решатель преобразует их, применяя различные стандартные схемы вычислений (метод Ньютона для итерационного уточнения корней; формулу Симпсона для интегирования; метод Рунге-Кутта для решения дифференциальных уравнений и т.п.). Результатом становятся условия, определяющие схему вычислений, которая может быть непосредственно откомпилирована. В процессе решения возможны различные дополнительные преобразования условий, направленные на повышение качества создаваемой программы.

В логической системе начата проработка обоих указанных выше направлений. Вычислительные возможности ГЕНОЛОГа и его компилятора описываются в последующих разделах. Результатом компиляции служат программы ЛОСа, быстродействие которых для использования логической системы как системы компьютерной математики, видимо, пока вполне достаточно. Наиболее часто применяемые вычислительные операции вынесены в базисные операторы ЛОСа. При необходимости, компилятор ГЕНОЛОГа можно было бы ориентировать и на получение СИ-программ. Однако, пока использование ЛОСа представляется более удобным из соображений совместного использования вычислительных и логических процедур.

Что касается обучения решателя программированию, то здесь освоены следующие простейшие типы задач: программирование составления таблицы значений функции на заданном интервале; вычисление определенных интегралов; численное решение одного или системы дифференциальных уравнений первого порядка; нахождение корней системы уравнений. Исходное условие задачи может содержать параметры, которые являются входными данными для программируемой процедуры. Это условие может вообще быть неявным (например, требовать предварительного реше-

ния физической задачи для последующего программирования на основе полученных формул). В результате решения возникает группа утверждений, которые сразу же преобразуются компилятором ГЕНОЛОГа в программу. Если имеются параметры, обеспечивается вход в интерфейс для задания их конкретных значений. Полученная программа сохраняется в блоке программ и может быть использована как отдельный вычислительный модуль.

Системы компьютерной математики предоставляют обширные возможности для составления новых программ на основе имеющейся у них библиотеки стандартных функций. Однако, такое составление предпринимается вручную. Функции системы могут лишь сделать этот процесс менее трудоемким - например, продифференцировать и упростить какие-то вспомогательные выражения, включаемые в программу.

Составление новых программ в математике и приложениях обычно бывает основано на предварительном решении таких задач, которые носят скорее теоретический, чем "символьно-вычислительный" характер. Эти задачи недоступны обычным системам компьютерной математики. Существующая версия логической системы тоже пока недостаточно развита, чтобы эффективно решать задачи этого типа. Однако, приемы решения данных задач мало чем отличаются от других приемов, уже имеющихся в системе. Обучение решателя теоретическим задачам проводится, и хотя оно оказалось несколько более трудоемким, чем обучение задачам на символьные вычисления, никаких приниципиальных ограничений, препятствующих постепенному выходу на требуемый уровень, здесь не видно. Разумеется, существенным усилителем решателя теоретических задач являются процедуры развития базы теорем и атоматического синтеза приемов по теоремам, речь о которых пойдет в третьем томе данной монографии.

4.6.13 Понимание текстов

Переходим к разделам, освоение которых является естественным для логической системы, но которые, по вполне понятным причинам, отстуствуют в системах компьютерной математики. В первую очередь, выделим проблему понимания текстов. Вряд ли можно всерьез рассчитывать, что без использования решателей будет создана система, способная по-настоящему хорошо понимать смысл текстов. Очевидно, такая система должна будет постоянно решать задачи - восстанавливая опускаемые "по умолчанию" компоненты текста, устраняя различные неоднозначности или смысловые искажения, используя извлеченную из текста информацию для ответов на возникающие при диалоге с ней вопросы, и т.п. Поэтому, развивая технику решателей, следовало позаботиться о том, чтобы определить способы "подключения" их к анализу естественноязычных текстов. Эта работа была начата, и проводится на материале текстовых задач по элементарной математике.

Реализованы процедуры, осуществляющие техническую предобработку - чтение слов, разбиение их на стандартные компоненты (корни, суффиксы, окончания) и снабжение морфологическими характеристиками. Следующий этап, промежуточный между технической и логической обработкой - синтаксический анализ фразы, заключающийся в разбиении ее на подфразы и нахождении подчинений слов друг другу. Далее - этап отождествления объектов, на которые происходят ссылки из различных мест одной и той же фразы, а также объектов, упоминавшихся в ранее прочитанных фразах задачи. Наконец, этап перехода от фразы к ее логическому "подстрочнику" - крайне грубой, но уже представленной в виде списка утверждений логического языка переформулировке смысла фразы.

Лишь после этого начинается работа решателя. Первая задача, которую он должен выполнить - перейти от неформализованного, с его точки зрения, подстрочника к списку утверждений, передающему тот же смысл, но уже в терминах аккуратной логической формализации, понятной решателю. Этот этап и представляет собой, собственно, семантический анализ фразы. По завершении его возникает результат математической переформулировки исходного текста задачи, который и решается. Полученный в виде математических формул ответ переводится обратно на естественный язык, с использованием сохранившейся информации о названиях объектов, упоминаемых в задаче.

Приведенный выше цикл обработки задачи был прослежен примерно на 50 текстовых задачах. В качестве примера приведем одну из них: "После встречи двух пароходов один из них пошел на юг, а другой на запад. Через два часа после встречи расстояние между ними было 60 км. Найти скорость каждого парохода, если известно, что скорость одного из них была на 6 км/час больше скорости второго". Ответ, получаемый решателем, имеет вид: "Скорость первого парохода равна 24 км/час, скорость второго парохода равна 18 км/час".

Цепочка обработки текстовой задачи до момента получения ее математической формулировки является чрезвычайно сложной. Поэтому обучение пока продвигается медленно, хотя приемы синтаксического анализа, извлеченные всего лишь из 50 задач, уже начинают самостоятельно выполнять значительную часть своего этапа обработки текста. Общая архитектура процесса анализа текста, после ряда коррекций, стабилизировалась, однако для ускорения обучения в целом, видимо, понадобятся дополнительные технологические решения.

4.6.14 Анализ изображений

Понимание изображений, как и понимание чего-либо вообще, очевидно, требует использования логических структур данных, фиксирующих смысл увиденного. Чтобы работать с такими структурами данных, а в особенности чтобы их получать, необходимы решатели. Хорошо известно, что во многих, если не в большинстве случаев человек распознает изображение лишь в контексте априори известного ему плана окружающей обстановки. Без этой априорной логической информации, на основе одних только зрительных данных, распознавание оказывалось бы в принципе невозможным. Видимо, логическое представление изображений открывает новые перспективы и для задач их сжатия. Оно позволяет отбрасывать множество несущественных деталей, и при определенном развитии техники восстановления изображений можно рассчитывать на получение результатов, общее восприятие которых ничем существенным не будет отличаться от восприятия оригиналов. Такой возможностью сжатия изображений человек пользуется уже давно - описывая их в художественной литературе и воспроизводя при чтении книги.

Как и в случае понимания текстов, в данной работе мы изучаем лишь общую схему подключения решателя к анализу изображений. Простейшая версия анализатора использует в качестве входных данных черно-белые двоичные битмэпы, создаваемые с помощью курсора мыши. Эти битмэпы обрабатываются на уровне интерпретатора ЛОСа для выделения системы линий, соединяющих узловые точки. Таким образом получается граф изображения, представленный как набор линий и набор точек. Линии сопровождаются наборами качественных и грубых количественных характеристик - разбиением на фрагменты одинакового знака кривизны, на почти прямолинейные фрагменты и т.п.

Выполняется сканирование наборов точек и линий. На нижних уровнях этого сканирования реализуются простейшие приемы, осуществляющие предобработку - отбрасывание мелких заведомо избыточных линий и создание дополнительных линий (например, объединяющих несколько исходных линий в одну длинную линию, либо возникающих при отождествлении близких точек, и т.п.). Наконец, на верхних уровнях сканирования реализуются приемы, усматривающие простейшие образы. Пока в качестве таких образов рассматривались русские рукописные буквы и цифры. Приемы для узнавания букв записаны на ГЕНОЛОГе. Приводится несложное логическое описание эталонного образа буквы, а дополнительные указатели идентификации позволяют создать компилятору программу, способную узнавать букву, заданным образом отличающуюся от эталона. Такие указатели, как и в других разделах, сильно упрощают обучение решателя, позволяя сравнительно компактными средствами аппроксимировать логику образа. Результаты распознавания атомарных фрагментов изображения заносятся в логический контекст, реализованный традиционным для решателя способом как задача на исследование. Сканирование этой задачи берет на себя управление всеми последующими действиями, необходимыми для логического анализа картинки.

На текущий момент введены приемы для всех русских букв, позволяющие решателю распознавать сравнительно аккуратно написанные небольшие слова. Каких-либо проблем с быстродействием не наблюдается - ответы выдаются мгновенно, даже для случаев, когда одна буква расположена внутри другой. Это неудивительно, так как основная работа по выделению линий осуществляется на уровне С-программ, а на уровень логического анализа выносятся лишь сравнительно маломерные структуры данных (десятки линий и точек). Возможности логического управления выбором окна анализа картинки и параметров предобработки позволяют надеяться, что такое быстродействие сохранится и впоследствии.

В случае более сложных изображений ввод логических описаний вручную станет затруднительным. По-видимому, потребуется развитие технологии обучения "на примерах", по которым решатель будет формировать такие описания самостоятельно.

4.6.15 Игровые ситуации

Логические процессы необходимы также при выборе очередного хода в игровых ситуациях. Хорошим модельным объектом здесь являются шахматы, для которых накоплен огромный обучающий материал. Несмотря на наличие сильно играющих шахматных программ, создается впечатление, что адекватное компьютерное моделирование рассуждений шахматиста так и не реализовано. В этих программах, как отмечал выдающийся шахматист и исследователь в области искусственного интеллекта М.М.Ботвинник, основной акцент делается на перебор. Нам не интересен "спортивный" аспект соревнования с такими программами, а важна лишь возможность выявления, с помощью богатейшего шахматного обучающего материала, общих принципов управления логическими процессами при планировании действий. Поэтому и было начато обучение решателя игре в шахматы.

Принятие решения о выборе хода в игровой ситуации существенно отличается от принятия решения о срабатывании приема при решении задачи. В первом случае речь идет об использовании логического процесса для выбора какого-то воздействия на внешнюю среду; во втором - об отдельном шаге этого логического процесса. Ошибка во втором случае может привести лишь к тому, что у решателя появятся

несколько "лишних" верных мыслей, которые, быть может, и не повлияют никак на окончательный результат и на время его получения. Ошибка в первом случае гораздо более критична.

Принятие решения о выборе очередного хода реализовано в логической системе как процесс решения задачи на исследование. Условия ее описывают текущую шахматную позицию, сопровождаемую, быть может, какой - либо предысторией - например, фрагментами ранее составленных планов, по-прежнему актуальными для позиции. Решение состоит в выводе следствий, дополняющих информацию о позиции. В частности, такими следствиями могут служить представляющие интерес цели или отдельные ходы. После того, как цель или ход "введены в рассмотрение", дальнейшие выводы уточняют их последствия и вводят оценки. Как обычно, в процессе решения могут быть поставлены вспомогательные задачи либо применены пакеты продукций, выполняющие какой-либо специализированный перебор с просмотром нескольких последовательных ходов.

Система оценок выбрана пока двухкомпонентная. Первая компонента определяет степень приоритетности оценки; вторая - собственно значение оценки. Ходы или цели, мажорируемые согласно таким оценкам, отбрасываются. По исчерпании выводов, выбирается любой из оставшихся ходов. Приемы записываются на ГЕНОЛОГе. В процессе их накопления развивается некоторая логическая формализация шахматных понятий, оптимизируемая для получения возможно более компактных и понятных текстов. Интересно то обстоятельство, что в шахматах предметный и структурный уровень описания приема практически идентичны. Если в математике вид каких-либо формул, встречающихся в уравнении, обычно мало что говорит о числах, являющихся их значениями, и наборот, то в шахматах "форма" и "содержание" совпадают. По этой причине, для шахмат вся логика принятия решения сосредоточена в теореме приема. В сравнении с математикой, это проще.

Проработка шахматного решателя пока находится на начальном этапе. Освоены простейшие эндшпильные ситуации, связанные с преследованием одинокого короля и пешечными окончаниями. Фактически, этот материал позволил лишь провести несколько первых коррекций языка и архитектуры процесса принятия решения.

4.6.16 Прочие возможности

Список областей, с необходимостью предполагающих использование логических процессов, нетрудно продолжить. Безусловно, к ним относится поиск в базах данных. Запросы на поиск представляют собой, по сути дела, небольшие задачи для решателя. Так как приемы организованы по принципу энциклопедии, сканирование задачи позволяет за единственный шаг обратиться к информации базы данных, связанной с упоминаемым в запросе объектом. Синтез приемов, используемых для обработки запроса, можно при этом осуществлять по исходной логической записи информации автоматически, с помощью сопровождающего ГЕНОЛОГ аппарата. Следующая перспективная для подключения логических процессов область - управление динамическими системами. Впрочем, работе в этом направлении должно предшествовать достижение решателями достаточно высокого уровня в распознавании изображений. Безусловно, очень важное направление - техническое проектирование, хотя и здесь началу работ должно предшествовать хорошее освоение системой как основ физики, так и моделирования физических процессов. Можно и нужно продолжать обучать логическую систему во всех областях, где имеются хорошие задачники, постепенно переходя к рассмотрению задач, предполагающих использование все более разносто-

ронних знаний и дающих системе возможность все дольше работать в автономном режиме. Архитектура системы позволяет увеличить ее объем в сотни раз, практически не замедляя при этом решения задач из уже проработанных разделов.

Объединение в одном комплексе даже таких областей, где уже имеются сильные экспертные системы, необходимо не только из-за общих преимуществ интеграции знаний. Хотя и это существенно - ведь из-за отсутствия каких-либо перегородок между приемами возможности предметных областей как бы "перемножаются". Рассмотрение различных областей на уровне ГЕНОЛОГа означает получение материала, необходимого для анализа того, как алгоритмика возникает из теоретических знаний. Чем более разносторонним будет накопленный материал, тем более достоверными и полными будут извлеченные из него представления об организации логических процессов, и тем реалистичнее станет в конечном счете создание саморазвивающейся интеллектуальной системы.

На этом мы завершаем краткий обзор возможностей и перспектив решателя. Более подробное описание его устройства будет приведено во втором томе монографии. Однако, такому описанию должно предшествовать изложение технических основ обучения решателей - созданных для этого языков ЛОС и ГЕНОЛОГ, а также многочисленных средств интерфейса обучения. В оставшейся части данного тома философия искусственного интеллекта уходит на второй план, а на первый план выходят скучные технические подробности, без которых, впрочем, никакое серьезное продвижение вперед в этой области не представляется возможным.

Глава 5

Алгоритмический язык ЛОС

5.1 Структуры данных и общая схема организации программы на языке ЛОС

Программы приемов решателя записываются на языке ЛОС (Логический Описатель Ситуаций) и реализуются интерпретатором этого языка. Язык ЛОС по своему уровню сопоставим с ПРОЛОГом, однако специально адаптирован к изложенной выше схеме сканирования задач и имеет много принципиальных отличий от ПРОЛОГа. Для логической системы он является языком низкого уровня, и запись на нем приемов "вручную"предпринимается крайне редко. Фактически приемы записываются на языке более высокого уровня, ГЕНОЛОГе, и далее компилятор ГЕНОЛОГа создает на основе этой записи исполняемую ЛОС-программу приема. Вместе с тем, основу языка ЛОС составляют около 200 предикатов и операций, которые могут рассматриваться не только как алгоритмический, но и как вполне развитый логический язык для описаний ситуаций в логико-сетевых структурах данных. Поэтому в тех случаях, когда речь идет не о работе с объектами предметной области, а о работе с объектами структурного уровня (иными словами, когда предметной областью являются сами структуры данных), логические уровни ЛОСа и ГЕНОЛОГа, по существу, совпадают, и программирование приемов непосредственно на ЛОСе становится вполне оправданным.

В языке ЛОС выделяются следующие основные типы объектов, обрабатываемых программой:

а) Символы переменных и логические символы. Максимально возможное число переменных и логических символов определяется конкретной версией интерпретатора; используемая в настоящее время версия допускает применение $2^{16}-1$ логических символов и стольких же переменных. Можно вводить, удалять либо изменять название ранее введенных логических символов. Каждое название логического символа представляет собой произвольную последовательность символов расширенного алфавита (допускаются русские и латинские буквы,цифры и пр.), длина которой не более 24. Собственно в программных файлах ЛОСа логические символы представлены при помощи своих номеров, а названия их хранятся в отдельном файле, что позволяет изменять эти названия, не изменяя программ.

б) Термы, построенные из переменных и логических символов (заголовок операции - произвольный логический символ, число операндов - любое). Терм длины 1 отождествляется с набором длины 1 (см.пункт в) и отличается от входящего в него логического символа.

в) Наборы $(a_1, \ldots, a_n)$ ранее определенных допустимых объектов.

г) Вхождения символов в термы и разрядов в наборы.

Заметим, что для скобок не предусмотрены специальные логические символы, а терм $\varphi(a_1 \ldots a_n)$ не является набором входящих в него логических символов и скобок - в структурах данных интерпретатора с ЛОСа скобки представлены неявным образом (указаны длины подтермов и ссылки на внешние операции). Тем не менее, во многих случаях терм можно рассматривать как последовательность составляющих его логических символов и символов переменных с опущенными скобками, применяя при этом операторы, предназначенные для работы с наборами. Для обозначения пустого набора выделен логический символ "пустоеслово", воспринимаемый рядом операторов языка как набор длины 0.

Все логические символы пронумерованы натуральными числами. При использовании их в качестве заголовков операторов языка ЛОС выделяются непосредственно реализуемые логические символы (номера от 1 до 240) и программно реализуемые логические символы. С непосредственно реализуемыми символами связаны соответствующие процедуры интерпретатора; выполнение операторов с программно реализуемыми заголовками происходит при помощи программ языка ЛОС.

Представление чисел в решателе осуществляется четырьмя различными способами. С одной стороны, для быстрых выкладок с целыми числами диапазона от -259 до 65535 используется кодирование их логическими символами (логический символ "0"имеет номер 260). С другой стороны, отличное от цифры десятичное число $a_1 \ldots a_n, a_{n+1} \ldots a_{n+m}$ кодируется набором $(a_1 a_2 \ldots a_n, a_{n+1} \ldots a_{n+m})$, где запятая является логическим символом (изменение знака числа происходит путем добавления либо удаления логического символа "минус"в начале набора), а цифра отождествляется с обозначающим ее логическим символом. При работе с термами применяется представление указанного числа в виде "величина $(a_1 \ldots a_n, a_{n+1} \ldots a_{n+m})$"(отрицательного числа - в виде "минус (величина $(a_1 \ldots a_n, a_{n+1} \ldots a_{n+m})$)"; цифры - в виде однобуквенного терма). Наконец, для приближенных вычислений с математическим сопроцессором и для вычислений с применением обычной 32-битной машинной арифметики используются форматы "с плавающей запятой"(64-битное представление) и "целое со знаком"(32 бита). Числа в указанных форматах кодируются специальным образом: первое - набором длины 2, второе - набором длины 1. Эти наборы несколько отличаются от обычных наборов ЛОСа, и работать с ними нужно только при помощи операторов, специально предназначенных для машинной арифметики.

Программе доступна совокупность данных, образованная, с одной стороны, значениями ее переменных (входные данные и вспомогательные объекты, формируемые программой), а с другой стороны, некоторая совокупность внешних данных, описывающих цепь задач, сложившуюся в процессе решения предложенной системе задачи. Как уже отмечалось в главе 3, задача Z представляет собой набор $(p_0, p_1, \ldots, p_n)$, где p_0 - логический символ, определяющий тип задачи; p_1 - набор посылок задачи; p_2 - набор весов посылок (логических символов от "0"до "20"); p_3 - набор наборов объектов, представляющих собой комментарии к посылкам, и т.д. Цепь задач представляет собой упорядоченную последовательность задач $(Z_0, Z_1, \ldots, Z_N)$, где Z_0 - исходная задача; Z_N - текущая решаемая задача; Z_{i+1} - подзадача задачи Z_i $(i = 1, \ldots, N-1)$. Эта последовательность сама не образует допустимого объекта (набора задач) и обращение к ней осуществляется через специальные операторы языка. Такие операторы, в частности, позволяют определить по каждому элементу Z_i цепи задач его текущий m_i и максимальный M_i уровни (логические символы от "0"до "20"). Исходная задача Z_0 имеет фиктивный характер и создается автоматически при включении системы.

Она представляет собой задачу на исследование с единственной фиктивной однобуквенной посылкой "вход"; при сканировании этой посылки запускается написанная на языке ЛОС программа интерфейса, позволяющая сформировать фактически решаемую задачу Z_1. Список общих комментариев к посылкам задачи Z_0 используется в качестве "доски объявлений", доступной любой процедуре системы. Главным образом, эта доска объявлений используется процедурами интерфейса.

Помимо указанных здесь основных типов данных, предусмотрены различные типы данных внешнего характера (файлы с древовидными информационными конструкциями, файлы программ ЛОСа и др.). Более подробно эти типы данных описываются ниже, при рассмотрении операторов языка ЛОС. Для работы с внешними данными используются ссылки, представленные посредством данных основных типов (как правило, наборы логических символов). Перечень внешних типов данных системы является, по существу, открытым, так как организация интерпретатора языка позволяет сравнительно легко подключать дополнительные непосредственно реализуемые операторы, работающие с данными нового типа.

Программа решателя организована по энциклопедическому принципу и распадается на программы отдельных логических символов. Программа логического символа φ имеет древовидную структуру и состоит из совокупности пронумерованных натуральными числами текстов, называемых фрагментами этой программы. Каждый фрагмент программы логического символа φ представляет собой последовательность $P_1, \ldots, P_n$ термов специального вида, называемых операторами языка. В большинстве случаев эти термы представляют собой просто утверждения логического языка, значениями переменных которых служат данные указанных выше типов.

Два специальных типа операторов, а именно "ветвь(N)" и "иначе(N)", где N - натуральное число, называются операторами перехода (для краткости скобки в операторах перехода опускаются, т.е. применяется запись "ветвь N", "иначе N"). Если в i - м фрагменте встречается оператор "ветвь j" либо "иначе j", то говорим, что этот фрагмент имеет ссылку на j - й фрагмент. Граф ссылок между фрагментами программы является деревом с корнем в 1-м фрагменте; 1-й фрагмент называется также началом программы символа φ. Оператор "иначе N"располагается в фрагменте только после некоторого другого оператора, не являющегося оператором перехода. Во избежание путаницы сразу заметим, что в интерфейсе программного редактора ЛОСа применяется не глобальная, как сказано выше, а локальная нумерация фрагментов программы: ссылки из каждого фрагмента на подфрагменты пронумерованы там независимым образом последовательными натуральными числами (по мере появления их в тексте фрагмента), начинающимися с 1. Указанная выше глобальная нумерация фрагментов введена здесь лишь для удобства описания языка; фактически в файле программы ЛОСа ссылка из фрагмента на подфрагмент задается смещением этого подфрагмента в файле.

В операторах используется специальный класс подтермов, называемых операторными выражениями. В большинстве случаев эти подтермы суть некоторые выражения логического языка. Входящие в операторные выражения переменные обозначаются в программе ЛОСа посредством буквы "x"с номером: x1, x2, ... По определению, каждое однобуквенное выражение является операторным. Синтаксис операторных выражений $\varphi(t_1 \ldots t_n)$ определяется независимо для каждого символа φ. Непосредственно реализуемые символы φ рассматриваются в следующем разделе данной главы; в случае программно реализуемого символа все операнды $t_1, \ldots, t_n$ сами должны быть операторными выражениями (число n может варьироваться, при условии, что программа способна самостоятельно определить его по заведомо доступным ей

данным). Аналогичным образом, синтаксис операторов $\varphi(t_1 \dots t_n)$, $n \geq 0$, определяется независимо для каждого символа φ, причем в случае программно реализуемого символа φ все $t_1, \dots, t_n$ суть операторные выражения. При определении оператора вида $\varphi(t_1 \dots t_n)$ указывается разбиение множества его свободных переменных на входные и выходные переменные; выходная переменная имеет (кроме случаев $\varphi \in \{$ "и", "или", "альтернатива"$\}$) ровно одно вхождение, являющееся вхождением некоторого операнда $t_i (i = 1, \dots, n)$. В некоторых случаях допускаются различные варианты такого разбиения, причем интерпретатор ЛОСа при реализации оператора выбирает то из них, для которого значения всех входных переменных уже определены программой, а всех выходных - не определены.

При обращении к программе логического символа φ, а также на каждом шаге выполнения этой программы выделяется некоторый набор переменных, значения которых считаются определенными , причем программа способна самостоятельно отличать их от прочих переменных.

Обращения к программе символа φ бывают следующих трех типов:

1) Обращение в процессе сканирования текущей решаемой задачи Z_N (см. описание процесса сканирования задачи в главе 4). Если φ - тип задачи и произошло обращение к процедуре, объединяющей приемы решения задач типа φ без привязки к конкретному вхождению логического символа в задачу (см. пункт 1 описания процедуры сканирования), то определены значения переменных $x1$ (задача Z_N) и $x5$ (текущий уровень m_N этой задачи). Если в задаче выделено конкретное вхождение символа φ (пункт 3 описания процедуры сканирования), то определены $x1 = Z_N$, $x2$ - данное вхождение символа φ в текущую посылку либо условие задачи, $x3$ - вхождение текущей посылки либо условия задачи в список посылок либо условий (при отсутствии такового - в саму задачу), $x4$ равно 0, если просматривается посылка задачи, и равно 1 в противном случае, $x5 = m_N$. Таким образом, тройка $(x2, x3, x4)$ определяет координаты вхождения в задачу выделенного символа φ. Если произошло обращение к процедуре символа, являющегося названием типа задачи (пункт 1 процедуры сканирования), то значения $x2, x3, x4$ равны логическому символу 0.

2) Обращение при выполнении программно реализуемого оператора либо нахождении значения операторного выражения $\varphi(t_1 \dots t_n)$. В этом случае определены все переменные $xi, 1 \leq i \leq n$, для которых t_i не является выходной переменной оператора. Остальные переменные $xi, i \in \{1, \dots, n\}$, считаются не определенными и должны оставаться таковыми вплоть до момента присвоения им специальными заключительными операторами результатов выполнения программы перед выходом из нее.

3) Приемы решения задач часто используют вспомогательные процедуры, распадающиеся на систему независимых подпроцедур, связанных с различными логическими символами. Такие процедуры обеспечивают получение информации справочного характера, связанной с соответствующими логическими символами (например, определяют арность символа; находят область допустимых значений операции с заданным заголовком, и т.п.). Эти процедуры называем далее справочниками. Каждая из них обозначается специальным логическим символом, отличным от символа "0" - типом данного справочника. Тип справочника ψ рассматривается как характеристика обращения к программе символа φ при попытке получить информацию, связанную с этим символом. Чтобы определить, что имеет место именно такой тип обращения, используется вспомогательный оператор "обращение(ψ)", истинный лишь в этом случае. Значения переменных $x1, \dots, xn$, определенных при обращениях к справочнику типа ψ, задаются осуществляющим запрос операторным выражением "справка($\psi\varphi t_1 \dots t_n$)"($x_i$ получает значение операторного выражения t_i).

Операторы языка ЛОС делятся на проверочные, перечисляющие и смешанного проверочно-перечисляющего типа. Проверочный оператор, при указании значений его входных переменных, либо принимает значение "ложь", либо однозначным образом определяет значения выходных переменных и принимает значение "истина". Перечисляющий оператор, при определении значений его входных переменных, генерирует некоторую конечную последовательность (возможно,пустую) наборов значений выходных переменных, принимая после каждой выдачи очередного набора значение "истина", после чего принимает значение "ложь". Операторы смешанного типа функционируют либо в проверочном, либо в перечисляющем режиме, в зависимости от конкретных значений их входных переменных. Оператор "ветвь N"по определению считается перечисляющим. Для возвращения к перечисляющему оператору и получения от него очередного набора значений выходных переменных при реализации программы создается стэк перечисляющих операторов $(Q_1 \dots Q_m)$, которые относятся, вообще говоря, к различным фрагментам программы символа φ, находящимся на пути от начала программы к текущему фрагменту.

Реализация очередного оператора P_i текущего фрагмента $P_1, \dots, P_n$ программы символа φ происходит следующим образом:

1) Если оператор имеет вид $\psi(t_1 \dots t_n)$, где ψ - программно реализуемый символ, то последовательно определяются значения входных операторных выражений t_i и осуществляется обращение к программе символа ψ; действия в случае непосредственно реализуемого ψ описаны в следующем разделе. Если по окончании выполнения оператора он принимает значение "ложь", то переход к пункту 2. В противном случае оказываются определены значения всех выходных переменных этого оператора. Если при реализации оператора P_i имел место режим перечисления, то оператор заносится в конец стэка перечисляющих операторов. Если $i < n$, то переход к выполнению следующего оператора P_{i+1}, иначе - переход к пункту 3.

2) Если P_{i+1} имеет вид "иначе N", то осуществляется переход к первому оператору N - го фрагмента программы. Если стэк перечисляющих операторов пуст, то выполнение программы символа φ завершается. При этом, если происходило сканирование задачи, то делается очередной шаг сканирования; если вычислялось операторное выражение $\varphi(\theta_1 \dots \theta_k)$, то выдается значение "0" (логический символ); если же обрабатывался оператор $\varphi(\theta_1 \dots \theta_k)$, то он получает значение "ложь". Если стэк перечисляющих операторов $(Q_1 \dots Q_m)$ непуст, то осуществляется возвращение к последнему оператору Q_m, который удаляется из стэка. При этом значения всех переменных, номера которых больше номера последней определенной до обращения к Q_m переменной, становятся не определены. Оператор Q_m определяет очередной набор значений своих выходных переменных либо, при отсутствии такого набора, принимает значение "ложь", и далее повторяется выполнение программы начиная с оператора, следующего за Q_m. Особо выделяется случай оператора Q_m вида "ветвь N". В этом случае просто выполняется переход к первому оператору N - го фрагмента программы.

3) В языке ЛОС имеется несколько специальных операторов, размещаемых обычно в конце фрагментов программы и используемых для управления ходом выполнения программы. Если P_i представляет собой один из таких заключительных операторов, то выполняются следующие действия:

а) $P_i =$ "выход". В этом случае программа символа φ осуществляет реализацию проверочного оператора, который получает значение "истина".

б) $P_i =$ "стоп". В этом случае программа осуществляет реализацию оператора (не обязательно проверочного), который получает значение "ложь".

в) P_i = "результат($xk_1\ t_1\ xk_2\ t_2 \ldots xk_s\ t_s$)". Программа осуществляет реализацию перечисляющего оператора $\varphi(\tau_1 \ldots \tau_p)$; выходным переменным $\tau_{k_1}, \ldots, \tau_{k_s}$ этого оператора присваиваются значения операторных выражений $t_1, \ldots, t_s$. Заметим, что при реализации проверочного оператора $\varphi(\tau_1 \ldots \tau_p)$ присвоение значений выходным переменным происходит таким же образом, но оператор "результат(...)" располагается непосредственно перед завершаюшим оператором "выход".

г) P_i = "продолжение". Этот оператор является тождественно ложным, т.е. происходит возвращение к последнему оператору Q_m стэка перечисляющих операторов либо (при пустом стэке) выход из программы.

д) P_i = "обрыв" либо P_i = "сброс(k)"; k - натуральное число. Происходит возвращение к оператору Q_{m-k} (в первом случае $k = 1$) из стэка ($Q_1 \ldots Q_m$) перечисляющих операторов; если такого оператора нет, то - выход из программы, как при пустом стэке.

е) P_i = "ответ(t)". В этом случае программа либо выполняла сканирование задачи, и тогда определяется ответ t на эту задачу, либо вычисляла операторное выражение (в частности, вида "справка($\psi\,\varphi \ldots$)"), получающее значение операторного выражения t.

ж) P_i = "пересмотр". В этом случае программа выполняет сканирование задачи, которое возобновляется повторно при нулевом текущем уровне задачи.

з) P_i = "контроль". Программа выполняет сканирование задачи. Если эта задача имеет хотя бы один терм (посылку либо условие) с нулевым весом (что бывает при изменении терма либо занесении нового терма), то начинается повторное сканирование задачи при нулевом текущем уровне. В противном случае - переход к последнему перечисляющему оператору Q_m (если его нет, то выход из программы).

Если P_n не является оператором одного из указанных видов, то либо имеет место сканирование задачи, и тогда осуществляется очередной шаг сканирования, либо происходит реализация проверочного оператора без выходных переменных, который получает значение "истина".

5.2 Основные операторы языка ЛОС

5.2.1 Общие операторы

Язык ЛОС имеет свыше 200 реализуемых интерпретатором операторов, ориентированных главным образом на обработку данных сетевого и логического типов. Часть этих операторов ("выход", "стоп", "результат($x1\ t_1 \ldots xn\ t_n$)", "ответ(t)", "обрыв", "сброс(n)", "продолжение", "контроль", "пересмотр") была рассмотрена в предыдущем разделе. Приведем еще ряд простых операторов управляющего характера.

Операторы "решить", "программа", "обращение(ψ)" принимают значение "истина", соответственно, если программа символа φ применяется при сканировании задачи, при реализации оператора и при вычислении операторного выражения t; в последнем случае $\psi \neq$ "0" означает, что t имело вид "справка($\psi\,\varphi \ldots$)", а $\psi =$ "0" - что t не имело такого вида. Оператор "определено(xi)" истинен, если определено значение переменной xi. Оператор "повторение" представляет собой вырожденный перечисляющий оператор, принимающий при каждом выходе на него значение "истина"; он аналогичен оператору repeat в ПРОЛОГе. Оператор "уровень($n_1 \ldots n_k$)" (n_1, $\ldots, n_k$ - логические символы от "0"до "20") истинен, если происходит сканирование задачи, текущий уровень которой равен одному из чисел $n_1, n_2, \ldots, n_k$; оператор "по-

следнийуровень если текущий уровень решаемой задачи Z_N равен ее максимальному уровню. Оператор "новый"применяется при сканировании задачи; он принимает значение "истина"в том случае, когда рассматриваемый терм задачи впервые попал в поле зрения системы при данном текущем уровне m_N. Более точно, этот оператор становится ложнм лишь в ситуациях повторного просмотра при значениях текущего уровня $U = 0, 1, \ldots, m_N$ временно выпавшего из поля зрения терма, имеющего вес $m_N + 1$ (см. пункт 3 описания процедуры сканирования задачи в последнем разделе первой главы). Оператор "новый" позволяет блокировать повторную попытку рассмотрения приема, если изменение окрестности терма не влияет на результаты такого рассмотрения.

Перейдем к описанию серии общелогических операторов языка. Если P - произвольный проверочный оператор, не имеющий выходных переменных, то утверждение "не(P)" является проверочным оператором, истинным тогда и только тогда, когда ложен P. Если $P_1, \ldots, P_n$ - операторы, то утверждение "и($P_1 \ldots P_n$)" является оператором. Если хотя бы один из операторов $P_1, \ldots, P_n$ реализуется в режиме перечисления, то конъюнкция их также считается реализуемой в режиме перечисления. В стэк перечисляющих операторов при этом заносится не последний перечисляющий оператор P_i, а вся конъюнкция. Аналогичным образом формируется дизъюнкция "или($P_1 \ldots P_n$)" операторов $P_1, \ldots, P_n$. Если каждый из операторов $P_1, \ldots, P_n$ проверочный и не имеет выходных переменных, то дизъюнкция является проверочным оператором. Если каждый оператор $P_i, i = 1, \ldots, n$, имеет непустое множество выходных переменных, то дизъюнкция реализуется в режиме перечисления (даже если все P_i - проверочные). При этом сначала перечисляются значения выходных переменных, определяемые оператором P_1, затем - P_2, и т.д. Дизъюнктивные конструкции, у которых часть операторов P_i имеет пустое множество выходных переменных, а часть - непустое, не рекомендуются, так как тогда наличие либо отсутствие режима перечисления определяется неоднозначно.

Если $P_1, \ldots, P_n$ - операторы, $n > 0$,P_0 - проверочный оператор, $x_1, \ldots, x_k$ - переменные, содержащие все выходные переменные операторов $P_0, P_1, \ldots, P_n$, то утверждение "длялюбого($x_1 \ldots x_k$ если $P_1 \ldots P_n$ то P_0)" является проверочным оператором. К моменту его реализации последняя определенная переменная должна иметь номер, меньший номеров переменных $x_1, \ldots, x_k$. Все свободные переменные оператора P являются его входными переменными. При фиксированных значениях этих переменных цепочка операторов $P_1, \ldots, P_n$ (среди которых могут встречаться перечисляющие операторы) определяет процесс перечисления удовлетворяющих условиям $P_1, \ldots, P_n$ наборов значений их выходных переменных, причем для каждого такого набора вычисляется истинностное значение оператора P_0. Как только здесь возникает значение "ложь", процесс обрывается и P получает значение "ложь". В противном случае, по завершении перечисления, оператор P получает значение "истина", а значения всех переменных $x_1, \ldots, x_k$ снова оказываются не определены. Заметим, что управляющие ходом перечисления операторы "повторение", "обрыв", "сброс(n)" внутри сложных операторов (в частности, в кванторных операторах) не используются. Аналогичным образом определяется оператор P вида "существует($x_1 \ldots x_k\ P_0$)", где P_0 - некоторый оператор (возможно, перечисляющий), все выходные переменные которого содержатся среди $x_1, \ldots, x_k$. Как и в случае квантора общности, к моменту реализации P последняя определенная переменная должна иметь номер, меньший номеров переменных $x_1, \ldots, x_k$. (Это правило является общим для всех операторов и операторных выражений со связанными переменными и далее особо не оговаривается. Вообще, при программировании на ЛОСе предпочтительно вводить новые

программные переменные так, чтобы очередной номер переменной был на единицу больше наибольшего из уже использованных номеров.) Если при фиксации значений входных переменных оператора P оператор P_0 оказывается истинным для некоторого набора значений своих выходных переменных, то определяемое им перечисление обрывается, P получает значение "истина", а значения переменных $x_1, \ldots, x_k$ становятся снова не определены.

К числу основных логических операторов присоединен оператор P = "альтернатива($P_0P_1P_2$)", где P_0 - проверочный оператор без выходных переменных; P_1, P_2 - некоторые операторы (не обязательно проверочные). Если P_0 получает на некотором наборе значений своих переменных значение "истина", то далее выполняется оператор P_1, иначе - оператор P_2. Здесь возникает возможность появления смешанных проверочно- перечисляющих операторов, так как если один из операторов P_1,P_2 проверочный, а другой - перечисляющий, то выбор фактического режима работы оператора P происходит в зависимости от значений его входных переменных. По аналогии с оператором "альтернатива($P_0P_1P_2$)" устроено операторное выражение "вариант(Pt_1t_2)", где P - проверочный оператор без выходных переменных; t_1 и t_2 - операторные выражения. Значение этого выражения совпадает со значением t_1 при истинном P и со значением t_2 при ложном P.

Для формирования наборов объектов, накапливаемых в процессе просмотра тех или иных структур данных, служат операторные выражения со связанными переменными "перечисление($x_1 \ldots x_k\, P\, t$)"и "выписка($x_1 \ldots x_k\, P\, t$)"(аналоги setof и findall в ПРОЛОГе). Здесь P - некоторый (как правило, перечисляющий) оператор; t - операторное выражение; $x_1, \ldots, x_k$ - переменные, включающие все выходные переменные оператора P. При определении значения этих выражений оператор P перечисляет некоторые наборы значений своих выходных переменных, и для каждого такого набора определяется значение выражения t. В первом случае (оператор "перечисление(...)") найденные указанным образом значения выражения t записываются в формируемый набор подряд, с исключением повторений; во втором случае (оператор "выписка(...)") исключение повторений не выполняется. Операторное выражение "суммавсех($x_1 \ldots x_k\, P\, t$)"осуществляет суммирование значений выражения t (эти значения должны быть числами, т.е. цифрами либо наборами, см. описание представления чисел в разделе 1), возникающих в процессе реализации перечисляющего оператора P так же, как для операторов "перечисление(...)", "выписка(...)". Это выражение используется , например, при принятии решений для интегральной оценки ситуации.

Вычисление значения уже упоминавшегося в первом разделе операторного выражения "справка($\psi\varphi t_1 \ldots t_n$)", где ψ, φ - логические символы; $t_1, \ldots, t_n$ - операторные выражения, осуществляется программой логического символа φ при "типе обращения"ψ к этой программе и исходных значениях $t_1, \ldots, t_n$ переменных $x_1, \ldots, x_n$. Это выражение используется в тех случаях, когда некоторая процедура, условно обозначаемая логическим символом ψ, естественным образом распадается на множество независимых фрагментов, связанных с различными логическими символами φ.

Оператор "равно(t_1t_2)"либо не имеет выходных переменных (если все входящие в него переменные имеют уже определенные на предыдущих этапах значения) и осуществляет проверку равенства значений выражений t_1 и t_2 (равенство наборов понимается как равенство определяемых ими древовидных конструкций), либо имеет выходную переменную t_1, которой присваивается значение выражения t_2.

Для изменения значения переменной x_N используется оператор "замена(x_N t)", где t - операторное выражение, определяющее заменяющий объект. Сразу заметим,

что в фактически исполняемой программе здесь вместо переменной x_N хранится ее символьный номер (логический символ номер $260 + N$), который и сообщается интерпретатору ЛОСа в качестве первого входного данного оператора "замена(...)". Аналогичный оператор "изменение(t_1t_2)" позволяет изменять разряд набора, вхождение которого определяется выражением t_1, на объект, являющийся значением выражения t_2. Применение операторов "замена(...)"и "изменение(...)" требует определенной аккуратности, так как после их реализации значения переменных могут измениться таким образом, что истинность предшествующих операторов рассматриваемого фрагмента программы нарушится.Эти операторы оказываются полезны при реализации циклических процедур в тех достаточно редких случаях, когда цикл не удается оформить с помощью перечисляющих операторов.

Возможно обращение к оператору либо операторному выражению, заголовок которых определяется некоторым операторным выражением. Это осуществляется при помощи терма "процедура(t_1 t_2 t_3)", у которого значением t_1 является указанный заголовок, причем при t_3 равном "0"данный терм реализует обращение к оператору, а при t_3 равном "1 к операторному выражению. Значением t_2 является набор входных данных соответствующего оператора либо операторного выражения. В случае оператора с выходными переменными позиции набора t_2, соответствующие выходным переменным, заняты логическим символом "неопред". Эти позиции после обращения к оператору "процедура(. . .)"изменяются на найденные значения выходных переменных.

В тех случаях, когда предпринимается обращение к процедуре, выполнение которой может оказаться неприемлемо трудоемким, используется фиктивный перечисляющий оператор "лимит(t_1)". Этот оператор устанавливает ограничение в t_1 шагов работы интерпретатора (количество таких шагов автоматически фиксируется в специальном регистре интерпретатора и может рассматриваться как машинно-независимые внутренние "часы"решателя) начиная с момента обращения к нему. По истечении этого числа шагов, если все еще не было выхода из следующей за данным оператором ветви программы, реализуется откат к оператору с присвоением ему истинностного значения "ложь". Если после обращения к подпрограмме, которое могло оказаться слишком трудоемким, нужно отменить откат к оператору "лимит(. . .)", то применяется оператор "Обрыв". Единственное его действие - исключение из стэка перечисляющих операторов последнего элемента (не обязательно установленного оператором "лимит(. . .)"), без каких-либо откатов.

В заключение подраздела упомянем операторное выражение "типданных(t)", позволяющее различать типы значения выражения t. Если A - логический символ либо символ переменной, то значение p выражения "типданных(t)"равно 0; если A - терм, то p равно 1; если A - вхождение в терм либо в набор, то p равно 2; если A - набор, не являющийся термом, то p есть 4. Значение $p = 3$ зарезервировано для тех случаев, когда A есть ссылка на недопустимый объект (сигнал о наличии ошибки).

5.2.2 Операторы просмотра и преобразования задачи

Для выделения отдельных элементов задачи служит ряд специальных операторных выражений. Так, "цели(t)"имеет своим заголовком список целей задачи, определяемой выражением t; "неизвестные(t)" - цель, перечисляющую неизвестные (если ее нет, то это выражение имеет значение "пустоеслово"); "комментарии(t_1)", "комментариипосылок(t_1)", "комментарииусловия(t_1t_2)", "комментариипосылки(t_1t_2)" - соответствующие списки комментариев (задача определяется здесь выражением t_1, а

вхождение рассматриваемого условия либо посылки - выражением t_2); "списокусловий(t)" и "списокпосылок(t)" - списки условий и посылок. Значением выражения "переменные(t)" служит список всех переменных, встречающихся в посылках и условиях задачи, а также среди ее неизвестных. Выражение "максимальныйуровень(t)" позволяет определить максимальный уровень задачи. Для обращения к процедуре решения задачи служат операторные выражения "ответзадачи(t)" и "прямойответ (t)". Выражение t в обоих случаях имеет своим значением некоторую задачу (т.е. набор допустимого вида; см.описание структуры данных задачи в третьей главе), причем в первом случае эта задача начинает решаться с максимальным уровнем, равным текущему уровню решаемой задачи Z_N, а во втором случае - с максимальным уровнем 4. Если перед вычислением значения выражения "ответзадачи(t)"применить оператор "уровеньобращения(k)", то новая задача будет решаться с максимальным уровнем k (установка уровня обращения к задаче имеет одноразовый характер и после обращения к задаче пропадает). Ответ либо логический символ "отказ", найденный при решении задачи, становится значением соответствующего операторного выражения.

Операторы "исходнаязадача(x)", "текущаязадача(x)"присваивают переменной x исходную и текущую задачи цепи задач, а оператор "надзадача(x t)"перечисляет в обратном порядке (т.е. от текущей задачи к исходной) все элементы цепи задач, тип которых равен значению выражения t. Если t имеет значение 0, то последнее ограничение отменяется. Сама текущая задача из перечисления исключается. Оператор "текущийуровень(x)"присваивает переменной x значение текущего уровня текущей задачи. Операторы "посылка($t_1t_2t_3$)"и "условие($t_1t_2t_3$)"используются в следующих двух вариантах:

а) Значением выражения t_3 является задача Z; t_1, t_2 - выходные переменные. В этом случае перечисляются все посылки либо, соответственно, условия задачи Z (порядок перечисления - слева направо), причем значением t_1 становится рассматриваемый терм, а значением t_2 - его вхождение в список посылок либо список условий. Если Z не имеет списка условий, то при реализации оператора "условие(...)"режим перечисления не включается; значением t_1 становится единственное условие задачи, а значением t_2 - его вхождение в задачу.

б) t_1 и t_3 - входные выражения, значения которых суть терм θ и задача Z; t_2 - выходная переменная. Выполняется проверка того, что θ - посылка (условие) задачи Z, и если это так, то t_2 становится равно вхождению θ в соответствующий список либо в саму задачу Z.

В некоторых случаях бывает необходимо осуществить просмотр всех посылок либо условий задачи, имеющих заданный вес; как правило, таковыми оказываются вновь введенные за некоторый период посылки либо условия. Для этого применяются операторы "новаяпосылка($t_1t_2t_3t_4$)", "новоеусловие($t_1t_2t_3t_4$)", реализация которых отличается от случая а) реализации операторов "посылка($t_1t_2t_3$)", "условие($t_1t_2t_3$)" лишь тем, что отбираются термы с весом, равным значению выражения t_4. Операторы "цель(t_1t_2)", "неизвестная(t_1t_2)"допускают два режима реализации, причем в каждом из них значением выражения t_1 является задача Z, не имеющая тип "доказать":

а) t_2 - выходная переменная, и тогда оператор перечисляет значения переменной t_2, являющиеся целями (неизвестными) задачи Z.

б) t_2 определено, и тогда оператор проверяет, что значением выражения t_2 является цель (неизвестная) задачи Z.

Ряд операторов служит для преобразования задачи. Прежде всего, это операторы

"заменаусловия($t_1t_2t_3$)"и "заменапосылки($t_1t_2t_3$)" (t_1 - задача, t_2 - вхождение заменяемого терма в соответствующий набор, t_3 - заменяющий терм); "занесениеусловия(t_1t_2)"и "занесениепосылки(t_1t_2)"(t_1 - терм, t_2 - задача, имеющая в первом случае тип "описать"); "удалениеусловия(t_1t_2)"и "удалениепосылки(t_1t_2)"(t_1 - задача, t_2 - вхождение удаляемого терма). Измененный либо новый терм получает в задаче вес 0; если оператор указанного типа является заключительным в фрагменте программы и обращение к этой программе произошло при сканировании задачи, то текущий уровень задачи заменяется на 0 и цикл сканирования повторяется сначала.

Для занесения в соответствующие списки новых комментариев используются операторы "примечание($t\ \theta_1 \dots \theta_n$)"(вводится общий комментарий ($\alpha_1 \dots \alpha_n$) либо, при $n = 1$, общий комментарий α_1 к посылкам задачи, определяемой выражением t; $\alpha_1, \dots, \alpha_n$ - значения выражений $\theta_1, \dots, \theta_n$; α_1 - логический символ), "замечание($t\ \theta_1 \dots \theta_n$)" (вводится общий комментарий к задаче, определяемой выражением t), "примечпосылки($t\ \tau\ \theta_1 \dots \theta_n$)" (вводится комментарий к посылке, определяемой своим вхождением τ в список посылок), "замечусловие($t\ \tau\ \theta_1 \dots \theta_n$)" ($t$ определяет задачу на описание; τ - вхождение в список условий того условия, к которому относится комментарий).

Проверка отсутствия заданного комментария выполняется аналогичными по своей структуре операторами "коммент($t\ \theta_1 \dots \theta_2$)"(отсутствие среди общих комментариев к задаче набора ($\alpha_1 \dots \alpha_n$) либо, при $n = 1$, символа α_1), "комментпосылок ($t\ \theta_1 \dots \theta_n$)", "комментусловия($t\ \tau\ \theta_1 \dots \theta_n$)", "комментпосылки($t\ \tau\ \theta_1 \dots \theta_n$)".

Завершают перечень операторов задач операторы "удалениепримечания($t\ \theta$)", "удалениезамечания($t\ \theta$)", "удалениепримечпосылки($t\ \tau\ \theta$)", "удалениезамечусловия ($t\ \tau\ \theta$)", выполняющие удаление определяемого выражением θ комментария (t определяет задачу, τ - вхождение рассматриваемой посылки либо условия.

5.2.3 Операторы логического представления данных

Начнем с перечисления операторных выражений, предназначенных для работы с логическими конструкциями. Выражения "списокпеременных(t)", "параметры(t)" определяют, соответственно, набор всех переменных терма, являющегося значением выражения t, и набор всех свободных переменных этого терма. В последнем случае значением выражения t может служить как один терм, так и набор термов. Для образования новых термов используется операторное выражение "запись($t\theta_1 \dots \theta_n$)"; если выражения $t, \theta_1, \dots, \theta_n$ определяют, соответственно, логический символ φ и термы либо символы (т.е. логические символы либо символы переменных) $\tau_1, \dots, \tau_n$, то формируется новый терм $\varphi(\tau_1 \dots \tau_n)$. В тех случаях, когда набор операндов имеет переменную длину, новые термы определяются операторным выражением "сборка($t\ \theta$)". Здесь t определяет логический символ φ; θ - набор операндов $\tau_1, \dots, \tau_n$ - термов либо символов. Для выделения фрагментов уже построенного терма применяются операторные выражения "подтерм(t)" (t указывает вхождение корня переписываемого подтерма), "набороперандов(t)" (t указывает вхождение первого символа терма $\varphi(\theta_1 \dots \theta_n)$, и значением выражения становится набор термов $\theta_1, \dots, \theta_n$), а также операторные выражения "первыйтерм(t)", "второйтерм(t)", "предпоследтерм (t)", "последнийтерм(t)"для наиболее часто рассматриваемых операндов $\theta_1, \theta_2, \theta_{n-1}$, θ_n терма $\varphi(\theta_1 \dots \theta_n)$, вхождение первого символа которого определяется выражением t. Операторные выражения "первыйоперанд(t)", "второйоперанд (t)", "предпоследоперанд(t)", "последнийоперанд(t)"позволяют определить вхождения заголовков указанных операндов $\theta_1, \theta_2, \theta_{n-1}, \theta_n$. Операторные выражения "следоперанд(t_1)"и "пред-

операнд(t_1)"позволяют находить вхождения операнда некоторой операции, соответственно, непосредственно следующего за вхождением t_1 операнда той же операции либо предшествующего ему. В концевых случаях, при отсутствии следующего либо предыдущего операнда, выдается само значение t_1. Операторное выражение "операндномер(t n)"имеет своим значением вхождение n - го операнда операции, расположенной по вхождению t. Здесь n - символьный номер (логический символ с номером $260+n$). Если операция имеет меньше чем n операндов, то выдается логический символ 0.

Операторное выражение "заменатермов ($t_1t_2t_3$)" определяет результат замены в терме θ набора вхождений ($\alpha_1 \dots \alpha_n$) расположенных слева направо непересекающихся подтермов (при $n = 1$ рассматривается не набор (α_1), а само вхождение α_1) на набор термов либо символов ($\tau_1 \dots \tau_n$). Здесь t_1 определяет θ, t_2 - вхождения, а t_3 - заменяющие термы. Выражение "исключениеоперанда(t_1t_2)"позволяет находить терм $\varphi(\theta_1 \dots \theta_{i-1}\theta_{i+1} \dots \theta_n)$, получающийся из терма $\varphi(\theta_1 \dots \theta_n)$ исключением операнда θ_i (если $n = 2$, то результатом исключения становится оставшийся операнд $\theta_j, j \neq i$); здесь t_1 определяет вхождение первого символа терма $\varphi(\theta_1 \dots \theta_n)$, а t_2 - вхождение операнда θ_i. Если выражение t определяет вхождение символа в некоторый терм θ, то операторное выражение "базавхождения(t)" имеет своим значением данный терм θ. Для определения переменной с заданным номером применяется выражение "икс(t)", где t определяет N - й после "0"логический символ; N - номер требуемой переменной. Выражение "кортежпеременных(t_1t_2)"определяет набор ($y_1 \dots y_k$) переменных, отличных от переменных набора - значения выражения t_2 и имеющий ту же длину, что и набор - значение выражения t_1. Аналогичной цели служит оператор "новаяпеременная(t_1t_2)", у которого t_1 определяет набор переменных, а выходной переменной t_2 присваивается переменная, не входящая в указанный набор.

Операторы "логсимвол(t)", "переменная(t)"позволяют распознавать логические символы и переменные; операторы "корень(t)", "стандарт(t) распознавать корневое вхождение в терм и вхождение символа, за которым идет открывающая скобка. Последний оператор бывает полезен, чтобы отличить обычное использование символа операции φ в конструкциях вида $\varphi(\theta_1 \dots \theta_n)$ от использования его в качестве синтаксической константы в описаниях вида термов. При обращении к оператору "символ (t_1t_2)" значением выражения t_1 является вхождение в терм либо в набор. Если t_2 - выходная переменная, то ей присваивается объект, имеющий указанное вхождение; в противном случае оператор проверяет совпадение этого объекта со значением выражения t_2. Аналогично, оператор "заголовок(t_1t_2)"либо присваивает выходной переменной t_2 первый символ терма, являющегося значением t_1, либо, если t_2 определено, проверяет совпадение этого заголовка со значением t_2. Операторы "первыйсимвол (t_1t_2)", "второйсимвол(t_1t_2)", "предпоследсимвол(t_1t_2)", "последнийсимвол(t_1t_2)" определяют, соответственно, заголовки (логические символы либо переменные) операндов $\theta_1, \theta_2, \theta_{n-1}, \theta_n$ подтерма $\varphi(\theta_1 \dots \theta_n)$, вхождение первого символа которого задается выражением t_1; если t_2 - выходная переменная, то ей присваивается найденный заголовок, иначе он сравнивается со значением t_2. Эти операторы часто используются при сканировании задачи для идентификации того или иного подтерма. Оператор "свобвхождение(t)"распознает свободное вхождение переменной в терм; оператор "лексикопредшествует(t_1t_2) проверяет лексикографическое предшествование термов. Последний оператор бывает полезен для стандартного упорядочения операндов коммутативных и ассоциативных операций.

Перечислим ряд операторов,используемых для перемещения по вхождениям в терм. Оператор "операнд(t_1t_2)" используется в следующих двух возможных режи-

мах:

а) t_2 определено и имеет значением вхождение α в терм; t_1 - выходная переменная, которой присваивается вхождение той операции, чьим операндом является α. Если такой нет, то оператор ложен.

б) t_1 определено и имеет значением вхождение первого символа подтерма $\varphi(\theta_1 \dots \theta_n)$; $n > 0$. t_2 - выходная переменная, значения которой перечисляют вхождения операндов $\theta_i; i = 1, \dots, n$.

Оператор "новоперанд($t_1 t_2$)"имеет своим входным данным t_1 вхождение операнда некоторой операции. Выходная переменная t_2 перечисляет вхождения операндов той же операции, следующие за t_1 (исключая само t_1).

Для просмотра всех внешних операций,а также всех подряд символов некоторого подтерма служит оператор "подчинено($t_1 t_2$). Существует три возможных режима его реализации:

а) t_1 определено и имеет своим значением вхождение α в некоторый терм. t_2 - выходная переменная, перечисляющая в направлении от α к корню терма все вхождения заголовков содержащих α подтермов (само вхождение α отбрасывается).

б) t_2 определено и имеет своим значением вхождение β заголовка некоторого подтерма; t_1 - выходная переменная, перечисляющая слева направо все вхождения символов в этот подтерм (начиная с заголовка).

в) t_1, t_2 определены и задают вхождения α, β заголовков подтермов τ_1, τ_2. Оператор истинен, если τ_1 лежит внутри τ_2 либо совпадает с τ_2.

Оператор "вхождениетерма($t_1 t_2 t_3$)"имеет входные данные t_1 (терм θ_1) и t_2 (терм θ_2); t_3 - выходная переменная, перечисляющая (слева направо) все вхождения терма θ_2 в терм θ_1. Для просмотра всех антецедентов $f_1, \dots, f_n$ импликативной конструкции "длялюбого($x_1 \dots x_k$ если $f_1 \dots f_n$ то f_0)"применяется оператор "антецедент($t_1 t_2$)", у которого значением t_1 служит указанная конструкция либо вхождение ее заголовка, а выходная переменная t_2 перечисляет вхождения корней утверждений $f_1, \dots, f_n$.

Для сравнения термов,определенных вхождениями своих заголовков, служит оператор "равныетермы($t_1 t_2$)".

Чтобы ускорить поиск в заданном наборе термов A терма, подобного терму t (получающегося из него переобозначением связанных переменных и перестановкой операндов коммутативных операций и симметричных отношений), можно использовать оператор "ключподобия($A\ t\ x$)". Его выходная переменная x перечисляет вхождения в список A тех термов, которые могут оказаться подобными терму t (сравниваются длины термов, их заголовки и суммы кодов входящих в терм логических символов и переменных; код любой переменной здесь равен 1).

Завершает перечень операторов логических структур данных оператор "результподст($t_1 t_2 t_3 t_4$)". Он допускает следующие два возможных режима реализации:

а) t_1, t_2 определяют набор переменных $(x_1 \dots x_k)$ и набор термов $(\tau_1 \dots \tau_k)$ соответственно; t_3 определяет терм θ. t_4 - выходная переменная, которой присваивается терм - результат подстановки термов $\tau_1, \dots, \tau_k$ вместо переменных $x_1, \dots, x_k$ в терм θ.

б) t_1 определяет набор переменных $(x_1 \dots x_k)$; t_3 и t_4 - термы θ и θ'. t_2 - выходная переменная, которой присваивается такой набор термов $(\tau_1 \dots \tau_k)$, подстановка которого в терм θ вместо переменных $x_1, \dots, x_k$ дает терм θ'; если x_i не входит в θ, то τ_i становится равно логическому символу "пустоеслово". При отсутствии указанного набора оператор ложен.

5.2.4 Операторы сетевого представления данных

Операторы сетевого представления данных связаны с рассмотрением наборов и вхождений в них (в сетевых конструкциях вершины,а иногда и ребра кодируются посредством наборов). Операторное выражение "набор($t_1 \dots t_n$)"($n \geq 1$) позволяет формировать набор значений выражений $t_1, \dots, t_n$. Если выражения $t_1, \dots, t_n$ определяют наборы либо символы (т.е. логические символы либо символы переменных), то выражение "конкатенация($t_1 \dots t_n$)" определяет конкатенацию этих наборов и символов; логический символ "пустоеслово"при этом рассматривается как набор длины 0. Выражения "суффикс($t\ \theta$)" и "префикс($t\ \theta$)" позволяют присоединять к концу (соответственно, к началу) набора, определяемого выражением t, значение выражения θ. Операторное выражение "окончание(t)"осуществляет удаление первого элемента набора (пустой набор этим выражением сохраняется); выражение "вставка($t_1t_2t_3$)"имеет своим значением набор ($\alpha_1 \dots \alpha_{i-1}\beta\alpha_i \dots \alpha_n$), полученный из набора ($\alpha_1 \dots \alpha_n$), определяемого выражением t_1, заменой i - го разряда, вхождение которого определяется выражением t_2, на объект β , определяемый выражением t_3. Если значения выражений t_1, t_2 суть набор ($\alpha_1 \dots \alpha_n$) и набор вхождений в этот набор разрядов $\alpha_{i_1}, \dots, \alpha_{i_s}$; $i_1 < \dots < i_s$ (если $s = 1$, то берется не набор вхождений, а само вхождение), то значением операторного выражения "исключение(t_1t_2)"служит результат удаления указанных разрядов из набора ($\alpha_1 \dots \alpha_n$). Выражение "заменаразряда($t_1t_2t_3$)"определяет результат замены в заданном наборе (значение t_1) заданного вхождения разряда (значение t_2) на заданный объект (значение t_3). Если выражения t_1, t_2 определяют наборы α, β, то выражение "пересечениесписков(t_1t_2)"имеет своим значением набор, полученный из α сохранением всех разрядов, встречающихся в β, причем с учетом кратности: число сохраняемых экземпляров одного и того же объекта набора α не превосходит числа его экземпляров в наборе β. Выражение "объединениесписков(t_1t_2)" имеет своим значением набор $\alpha\beta'$, где β' получено из β удалением всех элементов, входящих в α, а выражение "вычеркивание(t_1t_2)" - набор, получающийся из α в результате непродолжаемой последовательности одновременных вычеркиваний у α и β пар одинаковых элементов (порядок просмотра α - слева направо). Выражение "бланк($t_1t_2t_3$)", где t_1 определяет набор длины k, $k \geq 0$, t_2 - некоторый объект α; t_3 - N -й после "0"логический символ ($N \geq 0$), имеет своим значением набор ($\alpha \dots \alpha$) длины $N + k$. Наконец, последнее из серии операторных выражений, используемых для построения новых наборов, - выражение "копия(t)", формирующее копию набора (либо терма).

Выражение "буква(t)"имеет своим значением объект, вхождение которого в набор либо терм определяется выражением t. Если значением выражения t_1 является набор ($\alpha_1 \dots \alpha_n$), а значением t_2 - N - й после "0"логический символ ($N \geq 0$), то значением выражения "левпозиция(t_1t_2)"служит объект α_{N+1}; значением выражения "правпозиция(t_1t_2)" - объект α_{n-N}; значением выражения "окрестность(t_1t_2)" - вхождение $N + 1$ -го разряда в набор ($\alpha_1 \dots \alpha_n$). В этой же ситуации значением выражения "левыйкрай(t_1)" является вхождение разряда α_1; выражения "правыйкрай(t_1)" - вхождение α_n; выражения "начало(t_1)" - объект α_1 и выражения "конец(t_1)" - объект α_n. Если значением выражения t является вхождение разряда α_i в некоторый набор ($\alpha_1 \dots \alpha_n$), то выражение "левсосед(t)"определяет вхождение α_{i-1}, а "правсосед(t)" - вхождение α_{i+1} (если разряд крайний и сдвиг невозможен, то сохраняется значение α_i). Оператор "позиция(t_1t_2)"имеет выходную переменную t_1, перечисляющую слева направо все вхождения разрядов в набор, определяемый выражением t_2. Оператор "входит(t_1t_2)", аналогичный предыдущему, имеет два возможных режима реализа-

ции:

а) t_2 определено и имеет своим значением набор $(\alpha_1 \dots \alpha_n)$; t_1 - выходная переменная, перечисляющая объекты $\alpha_i (i = 1, \dots, n)$.

б) t_1, t_2 определены и имеют значения α, β; β - набор. В этом случае оператор проверяет вхождение объекта α в набор β.

Для поиска в наборе используются: оператор "разряд$(t_1 t_2 t_3)$", где t_1 определяет набор либо терм α; t_2 - логический символ либо переменную φ; t_3 - выходная переменная, перечисляющая все вхождения φ в α, а также операторы "ключ$(t_1 t_2 \theta)$" и "биключ$(t_1 t_2 t_3 \theta)$". У последних двух операторов t_1 определяет набор α; t_2 - логический символ β;t_3 - некоторый объект γ. Выходная переменная θ перечисляет все элементы набора α, имеющие в первом случае вид $(\beta, \dots)$ либо $\beta(\dots)$ либо равные β, а во втором случае - вид $(\beta, \gamma, \dots)$ либо $\beta(\gamma, \dots)$, либо $\beta(\gamma(\dots), \dots)$. Эти операторы часто применяются при поиске комментариев заданного типа, а также при отборе условий либо посылок задачи, имеющих заданный вид. Если выражения t_2, t_3 определяют наборы $(\alpha_1 \dots \alpha_n)$ и $(\beta_1 \dots \beta_m)$, а выражение t_1 - вхождение разряда $\alpha_i, i \le m$, то операторное выражение "соотвпозиция$(t_1 t_2 t_3)$"определяет вхождение разряда β_i. Аналогично, если t_1, t_2 определяют наборы $(\alpha_1, \dots, \alpha_n)$ и $(\beta_1, \dots, \beta_n)$, а t_3 - некоторый объект β, то выражение "таблзначение$(t_1 t_2 t_3)$"либо имеет своим значением объект β_j, такой, что $\beta = \alpha_j$ и $\beta \ne \alpha_1, \dots, \alpha_{j-1}$, либо, при отсутствии указанного объекта, имеет значение β. Оператор "Таблзначение$(t_1 t_2 t_3 t_4)$"позволяет перечислять все элементы t_4 набора t_2, у которых на соответствующей позиции набора t_1 располагается элемент, равный t_3.

Приведем ряд специальных операторов, используемых для перечисляения вхождений в наборы. Если выражения $t_1, \dots, t_n$ имеют своими значениями наборы $(\alpha_{11}, \dots, \alpha_{1m_1}), \dots, (\alpha_{n1}, \dots, \alpha_{nm_n})$ соответственно; $\theta_1, \dots, \theta_n$ - различные переменные, не встречающиеся в $t_1, \dots, t_n$, то оператор "серия$(\theta_1 t_1 \dots \theta_n t_n)$"имеет выходные переменные $\theta_1, \dots, \theta_n$ и перечисляет такие наборы $(\beta_{1i}, \dots, \beta_{ni})$ $(i = 1, \dots, \min(m_1, \dots, m_n))$ их значений, что β_{ji} - вхождение разряда α_{ji} в набор $(\alpha_{j1} \dots \alpha_{jm_j})$; $j = 1, \dots, n$. Просмотр наборов "соответствующих"вхождений справа налево выполняется оператором "контрсерия$(\theta_1\, t_1 \dots \theta_n\, t_n)$", причем $t_1, \dots, t_n$ имеют своими зачениями вхождения разрядов $\alpha_{1j_1}, \dots, \alpha_{nj_n}$ в некоторые наборы $(\alpha_{11}, \dots, \alpha_{1m_1}), \dots, (\alpha_{n1}, \dots, \alpha_{nm_n})$ и оператор перечисляет наборы $(\beta_{1j_1-i}, \beta_{2j_2-i}, \dots, \beta_{nj_n-i}); i = 0, 1, \dots, \min(j_1 - 1, \dots, j_n - 1)$.

Для просмотра разрядов набора начиная с некоторой фиксированной позиции (слева направо) служит оператор "постпозиция$(t_1\ t_2)$". Он допускает два режима реализации:

а) t_1 - выходная переменная; t_2 определяет вхождение разряда α_i в набор $(\alpha_1 \dots \alpha_n)$. Тогда t_1 перечисляет вхождения $\alpha_i, \alpha_{i+1}, \dots, \alpha_n$.

б) t_1, t_2 определены и имеют значения α, β - вхождения в некоторый набор. Оператор проверяет, что вхождение α расположено не раньше, чем β.

Иногда более удобно применять похожий оператор "новпозиция$(t_1 t_2)$", у которого значением t_2 является набор $(\alpha_1 \dots \alpha_n)$ либо вхождение разряда α_i в этот набор, а выходная переменная t_1 перечисляет: в первом случае - все вхождения $\alpha_1, \dots, \alpha_n$, а во втором - $\alpha_{i+1}, \dots, \alpha_n$.

В заключение перечислим ряд проверочных операторов для работы с наборами. Оператор "пересекаются$(t_1 t_2)$"проверяет наличие общего элемента в наборах либо термах (в споледнем случае под элементом понимается логический символ либо символ переменной); операторы "равнойдлины$(t_1 t_2)$"и "короче$(t_1 t_2)$ равенство длин наборов и тот факт, что длина набора t_1 меньше длины набора t_2. Оператор "различны(t)"проверяет попарное отличие друг от друга всех элементов набора. Если значе-

нием выражения t_1 является набор α, а значением выражения t_2 - N-й после "0"логический символ, то оператор "длинатекста(t_1t_2)"истинен, если α имеет длину N, а оператор "длинаменее(t_1t_2) если длина α меньше, чем N.

Наконец, оператор "включается(t_1t_2)"проверяет, что набор - значение выражения t_1 получается перестановкой и исключением части разрядов набора - значения выражения t_2.

5.2.5 Арифметические операторы

Арифметические операторы ЛОСа делятся на две группы: для работы с символьным представлением чисел (число N кодируется N-м после "0"логическим символом) и с обычными десятичными представлениями . В последнем случае цифры $0, \ldots, 9$ кодируются логическими символами "0", ..., "9"; неодноразрядное положительное целое число, имеющее десятичную запись $a_1 \ldots a_n$, кодируется набором логических символов, соответствующих цифрам $a_1, \ldots, a_n$; для представления десятичных дробей внутри этого набора на соответствующей позиции размещается логический символ ","(специальный символ для запятой); для получения отрицательных чисел в начале набора добавляется логический символ "минус". Заметим, что все десятичные числа, кроме цифр, кодируются наборами логических символов, а цифры кодируются самими логическими символами.

Для работы с символьным представлением чисел используются операторные выражения "плюссимв(t_1 t_2)", "вычитсимв(t_1 t_2)", "умножсимв(t_1 t_2)"(сложение, вычитание и умножение). Заметим, что в ЛОСе логический символ "0"имеет номер 260, и логические символы с номерами от 1 до 259 могут использоваться как символьные представления отрицательных целых чисел от -259 до -1. В интерпретаторе ЛОСа предусмотрена возможность для работы с 65535 логическими символами, т.е. наибольшее допустимое в символьном представлении целое число есть 65275. Обычно символьные представления чисел используются для таких технических целей, как нумерация разрядов наборов, представление весов посылок и условий задач, нумерация переменных, задание координат точек на экране, и т.п. Указанные выше диапазоны изменения символьных представлений чисел для этого вполне достаточны.

Если значениями t_1 и t_2 служат переменные, то значением выражения "вычитсимв (t_1t_2)"служит символьное представление разности номеров этих переменных; если значением t_1 является переменная x, а значением t_2 - символьное представление числа n, то значением выражений "плюссимв(t_1t_2)", "вычитсимв(t_1t_2)"служит переменная, номер которой получается, соответственно, увеличением либо уменьшением на n номера переменной x.

Оператор "частноесимв(t_1 t_2 t_3 t_4)"позволяет находить неполное частное t_3 и остаток t_4 от деления числа t_1 на t_2 (все представления здесь - символьные). Оператор "менее(t_1t_2)"истинен, если число t_1 в символьном представлении меньше числа t_2. Для перехода от одного представления чисел к другому служат операторные выражения "симвномер(t)"(логический символ с десятичным номером, определяемым выражением t) и "номерсимв(t)"(десятичный номер логического символа, определяемого выражением t).

Для работы с десятичным представлением служат операторные выражения "минус(t)", "плюс(t_1t_2)", "вычитание(t_1t_2)", "умножение(t_1t_2)", и оператор "меньше (t_1 t_2)". Для реализации деления десятичных чисел применяются программы, написанные на ЛОСе; в этих программах можно использовать вспомогательные операторы "блокделения($t_1t_2t_3t_4$)" (t_1, t_2 - делимое и делитель; выходным переменным t_3, t_4

присваиваются неполное частное и остаток, причем оператор рассчитан только на случай, когда t_1 и t_2 не превосходят 65535) и "шагделения($t_1t_2t_3$)" (t_1, t_2 - целые неотрицательные; $t_1 \leq 1000$; $1 < t_2 \leq 100$; выходной переменной t_3 присваивается целая часть от деления t_1 на $t_2 + 1$). Здесь приведены лишь базисные операторы ЛОСа, непосредственно реализуемые его интерпретатором; для вычислений в десятичных числах на ЛОСе запрограммирована целая серия дополнительных операторов, которые будут указаны в последующих разделах. Отметим, что вычисления в десятичных представлениях на ЛОСе выполняются с "плавающим"числом разрядов; это число ограничено сверху лишь общими требованиями интерпрератора к длине формируемых наборов и может достигать даже тысячи знаков. Приведенные выше базисные операторы сложения, вычитания,умножения и изменения знака выполняются без округления; программно реализованные на ЛОСе вычислительные операторы выполняют округление с сохранением заданного числа знаков после запятой, причем таким образом, что гарантируется получение нижней либо верхней (по выбору) оценки для точного значения.

Для вычислений в машинных форматах "с плавающей запятой"и "целое со знаком"используются группа операторных выражений "выч(...)"и операторов "Выч(...)". Заметим, что кроме непосредственного представления чисел в этих форматах, можно создавать массивы таких чисел. Манипулирование с массивом осуществляется при помощи ссылки на него, которая формально представляет собой некоторый набор длины 2 (как и представление числа в машинном формате, такой набор несколько отличается от обычного набора ЛОСа, и для работы с ним следует применять только указанные выше операторные выражения и операторы "выч","Выч").

Операторное выражение "выч(число a)"позволяет перейти от обычного десятичного числа a (цифры либо набора цифр и знаков "минус","запятая") к формату "с плавающей запятой"; операторное выражение "выч(целое a) к формату "целое со знаком". Обратное преобразование осуществляется операторным выражением "выч(выход a)". Здесь a - число в формате "с плавающей запятой"либо "целое со знаком". В первом случае значением выражения служит пара десятичных чисел (A_1A_2), такая, что рассматриваемое число равно произведению A_1 на 10 в степени A_2 (по мере возможности, A_2 выбирается равным 0). Во втором случае значением выражения является целое десятичное число.

Чтобы извлекать элементы массива, используется выражение "выч(значение M i)". Здесь M - ссылка на массив одного из двух машинных форматов; i - номер элемента массива, представленный в формате "целое со знаком". Значением выражения является представление в машинном формате i-го элемента массива.

Операции над числами в одинаковых машинных форматах реализуются выражениями "выч(плюс a b)", "выч(минус a)"(изменение знака), "выч(вычитание a b)", "выч(умножение a b)", "выч(дробь a b)"(в случае целых чисел берется неполное частное), "выч(степень a b)"(в случае целых чисел b должно быть неотрицательным), "выч(модуль a)". Для вычислений в формате "с плавающей запятой"используются также выражения "выч(логарифм a b)" (a - основание логарифма), "выч(натурлог a)"(натуральный логарифм), "выч(синус a)", "выч(косинус a)", "выч(тангенс a)", "выч(котангенс a)", "выч(арксинус a)", "выч(арккосинус a)", "выч(арктангенс a)", "выч(арккотангенс a)", "выч(квадркорень a)" (квадратный корень), "выч(эксп a)" (экспонента), "выч(гипсинус a)" (гиперболический синус), "выч(гипкосинус a)", "выч(гиптангенс a)", "выч(гипкотангенс a)".

Для получения псевдоконстант при построении графиков используются выражения "выч(плюсбеск)" ($2e99$ - плюс-бесконечность), "выч(минусбеск)" ($-2e99$), "выч(

нет)" ($1e99$) - указатель на пропуск точки при построении графика).

Копирование представления числа в машинном формате выполняется при помощи выражения "выч(копия a)".

Чтобы создать (неинициализированный) массив из n чисел в машинном формате, используется оператор "Выч(набор a n x)". Здесь a - указатель на тип чисел (логический символ "число с плавающей запятой; "целое целое со знаком). Переменной x присваивается ссылка на созданный массив. Для регистрации в массиве m в качестве элемента с номером n (нумерация начинается с 0) числа a в соответствующем машинном формате применяется оператор "Выч(запись m n a)".

Для деления с остатком чисел типа "целое со знаком"служит оператор "Выч(деление m n p q)". Здесь m, n - делимое и делитель; переменной p присваивается неполное частное, q - остаток.

Сравнение чисел осуществляется операторами "Выч(меньше a b)"и "Выч(меньшеилиравно a b)", "Выч(равно a b)". Чтобы проверять отличие числа от 0 , служит оператор "Выч(0 a)"(истинен, если a не равно 0).

Оператор "Выч(изменение a b)"изменяет старое представление числа a, перенося в него значение числа b.

Чтобы ускорить вычисления по цепочке действий, не требующих условных переходов, предусмотрено создание микропрограммы - набора P (как структуры данных ЛОСа) псевдокоманд; такая микропрограмма выполняется за одно обращение к оператору "Выч(программа P A B)". Здесь A - набор вход-выходных данных типа "с плавающей запятой"; B - набор вход-выходных данных типа "целое со знаком". Если вычисления выходят за рамки о.д.з., то оператор ложен. Каждая псевдокоманда предствляет собой набор $(Nn_1 \dots n_k)$, где N - номер операции, $n_1, \dots, n_k$ - номера операндов (входных и выходных). $N, n_1, \dots, n_k$ имеют символьный формат. Номера операндов берутся из набора A либо B, соответствующего типу операнда (такой тип однозначно определяется по номеру операции). Можно выходить за рамки наборов A, B, используя номера, большие их длины (но не большие 200). Такие номера рассматриваются как вспомогательные переменные, которым присваиваются значения, используемые в дальнейших вычислениях по микропрограмме. Нумерация элементов наборов A, B начинается с 1.

Рассматриваются следующие типы псевдокоманд (номер операции совпадает с номером в приводимом списке):

1. Сложение в формате "с плавающей запятой". Первые два операнда - слагаемые, третий - сумма (аналогичное размещение операндов используется и для других операций).
2. Сложение в формате "целое со знаком".
3. Изменение знака числа в формате "с плавающей запятой".
4. Изменение знака числа в формате "целое со знаком".
5. Умножение в формате "с плавающей запятой".
6. Умножение в формате "целое со знаком".
7. Деление в формате "с плавающей запятой".
8. Деление с остатком для формата "целое со знаком".
9. Возведение в степень в формате "с плавающей запятой".
10. Экспонента.
11. Возведение в натуральную степень числа типа "целое со знаком".
12. Модуль в формате "с плавающей запятой".
13. Модуль в формате "целое со знаком".
14. Натуральный логарифм.

15. Логарифм - общий случай.
16. Квадратный корень.
17. Синус.
18. Косинус.
19. Тангенс.
20. Котангенс.
21. Секанс.
22. Косеканс.
23. Арксинус.
24. Арккосинус.
25. Арктангенс.
26. Арккотангенс.
27. Гиперболический синус.
28. Гиперболический косинус.
29. Гиперболический тангенс.
30. Гиперболический котангенс.
31. Сигнум.
32. Тождественная операция (присвоение значения выходной переменной).

5.2.6 Операторы интерфейса

Клавиатура, мышь и меню

Для ввода символов с клавиатуры используется оператор "клавиатура(t)". Здесь t - выходная переменная, которой присваивается логический символ, кодирующий нажатую клавишу. Для того, чтобы определить в процессе написания программы на ЛОСе логический символ, соответствующий той или иной клавише, достаточно, находясь в текстовом редакторе решателя, нажать сначала клавишу F8, а затем - ту клавишу, код которой определяется; при этом с позиции курсора будет прорисовано название соответствующего логического символа. Драйвер клавиатуры, связанный с оператором "клавиатура", имеет следующие регистры:

а) Регистры больших и малых латинских букв;

б) Регистры больших и малых русских букв;

в) Регистры "левый Ctr"и "правый Ctr".

Переключение между латинскими и русскими буквами в этом драйвере выполняется при помощи клавиш F11,F12. Нажатие клавиши F11 переводит драйвер в режим русских букв; F12 - в режим латинских букв. Кроме того, имеется оператор "русшрифт(t_1)", позволяющий переключать драйвер клавиатуры: при t_1 = "1" - на русский шрифт; при t_1 = "0" - на латинский.

При нажатии левой либо правой клавиши мыши оператор "клавиатура(t)"получает логический символ "мышь". Для уточнения данных, введенных при помощи мыши, после получения указанным образом логического символа "мышь", используется оператор "мышь($t_1t_2t_3$)". Его выходные переменные t_1, t_2, t_3 получают в качестве значений, соответственно, указатель нажатой клавиши (логический символ "0" - нажата левая клавиша; "1" - нажата правая клавиша), номер столбца, на котором при нажатии клавиши находился курсор мыши, и номер строки, где он находился. Эти номера имеют символьное представление (т.е. суть логические символы с номерами $260 + N$, где N - номер столбца либо строки).

Возможно обращение через оператор "Меню(...)"к простейшим функциям операционной системы Windows, обслуживающим работу с меню - строкой, прорисовываемой в верхней части экрана. Это меню (далее будем называть его корневым) автоматически создается при запуске решателя, однако вначале является пустым и на экран не выводится. Для формирования новых пунктов корневого меню и его подменю служит операция "меню(плюссимв $t_1\, t_2\, t_3\, t_4\, t_5$)". Здесь t_1 - номер меню (в символьном представлении; для корневого меню равен "0"); t_2 - идентификатор нового концевого пункта (логический символ) либо номер подменю, вводимого в качестве нового пункта; t_3, t_4 - начало и конец в буфере текстов для названия нового пункта меню; t_5 - указатель на то, вводится ли концевой новый пункт ("0") либо новое подменю ("1"). Новые пустые подменю вводятся заблаговременно при помощи операции "Меню(новый t_1)". Выходной переменной t_1 при этом присваивается номер введенного подменю. После того, как корневое меню и все его подменю заполнены, для прорисовки результата на экране применяется операция "Меню(актив)". Расчистка корневого меню и удаление всех его подменю выполняются операцией "Меню(0)". Наконец, для получения входного сигнала после выбора требуемого концевого пункта корневого меню либо некоторого его подменю применяется операция "Меню(вход t_1)". Она присваивает выходной переменной t_1 идентификатор выбранного пункта. Заметим, впрочем, что оператор "клавиатура(t_1)"тоже реагирует на выбор пункта меню присвоением переменной t_1 его идентификатора, и операция "Меню(вход ...)"является избыточной.

Экранные операции

Выдача изображений на экран осуществляется при помощи оператора "видео($M\ t_1$ $\ldots t_n$)", где M - логический символ, определяющий тип экранной операции; $t_1, \ldots, t_n$ - входные данные этой операции и ее выходные переменные. Параметр n меняется в зависимости от M.

Подготовка области экрана к прорисовке изображения выполняется при помощи операции "видео(прямоугольник $t_1t_2t_3t_4t_5$)". Здесь t_1, t_2 - номера столбца и строки для верхнего левого угла прямоугольника; t_3, t_4 - номера столбца и строки для правого нижнего угла (все номера здесь и в рассматриваемых ниже экранных операциях - в символьном представлении). t_5 - логический символ, определяющий цвет прямоугольника. Для указания цвета используются логические символы начиная с "1"до "шестнадцать": 1 - черный; 2 - синий; 3 - зеленый; 4 - зеленовато-голубой; 5 - красный; 6 - пурпурный; 7 - коричневый; 8 - светло-серый; 9 - темно-серый; 10 - светло-синий; 11 - светло-зеленый; 12 - светло-голубой; 13 - светло-красный; 14 - светло-пурпурный; 15 - желтый; 16 - белый. Если t_5 равно "0", то изображение на экране внутри указанного прямоугольника не изменяется. В любом случае после применения операции "видео(прямоугольник ...)"границы прямоугольника становятся границами той области, внутри которой перечисляемыми ниже текстовыми операциями выполняется выдача текста (в частности, автоматически осуществляется переход к новой строке по достижении правой границы).

Для прорисовки группы отрезков применяется операция "видео(отрезок t_1t_2)". Здесь t_1 - набор $a = (a_1 \ldots a_s); a_i = (a_{i1}a_{i2}a_{i3}a_{i4})$, где a_{i1}, a_{i2} - позиция (номера столбца и строки) начала i - го отрезка; a_{i3}, a_{i4} - позиция конца этого отрезка. t_2 - цвет всех выдаваемых на экран отрезков.

Дуга окружности изображается при помощи операции "видео(окружность $t_1 \ldots$ t_9)". Здесь t_1, t_2 - координаты центра окружности (номера столбца и строки); t_3 -

радиус окружности (в символьном представлении); t_4, t_5 - координаты точки, через которую проходит исходный радиус дуги; t_6, t_7 - координаты точки, через которую проходит заключительный радиус дуги; t_8 - направление прорисовки (логический символ "0 против часовой стрелки; "1 по часовой стрелке); t_9 - цвет окружности.

Текстовые фрагменты выдаются на экран решателя при помощи его собственного шрифта, в котором каждый символ представляется матрицей размера 8×19. Интерпретатор ЛОСа нетрудно было бы расширить таким образом, чтобы стало доступным использование стандартных шрифтов, однако при этом потребуется существенная коррекция программ формульного редактора, планирующего компоновку формул с учетом геометрических параметров символьных элементов. В интерфейсе решателя предусмотрена возможность изменения шрифта; кроме того, операторы ЛОСа позволяют вводить и использовать различные наборы шрифтов указанного выше размера. При работе решателя в оперативной памяти создается специальная область (далее называем ее видеокассой), хранящая двоичные матрицы символьных элементов. Изменение содержимого видеокассы выполняется при помощи операции "видео(бланк $t_1 t_2$)", где t_1 - логический символ, определяющий прорисовываемую букву (соответствие - то же, что для драйвера клавиатуры); t_2 - набор $(a_1 \dots a_{19})$; $a_i = (a_{i1} \dots a_{i8})$ - набор из логических символов "0"и "1". Этот оператор заносит двоичную матрицу, определяемую набором t_2, на позицию видеокассы, соответствующую логическому символу t_1. Чтобы прочитать матрицу изображения буквы, хранящейся в видеокассе, применяется операция "видео(развертка $t_1 t_2$)". Здесь t_1 - логический символ, определяющий букву; выходной переменной t_2 присваивается набор длины 19, состоящий из восьмиэлементных наборов логических символов "0"и "1" - требуемая матрица изображения. При запуске решателя видеокасса загружается из вспомогательного файла; для сохранения ее измененной версии в этом файле служит специальный оператор (см. ниже операторы работы с файлами решателя).

При выдаче на экран фрагмента текста начальная позиция либо задается в операциях явным образом, либо используются координаты из указателя текущей позиции, который хранит некоторые значения номеров текущих столбца и строки. Для занесения в указатель текущей позиции новых значений t_1 и t_2 номеров столбца и строки используется операция "видео(позиция t_1 t_2)". Для получения хранящихся в указателе номеров используется операция "видео(точка $t_1 t_2$)". По окончании выдачи текстовой операцией фрагмента текста указатель текущей позиции автоматически устанавливается на первую свободную после фрагмента позицию. Как уже отмечалось выше, при достижении правой границы текущего прямоугольника (т.е. того, который был указан при последнем обращении к операции "видео(прямоугольник ...)") происходит автоматическая смена строки. Каких-либо автоматических вставок знаков переноса (по крайней мере на уровне базисных операторов) при этом не предусмотрено.

Выдача буквы с заданной позиции выполняется операцией "видео(буква t_1 t_2 t_3 t_4 t_5)". Здесь t_1, t_2 - номера столбца и строки позиции, с которой выдается буква; t_3 - логический символ, кодирующий выводимую на экран букву; t_4 - цвет фона; t_5 - цвет символа. Если буква выдается с текущей позиции, то можно положить в качестве t_1 логический символ "продолжение"; в качестве t_2 - логический символ "0".

Для работы с фрагментами текстов введен специальный накопитель, называемый далее буфером текстов. Он представляет собой массив из 2000 ячеек, каждая из которых хранит некоторый экранный символ (фактически - номер этого символа в видеокассе, т.е. число от 0 до 255), а также цвет фона и цвет символа. Буфер

текстов используется для обменов с файлами, хранящими текстовую информацию, а также для временного хранения текстовых фрагментов (например, ранее выданных на экран и требующих в определенных ситуациях повторной выдачи, быть может, с изменением цветовых атрибутов). Большинство из приводимых далее операций, выдающих на экран текстовые фрагменты, автоматически заносят эти фрагменты в заданную область буфера текстов. При ссылке на ячейки буфера текстов они нумеруются числами от 0 до 1999, причем эти номера имеют символьное представление.

Операция "видео (логсимвол $t_1\,t_2\,t_3\,t_4\,t_5\,t_6\,t_7$)"выводит на экран название логического символа t_3. При этом t_1, t_2 суть номера исходных столбца и строки (при выдаче с текущей позиции - "продолжение"и "0"); t_4, t_5 - цвет фона и символов; t_6 - номер первой ячейки буфера, с которой в него заносится название логического символа; t_7 - выходная переменная, которой присваивается номер первой ячейки буфера текстов, идущей после данного названия. Если на экране либо в буфере текстов не хватило места, то оператор ложен. В этом последнем случае прорисовка на экране начала названия логического символа не выполняется.

Для выдачи на экран подтерма заданного терма в скобочной записи используется операция "видео(терм $t_1\,t_2\,t_3\,t_4\,t_5\,t_6\,t_7\,t_8$)". Здесь t_1, t_2 - номера столбца и строки исходной позиции (либо логические символы "продолжение", "0"); t_3 - вхождение первого символа выдаваемого на экран подтерма; t_4 - указатель раскраски. Этот указатель представляет собой набор $(a_0, \ldots, a_s)$, где $a_0 = (a_{01}, a_{02})$ - цвет фона и символов, используемые при прорисовке подтерма; $a_i = (a_{i1}, a_{i2}, a_{i3}), i = 1, \ldots, s$ - указывает, что подтерм рассматриваемого подтерма, вхождение первого символа которого есть a_{i1}, следует раскрасить цветом фона a_{i2} и цветом символов a_{i3}. Вхождения $a_{11}, \ldots, a_{s1}$ упорядочены слева направо, причем соответствующие подтермы могут пересекаться (т.е. быть вложенными друг в друга). Возможность раскраски подтермов существенным образом используется в редакторе программ ЛОСа, так как позволяет создавать многоцветную "указку", выделяющую при просмотре сложных логических конструкций цепочку вложенных термов и облегчающую концентрацию внимания и декомпозицию этих конструкций при чтении программ. Значение t_5 используется в тех случаях, когда весь терм не поместился на выделенную область экрана. Тогда для продолжения выдачи оставшейся части терма (например, после нажатия соответствующей клавиши) входной параметр t_5 полагается равным тому вхождению символа, с которого следует продолжать выдачу подтерма. В исходной ситуации, с начала выдачи подтерма, значение t_5 должно быть равно t_3. t_6 - номер исходной ячейки в буфере текстов, начиная с которой в буфер заносятся выводимые на экран символы. t_7, t_8 - выходные переменные: t_7 - индикатор переноса: если выдача всего подтерма успешно завершена, то он получает значение "0"; иначе - его значением становится вхождение того символа, с которого следует продолжать выдачу на экран. t_8 становится равно номеру первой ячейки в буфере текстов, следующей после заключительного символа подтерма. Если прорисовка терма на экране нежелательна, а требуется лишь зарегистрировать последовательность выводимых на экран символов в буфере текстов и при этом получить некоторую дополнительную информацию о соответствии вхождений в терм позициям буфера текстов, то в указателе раскраски полагается $a_{01} = a_{02}$. В этом случае, если t_1 положить равным номеру некоторой позиции в буфере текстов, то t_7 становится равно вхождению в подтерм логического символа либо переменной, к прорисовке которого относится указанная позиция. Значение t_2 в указанной ситуации игнорируется. Если в буфере текстов не хватило места, то оператор ложен.

Для выдачи набора термов можно использовать операцию "видео(серия $t_1\ t_2\ t_3$

$t_4\,t_5\,t_6$)", которая реализуется быстрее, чем последовательное выполнение операций прорисовки отдельных термов. Здесь t_1, t_2 - номера столбца и строки исходной позиции (либо "продолжение", "0");t_3, t_4 - цвет фона и символов; t_5 - набор термов. Термы этого набора последовательно прорисовываются на экране начиная с исходной позиции; буфер текстов при этом заполняется начиная с нулевой ячейки. Выходной переменной t_6 присваивается набор той же длины, что t_5. На позиции этого набора, соответствующей позиции некоторого терма в наборе t_5, располагается пара номеров исходной и заключительной ячеек буфера текстов, содержащих текст данного терма.

Для выдачи на экран содержимого заданного отрезка буфера текстов используется операция "видео(слово $t_1\,t_2\,t_3\,t_4\,t_5$)". Здесь t_1, t_2 - номера столбца и строки исходной позиции на экране (либо "продолжение", "0"); t_3 - номер ячейки буфера текстов, начиная с которой выдается текст; t_4 - номер ячейки, до которой включительно происходит выдача (если t_4 - логический символ "продолжение", то выдача происходит вплоть до первой ячейки, хранящей "машинный ноль" либо, если такой нет, до конца буфера текстов). Выходная переменная t_5 является индикатором переноса: "0" - нет переноса, иначе - номер ячейки буфера текстов, с которой следует продолжать выдачу.

Операция "видео(запись $t_1\ t_2\ t_3\ t_4$)"позволяет занести в ячейку буфера текстов, имеющую номер t_1, букву t_2 (т.е. t_2- кодирующий эту букву логический символ)с цветом фона t_3 и цветом символа t_4. В случае $t_1 = 0; t_2 = t_3 =$ "1"в указанную ячейку заносится "машинный ноль", который воспринимается рядом операций как указатель на конец используемой части буфера текстов. Операция "видео(значение $t_1\ t_2\ t_3\ t_4$)"присваивает выходной переменной t_2 логический символ, обозначающий букву, расположенную в t_1-й ячейке буфера текстов; при этом значения выходных переменных t_3, t_4 становятся равны, соответственно, цвету фона и цвету символа, хранящимся в ячейке. Операция "видео(0)"заполняет весь буфер текстов "машинными нулями". Операция "видео(пустоеслово t_1t_2)" заполняет "машинными нулями" отрезок буфера текстов начиная с t_1-й ячейки и кончая t_2-й ячейкой. Для изменения цветовых параметров хранящихся в буфере текстов символов (без изменения их изображения на экране) служат операции "видео(актив $t_1t_2t_3t_4$)"и "видео(вхождение $t_1\,t_2\,t_3$)". У них t_1, t_2 - номера первой и последней ячеек изменяемого отрезка буфера текстов. В первом случае цвет фона всех букв этого отрезка заменяется на t_3, а цвет символов - на t_4; во втором случае изменяется на t_3 только цвет фона.

Для определения границ текущего прямоугольника служит операция "видео(лимит $t_1t_2t_3t_4$)", присваивающая своим выходным переменным t_1, t_2, t_3, t_4, соответственно, номера столбца и строки верхнего левого угла прямоугольника и номера столбца и строки правого нижнего угла. Сдвиг изображения внутри текущего прямоугольника выполняется операцией "видео(спуск $t_1\,t_2\,t_3$)". Здесь логический символ t_1 указывает направление сдвига: "префикс" - вниз; "суффикс" - вверх; "правсосед" - вправо; "левсосед" - влево. t_2 - число (в символьном представлении) пикселей, на которые сдвигается изображение; t_3 - цвет чистых полос, возникающих при сдвиге.

Для сохранения текущего экрана в буфере битмэпов используется операция "видео(минус)"; восстановление экрана по содержимому данного буфера выполняется операцией "видео(плюс)", причем буфер битмэпов после этого обнуливается. Для временного сохранения содержимого буфера текстов введены два дополнительных буфера текстов. Операция "видео(копия t_1)" при t_1 равном "0" копирует буфер текстов в первый дополнительный буфер; при "3" - во второй; при "2" - обменивает содержимое буфера текстов и первого дополнительного буфера; при "1" - копирует первый дополнительный буфер в буфер текстов; при "4" - копирует второй допол-

нительный буфер в буфер текстов.

В дополнение к буферу текстов, для сохранения текстовых заготовок в оперативной памяти с целью быстрой выдачи их на экран введен так называемый массив текстов. Главным образом, он используется в текст-формульном редакторе. Размеры массива текстов составляют 56000 ячеек; он позволяет сохранять извлекаемые из буфера текстов фрагменты и возвращать сохраненные фрагменты в буфер текстов для различных экранных операций.

Создание массива текстов осуществляется оператором "видео(начало)"; удаление его - оператором "видео(конец)". Для пересылки в конец массива текстов фрагмента буфера текстов начиная с позиции m и кончая позицией n, используется оператор "видео(образ m n i j)". Его выходным переменным i, j присваиваются, соответственно, символьные номера первой и последней ячейки массива текстов, отведенные для сохраненного фрагмента. Для обратной пересылки служит оператор "видео(прообраз m n k)". Здесь m, n - символьные номера первой и последней ячейки фрагмента массива текстов, k - номер ячейки буфера текстов, начиная с которой размещается фрагмент, извлеченный из массива текстов.

Чтобы расчищать массив текстов, служит оператор "видео(сброс m)", устанавливающий указатель номера первой неиспользованной ячейки этого массива на символьный номер m. Если нужно изменить ранее записанный в массив текстов фрагмент, то применяется оператор "видео(заменавхождения m n k p)". Символьные номера m, n здесь указывают начало и конец изменяемого фрагмента; заменяющий фрагмент берется из буфера текстов - начиная с ячейки k и кончая ячейкой p. Если $p < k$, то происходит просто исключение фрагмента массива текстов. Если длина заменяющего фрагмента не равна длине заменяемого, то выполняется соответствующий сдвиг всех фрагментов массива текстов, расположенных после измененного. Оператор "видео(вычеркивание m n)"исключает фрагмент массива текстов начиная с позиции m до позиции n.

Оператор "видео(См m n k p)"позволяет определять находящуюся в массиве текстов на позиции m букву n с цветом фона k и цветом символа p. Оператор "видео(посылка m n k p)", наоборот, заносит на позицию m массива текстов букву n с цветом фона k и цветом символов p.

Для организации поиска в массиве текстов служит оператор "видео(ключ m n k p)". По заданным начальной и концевой позициям m, n и букве k он перечисляет все номера позиций p, на которых располагается эта буква.

Для прорисовки графиков функций одной переменной используется оператор "видео(график A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9)". Здесь A_1 - ссылка на массив числовых данных в формате с плавающей запятой (ординаты точек графика); A_2 - длина этого массива (десятичное число); A_3, A_4 - нижняя и верхняя границы отображаемого на экране диапазона значений (формат с плавающей запятой); A_5, A_6 - верхняя и нижняя полосы для графика (символьные числа); A_7 - столбец, начиная с которого рисуется график (символьное число); A_8, A_9 - цвет фона и цвет линии. Масштабирование таково, чтобы нижняя и верхняя границы диапазона значений соответствовали нижней и верхней полосам графика. Прямоугольник, выделенный для графика, должен иметь количество столбцов, не меньшее A_2.

Для небольших изображений можно непосредственно на ЛОСе создавать битмэпы, прорисовывать их и сохранять в информационных блоках. Первоначально битмэп формируется в виде набора $(A_1, \ldots, A_m)$ наборов $A_i = (B_{i1}, \ldots, B_{in})$; каждое B_{ij} - логический символ "0"либо "1". Здесь $A_1, \ldots, A_m$ - строки, перечисляемые сверху вниз; в каждой строке элементы перечисляются слева направо. Далее битмэп перекo-

дируется в более компактную структуру данных - некоторый набор логических символов (двоичные знаки выписываются последовательно и делятся на отрезки длины 15; каждый тако отрезок образует номер очередного логического символа, уменьшенный на 1). Такие наборы называем кодами двоичных матриц. Перекодировка выполняется оператором "видео(матрица логсимвол a b)", у которого a - исходное представление битмэпа, b - его код. Обратный переход от кода b к явному представлению битмэпа a осуществляется оператором "видео(матрица набор b m n a)", у которого появляются дополнительные параметры m (число строк) и n (число столбцов). Наконец, собственно прорисовка битмэпа по его двоичному коду b осуществляется оператором "видео(матрица видео p q r s b m n)". Здесь p, q - столбец и строка, определяющие верхний левый край прямоугольника, в котором нужно прорисовать двоичную матрицу; r, s - цвет нулевых и единичных элементов; m, n - число строк и столбцов.

Ряд редко используемых операций "видео(...)"будет указан при описании тех процедур интерфейса, в которых они применяются.

Названия логических символов

В структурах данных решателя логические символы представлены своими номерами (от 1 до 65535), или, точнее, четырехбайтными словами, содержащими эти номера. Для определения номера логического символа по его названию либо названия символа по его номеру используется специальная таблица, хранящаяся в файлах вида Vi.lsi; $i = 1, \ldots, 7$. Каждый такой файл рассчитан на хранение названий 10000 символов. Для хранения названия используется 24-байтное слово, так что предельно допустимая длина названия равна 24. В качестве названия допускаются произвольные последовательности русских либо латинских букв, цифр и спецзнаков. Большие и малые буквы в названии различаются. В начале файла размещаются двухбайтные номера всех логических символов, названия которых он хранит, причем эти номера расположены в лексикографическом для соответствующих названий порядке (эти номера занимают ровно 20000 байт. За ними располагаются сами названия, строго по порядку номеров символов. Если для символа еще не введено название, то соответствующие 24 байта заполнены нулями.

Для работы с названиями логических символов служит оператор "словарь(M t_1 $\ldots$ t_n)". Здесь M - логический символ, определяющий тип выполняемой операции; $t_1, \ldots, t_n$ - набор входных данных либо выходных переменных. Для ввода нового логического символа с заданным названием (точнее, присвоения этого названия некоторому логическому символу, еще не имеющему названия) служит операция "словарь(запись t_1)". Перед обращением к ней в начальном отрезке буфера текстов должно быть сформировано требуемое название, после которого размещается "машинный ноль". Если логический символ с таким названием уже имеется, то оператор ложен; в противном случае выбирается некоторый логический символ, не имевший названия, ему присваивается указанное в буфере текстов название, и выходной переменной t_1 присваивается номер выбранного логического символа. Если не удалось найти символа без названия, то t_1 присваивается "0". Для определения названия заданного логического символа служит операция "словарь(значение t_1)". Если t_1 - логический символ, то в начальный отрезок буфера текстов заносится слово, обозначающее этот символ. При отсутствии названия данного символа первая ячейка буфера текстов обнуливается. Изменение ранее введенного названия логического символа выполняется операцией "словарь(изменение)". Перед обращением к ней в буфер тек-

стов заносится слово вида "A0B0 ...", где A - заменяемое название; B - заменяющее название; 0 - "машинный ноль". Если B уже использовано либо A не использовано, то оператор ложен; в первом случае название A удаляется,но B не вводится. Проверка существования логического символа с заданным названием осуществляется операцией "словарь(проверка t_1)". Если существует логический символ, название которого хранится в начальном отрезке буфера текстов, то выходной переменной t_1 присваивается этот символ, иначе оператор ложен. Для удаления хранящегося в начальном отрезке буфера текстов названия логического символа служит операция "словарь(исключение)".

Наконец, упомянем оператор "кодтекста(t_1 t_2 t_3)", используемый для перевода текстов термов либо наборов термов в сами термы либо наборы термов. Указанный текст размещается в начальном отрезке буфера текстов. Если t_1 есть "0", то этот текст транслируется в терм; если t_1 есть "1", то текст транслируется в набор термов; если t_1 есть "2", то текст транслируется в набор логических символов и термов (т.е. каждый однобуквенный терм рассматривается как символ); если t_1 есть"3", то текст транслируется в единственный логический символ. Если трансляция выполнена успешно, то выходная переменная t_3 получает значение "0", а выходной переменной t_2 присваивается результат трансляции. Если же в тексте обнаружена ошибка, то t_3 указывает на ее тип:

1) t_3 = "2" - после номера переменной идет посторонний символ. Тогда t_2 - ссылка на начало переменной в буфере текстов.

2) t_3 = "3" - номер переменной больше 511. Тогда t_2 - начало переменной в буфере текстов.

3) t_3 = "4" - неправильно набран логический символ; t_2 - его начало в буфере текстов.

4) t_3 = "5" - нехватка места для формирования результата.

5) t_3 = "6" - избыточная правая скобка; t_2 - ссылка на начало текста символа, после которого идет эта скобка.

6) t_3 = "8" - незакрытая левая скобка.

Словарь текстового анализатора

Для работы с текстами естественного языка используется словарь текстового анализатора, хранящий 65535 фрагментов слов (корни, окончания, суффиксы, приставки). Длина одного словарного фрагмента не должна превышать 24 букв. Каждому словарному фрагменту этого словаря (далее называемого внешним словарем) сопоставлен кодирующий его логический символ. При чтении слова оно разбивается на фрагменты, и решателю передаются лишь коды фрагментов.

Для регистрации во внешнем словаре нового словарного фрагмента сначала происходит размещение этого фрагмента в некотором отрезке буфера текстов, и далее выполняется оператор "внешсловарь(запись s m n)", где m, n - номера начальной и последней ячеек отрезка; s - логический символ, который становится кодом фрагмента. Символ s мог уже являться кодом некоторого словарного фрагмента; в этом случае старая версия фрагмента изменяется во внешнем словаре на новую. Чтобы получить словарный фрагмент, закодированный некоторым логическим символом, применяется оператор "внешсловарь(значение s m p q n)". У него s - логический символ, кодирующий фрагмент; m - номер ячейки буфера текстов, начиная с которой размещается словарный фрагмент; p,q - цвет фона и символов для размещаемого в буфере текстов фрагмента. Выходной переменной n передается номер первой ячейки

буфера текстов после фрагмента. Если символ s не являлся кодом словарного фрагмента, то оператор ложен.

Чтобы исключить из внешнего словаря словарный фрагмент, кодируемый логическим символом s, применяется оператор "внешсловарь(исключение s)".

Для чтения слова и разбиения его на словарные фрагменты применяется оператор "поиксслова(m s n)". В качестве входного данного он получает номер m позиции в буфере текстов, начиная с которого размещается еще не прочитанная часть слова. Выходная переменная s перечисляет логические символы, являющиеся кодами словарных фрагментов, расположенных в буфере текстов начиная с позиции m. При этом переменная n перечисляет номера первых идущих после указанных словарных фрагментов позиций буфера текстов. Окончательный отбор разбиения слова на словарные фрагменты выполняется уже с помощью программы, реализованной на ЛОСе.

Имеется возможность просмотра всех словарных фрагментов внешнего словаря, начинающихся с заданной последовательности букв. Для этого служит оператор "текслово(m n s)". Он получает в качестве входных данных номера m, n первой и последней ячеек буфера текстов, между которыми хранится указанная последовательность букв. Выходная переменная s перечисляет коды словарных фрагментов, начинающихся с данной последовательности. Перечисление происходит в лексикографическом порядке.

Информационные блоки

Логическая и текстовая информация, используемая решателем, может быть размещена в специальным образом организованных файлах. Эти файлы разбиты на группы, называемые далее информационными блоками. Информационный блок хранит древовидную конструкцию, образованную объектами следующих типов:

а) Корневой указатель - каталог. Этот объект фактически представляет собой таблицу ссылок на другие объекты информационного блока, длина которой равна числу логических символов, т.е. 65535. Если на i - й позиции этой таблицы находится 0, то ссылка по логическому символу с номером i из данного указателя считается неопределенной, иначе - определен переход из указателя по i-му символу к некоторому объекту информационного блока.

б) Указатель - список. Этот объект представляет собой набор $((a_1, S_1), \ldots (a_n, S_n))$ пар, имеющих попарно различные первые элементы a_i; каждый такой элемент (называемой меткой перехода в данном указателе) представляет собой либо логический символ, либо неоднобуквенный терм. S_i есть набор ссылок на объекты информационного блока, к которым осуществляется переход по метке a_i.

в) Логический терминал. Этот объект представляет собой набор логических символов, переменных и неоднобуквенных термов (однобуквенные термы при записи их в логический терминал автоматически преобразуются в логические символы либо символы переменных).

г) Текстовый терминал. Этот объект представляет собой просто последовательность байт, кодирующих текст (в зависимости от способа хранения текста, между байтами, кодирующими символы, могут вставляться байты, кодирующие их цветовые атрибуты).

В действительности структура ссылок в информационном блоке "почти"древовидная, так как разрешаются ссылки на один и тот же терминал из различных указателей.

Интерпретатор ЛОСа обеспечивает работу с 13 информационными блоками, имеющими номера от 1 до 13. За ними закреплена определенная функциональная нагрузка, которая будет уточняться далее по мере описания структуры и функционирования решателя. Корневым объектом блоков с номерами 1 и 4 является указатель - список; каждый из этих блоков представляет собой единственный файл: для блока 1 это файл I0000.lsi; для блока 4 - файл R0000.lsi. Остальные блоки имеют своим корневым объектом указатель - каталог и состоят из группы файлов. Все файлы одного и того же информационного блока имеют вид Xijkl.lsi, где X - буква, соответствующая номеру блока; ijkl - восьмеричный номер файла в блоке. Нумерация файлов сплошная и начинается с 0000. Указатель-каталог в информационном блоке может быть лишь корневым; он занимает полностью файл X0000.lsi. Буквенная кодировка информационных блоков (кроме уже указанных 1-го и 4-го) такова: E - 2; H - 3; Q - 5; M - 6; W - 7; P - 8; T - 9; A - 10; B - 11; C - 12; D -13. Каждый из файлов информационного блока имеет менее 512 Кбайт; по мере увеличения размеров этих файлов происходит автоматическое разрезание их на две приблизительно одинаковые по размеру части, со сдвигом нумерации последующих за разрезаемым файлов. Все объекты, достижимые из корневого каталога по заданному логическому символу (т.е. "ветвь"этого логического символа в информационном блоке), размещаются в одном и том же файле, так что переходы через указатель - список возможны только в рамках одного файла.

Для работы с информационными блоками используются операторы "файл($M\ t_1 \ldots t_n$)"и "указатель($M\ t_1 \ldots t_n$)". Здесь M - логический символ, определяющий тип выполняемой операции; $t_1, \ldots, t_n$ - набор входных данных либо выходных переменных. Операции применяются к заранее выделенному "активному"информационному блоку. Выделение информационного блока с номером t_1 (номер - в символьном представлении) вместо ранее выделенного осуществляется при помощи операции "файл(актив t_1)". Если информационного блока с таким номером еще не было, то он при этом будет введен (пока без корневого объекта). Проверка наличия информационного блока с номером t_1 выполняется оператором "файл(проверка t_1)". Операция "файл (число t_1)" присваивает выходной переменной t_1 номер активного информационного блока.

Для ссылок на объекты активного информационного блока в ЛОСе служат наборы (a_1, a_2), образованные двумя логическими символами a_1, a_2. Эти символы кодируют некоторым образом (подробнее см. в описании интерпретатора ЛОСа) номер файла информационного блока и смещение в данном файле начала рассматриваемого объекта. Ссылкой на корневой объект произвольного информационного блока служит пара ("0","3").

Для перехода по заданной метке в корневом указателе - каталоге служит операция "указатель(логсимвол $t_1\ t_2\ t_3$)", где t_1 - ссылка на каталог (т.е. набор ("0","3")); t_2 - логический символ, представляющий собой метку перехода. Если переход по данной метке в каталоге не определен, то оператор ложен. Иначе - выходной переменной t_3 присваивается ссылка на объект, к которому имеет место переход по метке t_2.

Для перехода по заданной метке в указателе - списке служит операция "указатель(список $t_1\ t_2\ t_3$)". Здесь t_1 - ссылка на рассматриваемый указатель-список; t_2 - метка перехода (логический символ либо неоднобуквенный терм). Если в данном указателе-списке не предусмотрен переход по метке t_2, то оператор ложен. Иначе выходной переменной t_3 присваивается набор ссылок на объекты, переход к которым осуществляется по данной метке. Заметим, что порядок ссылок в наборе - обратный, т.е. первой идет ссылка, которая была зарегистрирована (с помощью указываемых

далее операций) последней.

Чтобы определить множество всех меток перехода, используемых в указателе-списке, применяется операция "указатель(область $t_1\ t_2$)". У нее t_1 - ссылка на указатель; выходной переменной t_2 присваивается набор всех меток перехода в данном указателе. Ввод нового указателя (каталога либо списка) осуществляется операцией "указатель(новый $t_1\ t_2\ t_3\ t_4\ t_5$)". Здесь t_1 - логический символ "логсимвол" (если вводится корневой каталог - в случае пустого информационного блока) либо "список"(если вводится указатель - список). Если вводится корневой указатель, то $t_2 = t_3 =$ "0"; иначе t_2 - ссылка на внешний указатель, а t_3 - метка перехода, по которой в нем регистрируется новый указатель. В случае ввода каталога t_4 игнорируется, а в случае указателя - списка t_4 есть длина в байтах поля, зарезервированного для нового пустого указателя. При превышении данной длины регистрация в указателе новых ссылок связана с переписыванием в файле всего указателя на новой позиции, а до этого регистрация выполняется без переписывания указателя. Для оценки величины t_4 уточним, что хранение n ссылок по одной метке M требует $3n + 3m + 2$ байт, где m - число логических символов и "самостоятельных"(не идущих непосредственно за логическим символом либо символом переменной) закрывающих скобок в M. Выходной переменной t_5 присваивается ссылка на новый указатель.

Исключение объекта, на который имеется ссылка из некоторого указателя (каталога либо списка), осуществляется операцией "указатель(исключение $t_1\,t_2\,t_3$)". Здесь t_1 - ссылка на данный указатель; t_2 - метка, по которой из него достижим исключаемый объект. Если указатель является каталогом, то t_3 несущественно, иначе t_3 - ссылка на исключаемый объект. Происходит удаление ссылки из указателя по метке t_2 на исключаемый объект. Если этот объект предствлял собой указатель-список, то вся его ветвь автоматически удаляется из информационного блока. Сохраняются лишь те ее терминалы, на которые имеются ссылки из указателей, не относящихся к данной ветви. Если исключаемый объект - терминал, то он удаляется из информационного блока лишь при условии, что на него не было других ссылок.

Ссылка на объект может быть извлечена из одного указателя и перенесена в другой указатель. Это выполняется операцией "указатель(перестановка $t_1\ t_2\ t_3\ t_4\ t_5$)". Здесь t_1 - ссылка на первый указатель (каталог либо список); t_2 - метка, по которой из него имеется ссылка на переносимый объект. Если первый указатель является каталогом, то t_3 несущественно, иначе t_3 - ссылка на переносимый объект. t_4 есть ссылка на второй указатель (заметим, что он может совпадать с первым); t_5 - метка, по которой переносимый объект требуется зарегистрировать во втором указателе. Если указатели и метки совпадают, то просто происходит изменение порядка ссылок по рассматриваемой метке (новая ссылка заносится в начало списка ссылок). Важно заметить, что оператор применим лишь тогда, когда оба указателя относятся к одному и тому же файлу информационного блока (это гарантируется для блоков с номерами, отличными от 1 и 4, лишь при условии, что оба они относятся к ветви одного и того же логического символа в корневом каталоге). Если это не так, то для перенесения объекта (фактически - ветви информационного блока) требуется создать его копию и удалить оригинал.

Если некоторый терминал информационного блока уже создан и необходимо создать дополнительную ссылку на него, то применяется операция "указатель(терминал $t_1\,t_2\,t_3$)". Здесь t_1 - ссылка на указатель, из которого создается дополнительная ссылка; t_2 - метка этой ссылки; t_3 - ссылка на терминал. Как и в предыдущем случае, новый указатель должен быть расположен в том же файле информационного блока, что и терминал.

Для определения типа объекта информационного блока служит операция "указатель(тип t_1 t_2)". Здесь t_1 - ссылка на объект. Выходной переменной t_2 присваивается логический символ, указывающий тип объекта: "логсимвол" - каталог; "список" - указатель-список; "терминал" - текстовый терминал; "терм" - логический терминал.

Если все метки некоторого указателя - списка суть логические символы (как правило, номера в символьном представлении), то для одновременного увеличения либо уменьшения на заданную величину всех его меток, больших или равных заданной, служит операция "указатель(плюссимв t_1 t_2 t_3)". Здесь t_1 - ссылка на указатель; t_2 - метка, начиная с которой происходит смещение номеров; t_3 - логический символ, определяющий константу сдвига. Если он больше "0"(т.е. его номер больше 260), то номера увеличиваются, иначе - уменьшаются. Эта операция позволяет модифицировать указатель без переписывания его в файле.

Перечислим операции, используемые для чтения хранящихся в файле терминалов. Операция "файл(терм t_1 t_2)" по ссылке t_1 на логический терминал присваивает выходной переменной t_2 набор записанных в этом терминале логических символов, символов переменных и неоднобуквенных термов. Если t_1 - ссылка на текстовый терминал, в котором хранится текст с пропущенными цветовыми атрибутами, то операция "файл(набор t_1 t_2 t_3 t_4 t_5)" переносит этот текст в буфер текстов начиная с ячейки данного буфера, имеющей номер (в символьном представлении) t_4. При этом все символы получают одинаковые цвет фона t_2 и цвет символов t_3. Выходной переменной t_5 присваивается номер последней использованной при чтении ячейки буфера текстов. Если же текстовый терминал хранит текст с явно указанными цветовыми атрибутами, то для его чтения используется операция "файл(слово t_1 t_2 t_3)", где t_2 - номер начальной ячейки буфера текстов; t_3 - выходная переменная, которой присваивается номер последней использованной ячейки. Если текстовый терминал по ссылке t_1 хранит таблицу видеокассы, то для загрузки ее в видеокассу используется операция "файл(видео t_1)".

Ввод нового терминала осуществляется операцией "файл(запись t_1 t_2 t_3 t_4 t_5)". Здесь t_1 - логический символ, определяющий тип вводимого объекта: "терм" - логический терминал; "слово" - текстовый терминал с явно указанными цветовыми атрибутами; "набор" - текстовый терминал с пропущенными цветовыми атрибутами; "видео" - текстовый терминал, хранящий таблицу видеокассы. Если создается логический терминал, то t_2 - набор логических символов, символов переменных и термов (допускаются и однобуквенные термы, однако в терминал они записываются как символы, и при чтении из терминала восстанавливаются как символы). В остальных случаях значение t_2 несущественно. Содержимое нового текстового терминала извлекается из буфера текстов начиная с нулевой ячейки и кончая указателем конца текста ("машинный нуль") либо концом буфера. t_3 - ссылка на указатель (каталог либо список), в котором регистрируется новый терминал; t_4 - метка, по которой он регистрируется. Выходной переменной t_5 присваивается ссылка на новый терминал.

Изменение содержимого терминала выполняется при помощи операции "файл (изменение t_1 t_2 t_3 t_4)". Здесь t_1 - логический символ, определяющий тип терминала таким же образом, как в операции "файл(запись ...)"; если изменяется логический терминал, то t_2 задает новый набор символов и термов, иначе t_2 несущественно. Как и при вводе терминала, информация для изменения текстовых терминалов берется из начального отрезка буфера текстов. t_3 - ссылка на изменяемый терминал. Выходной переменной t_4 присваивается ссылка на измененный терминал. Так как изменение терминала связано с переписыванием его новой версии в конце файла (старая версия снабжается специальной пометкой, причем ее поле "портится" - используется для

хранения ссылки на новую версию), то при последующей работе с терминалом следует использовать только ссылку t_4.

При изменении указателей и терминалов, хранящихся в информационном блоке, старые их версии (снабженные специальными пометками) накапливаются и приводят к дополнительному увеличению размеров файла. Кроме того, так как переход к новой версии объекта осуществляется по цепочке ссылок, хранящихся в старых его версиях, может несколько замедляться работа программ. Поэтому предусмотрена процедура "уплотнения"информационного блока, переписывающая измененные его файлы без старых версий объектов и выполняюшая необходимую коррекцию всех ссылок (смещений в фале) из указателей. Эта же процедура, при усмотрении превышающего 450 Кбайт файла информационного блока (кроме блоков 1 и 4), осуществляет разрезание его на две примерно равные части, с увеличением на 1 номеров последующих файлов данного блока. Обращение к указанной процедуре выполняется операцией "файл(уплотнение)". После выполнения ее информационный блок перестает быть активным. Если t_1 - ссылка на некоторый объект информационного блока (кроме корневого каталога), который был изменен, и таким образом содержит ссылку на новую его версию, та - ссылку на ее новую версию, и т.д., то ссылка на концевой объект данной цепочки (т.е. на "реальную"текущую версию объекта) присваивается операцией "файл(ссылка t_1 t_2)"выходной переменной t_2. Операция "файл(выписка t_1)"присваивает выходной переменной t_1 набор номеров всех информационных блоков, для которых, возможно, требуется уплотнение (т.е. в некотором файле блока имеется хотя бы одна "отключенная"версия объекта).

Блок программ ЛОСа

Программа решателя, записанная на ЛОСе, хранится в группе файлов, называемой далее блоком программ. Эти файлы имеют вид Lijkm, где i,j,k,m - восьмеричные цифры. Файл L0000 хранит каталог программ - в нем для каждого логического символа, имеющего свою программу, указана ссылка на корневой фрагмент данной программы. Такая ссылка состоит из номера файла Lijkm (отличного от L0000) и смещения в этом файле начала фрагмента (более подробно формат данных в файлах блока программ приведен в описании интерпретатора ЛОСа). Все фрагменты программы одного и того же логического символа хранятся в одном файле блока программ. Так как программы языка ЛОС реализуются интерпретатором, то существенно облегчено изменение программой своих собственных фрагментов непосредственно в процессе работы решателя. Разумеется, здесь должны соблюдаться определенные ограничения на изменение тех фрагментов программ, которые на текущий момент "активизированы", т.е. начато их выполнение и в стэках интерпретатора хранятся значения программных переменных, связанных с этими фрагментами. Однако, эти фрагменты продублированы в оперативной памяти, так что даже их при определенных условиях можно изменить. Например, реализованный на ЛОСе программный редактор позволяет изменять все без исключения фрагменты программ, в том числе свои собственные (последние изменения требуют особой аккуратности, во избежание необратимой порчи блока программ). Для обеспечения возможности восстановления файлов решателя при аварийных ситуациях в его интерфейсе предусмотрена возможность копирования полного комплекта его файлов в резервную директорию. Перед копированием выполняется проверка корректности структур данных в этих файлах (как в блоке программ, так и в информационных блоках); при обнаружении нарушений происходит выход в операционную систему, а копирование не выполня-

ется.

Для работы с фрагментами программ в ЛОСе используется оператор "прогфайл $(M\, t_1 \ldots t_n)$". Здесь M - логический символ, определяющий тип выполняемой операции; $t_1, \ldots, t_n$ - набор входных данных либо выходных переменных. Ссылка на фрагмент программы в блоке программ представляет собой пару логических символов. Более подробно структура таких ссылок будет указана ниже, при описании интерпретатора (она несущественна при использовании приводимых далее операций). Операция "прогфайл(логсимвол $t_1\, t_2\, t_3$)"позволяет по заданному логическому символу t_1 находить ссылку (t_2, t_3) на начало корневого фрагмента программы символа t_1; если такой программы нет, то оператор ложен. Операция "прогфайл(значение $t_1\, t_2\, t_3$)"по ссылке (t_1, t_2) на фрагмент в блоке программ присваивает выходной переменной t_3 набор операторов этого фрагмента (каждый оператор, в том числе односимвольный, представляется термом). Операторы "ветвь N"и "иначе N"в данном наборе представлены как "ветвь$(a_1 a_2)$"; "иначе$(a_1 a_2)$", где (a_1, a_2) - ссылка на тот фрагмент, переход к которому определяется оператором.

Изменение ранее введенного фрагмента программы либо ввод нового фрагмента выполняются операцией "прогфайл(запись $t_1 t_2 t_3 t_4 t_5 t_6$)". Здесь t_4 - набор операторов (термов), образующих новый фрагмент. В нем могут встречаться в указанном выше формате ссылки на другие фрагменты программы (напомним, что повторные ссылки на один и тот же фрагмент программы не разрешаются, так что те фрагменты, на которые имеются ссылки из нового фрагмента, либо должны быть непосредственнымии подфрагментами заменяемого фрагмента ,либо должны быть заранее "отключены"при помощи операции "прогфайл(вычеркивание ...)"; см. ниже). Кроме того, допускаются термы вида "ветвь(m)"; "иначе(m)", у которых m - логический символ, представляющий собой метку недоопределенного перехода из фрагмента. Такие недоопределенные переходы, после записи их в блок программ, восстанавливаются в том же виде операцией "прогфайл(значение ...)". Если новый фрагмент заносится в качестве корня программы некоторого логического символа f (возможно, уже имевшейся до этого), то t_1 есть символ "0"; $t_2 = f$; значение t_3 несущественно. Иначе t_1 равно "1"; (t_2, t_3) есть ссылка на вхождение оператора "ветвь" либо "иначе" в тот внешний фрагмент, который через данный оператор должен ссылаться на новый фрагмент. Под ссылкой на вхождение оператора перехода "ветвь" либо "иначе" в некоторый фрагмент здесь понимается пара (A_1, A_2), у которой A_1 - ссылка на фрагмент (пара логических символов); A_2 - номер (начиная с 1) данного оператора перехода в фрагменте. Если новый фрагмент F заносится вместо некоторого другого фрагмента G, то G и все ветви фрагмента G, не упомянутые в F, удаляются. По окончании регистрации в блоке программ нового фрагмента выходным переменным t_5, t_6 присваивается ссылка на этот фрагмент.

Если фрагмент F и всю его ветвь требуется временно отключить для последующего использования в другом месте программы, то применяется операция "прогфайл (вычеркивание $t_1\, t_2\, t_3\, t_4$)". Если указанный фрагмент F является корнем программы некоторого логического символа f, то t_1 есть "0"; $t_2 = f$; t_3 - несущественно. Иначе t_1 равно "1"; (t_2, t_3) - ссылка на вхождение оператора "ветвь"либо "иначе"во внешний фрагмент, по которому происходит переход к фрагменту F. Если t_1 есть "0", то из каталога программ исключается ссылка по символу f; иначе - ссылка из внешнего фрагмента на F заменяется меткой t_4 недоопределенного перехода (логическим символом). В обоих случаях исключаются только ссылки на фрагмент F, а сам он и вся его ветвь сохраняются в блоке программ без изменений.

Для удаления ветви ранее отключенного фрагмента программы используется опе-

рация "прогфайл(исключение t_1t_2)". Здесь (t_1, t_2) - ссылка на данный фрагмент. Этот фрагмент и все достижимые из него фрагменты снабжаются в блоке программ специальными пометками. Аналогичным образом, операция "прогфайл(сброс t_1)" уничтожает всю программу логического символа t_1. Операция "прогфайл(корень $t_1\ t_2\ t_3$)"подключает ранее отключенную ветвь фрагмента, ссылкой на который является пара (t_2, t_3), в качестве программы логического символа t_1 (ранее имевшаяся программа этого символа удаляется).

При замене фрагмента программы на новую его версию старая версия снабжается специальной пометкой, но сохраняется в файле блока программ. Для того, чтобы периодически "расчищать" блок программ от накапливающихся в нем неиспользуемых отрезков, используется операция "прогфайл(уплотнение)". Она инициирует перезапись фрагментов программ без сохранения указанных отрезков, с соответствующей коррекцией ссылок между фрагментами. Так как при этом полностью нарушается корректность "старых"ссылок из регистров интерпретатора и из хранящихся в оперативной памяти копий фрагментов программы на фрагменты в блоке программ, то после указанного уплотнения блока программ происходит автоматический перезапуск решателя (это выглядит как возвращение к исходному меню). Заметим, что процедуре уплотнения подвергаются не все файлы блока программ, а лишь те из них, которые были изменены после последнего уплотнения. Как и при уплотнении информационных блоков, происходит разрезание пополам тех файлов, которые превысили 450 Кбайт. Заметим, что процедура уплотнения сначала предпринимает проверку корректности блока программ, и лишь после этого начинает собственно уплотнение. При обнаружении ошибки эта процедура выдает значение "истина" (в отличие от процедуры уплотнения информационных блоков), а в случае успешного уплотнения - значение "ложь".

Если произошло изменение программы некоторого логического символа, причем перезапуск решателя нежелателен (например, в цикле логического вывода и автоматической генерации приемов), то для устранения рассогласования между копиями фрагментов программы, хранящимися в оперативной памяти, и измененными фрагментами в блоке программ, используется операция "прогфайл (компонента t_1)". Эта операция удаляет из оперативной памяти все копии фрагментов программы логического символа t_1 и восстанавливает в копии каталога программ, хранящейся в оперативной памяти, ссылку на корневой фрагмент этой программы по блоку программ. Операция "прогфайл(пересмотр)"уничтожает вообще все хранящиеся в оперативной памяти решателя копии фрагментов программ и реализует его перезапуск.

Приведем в заключение ряд достаточно редко используемых операций над блоком программ. Операция "прогфайл(позиция $t_1t_2t_3t_4t_5t_6$)"по ссылке (t_1, t_2) на фрагмент программы; набору t_3 операторов этого фрагмента и вхождению t_4 в этот набор оператора "ветвь" либо "иначе" присваивает выходным переменным t_5, t_6 ссылку на вхождение данного оператора в рассматриваемый фрагмент (см. выше формат таких ссылок при описании операции "прогфайл(запись ...)"). Эта операция сохранилась от старой версии интерпретатора, в которой формат ссылок был иным; требуемую ссылку теперь легко получить простым подсчетом количества операторов перехода, предшествующих данному. Операция "прогфайл(продолжение $t_1\ t_2\ t_3$)"используется для поиска недоопределенных переходов. У нее t_1 - логический символ, в программе которого ищутся такие переходы. Находится первый такой переход "ветвь А" либо "иначе А"; здесь А - логический символ, являющийся меткой недоопределенного перехода. Определяется путь $(F_1, \ldots, F_s)$ от корневого фрагмента F_s программы символа t_1 к тому фрагменту F_1, в котором найден указанный переход (т.е. каждый

фрагмент F_{i+1} имеет ссылку на F_i; $i = s-1, \ldots, 1$). Выходной переменной t_2 присваивается метка A; переменной t_3 - набор $(a_1, \ldots, a_s)$ пар логических символов - ссылок на фрагменты $F_1, \ldots, F_s$. Операция "прогфайл(имя t_1 t_2 t_3)"позволяет по заданным логическому символу t_1 и оператору t_2 (терму) находить первый фрагмент программы символа t_1, содержащий данный оператор. При этом выходной переменной t_3 присваивается такой же набор ссылок, определяющих путь к найденному фрагменту, как и для предыдущей операции.

Операции трассировки

Для отладки программ языка ЛОС используется программа логического символа "прерывание", написанная на ЛОСе. Эта программа реализована как программа фиктивного операторного выражения (именно, однобуквенного терма "прерывание"), причем данное операторное выражение в явном виде в программах ЛОСа не встречается, а обращение к указанной программе выполняется интерпретатором автоматически - при обнаружении тех или иных ошибочных действий либо при выполнении условий обрыва, заданных используемым режимом трассировки. Программа символа "прерывание" реализует обычные операции отладчика - позволяет устанавливать требуемый режим трассировки, просматривать программу текущего и внешних операторов, определять значения программных переменных, и т.п. Более подробно об этой программе будет рассказано в специальном разделе данного описания. Для использования в отладчике был введен ряд базисных операций ЛОСа, объединенных в операторе "трассировка($M\, t_1 \ldots t_n$)". Эти операции существенным образом используют специфику применяемого интерпретатора ЛОСа, поэтому их описание будет приведено в разделе, посвященном программе отладчика ЛОСа.

Глава 6

Библиотека вспомогательных операторов ЛОСа

В этой главе мы приведем перечень реализованных в логической системе вспомогательных операторов, применение которых существенно упрощает программирование на ЛОСе. Эти операторы разбиваются на группы, аналогичные тем, на которые были разбиты базисные операторы ЛОСа: операторы для работы с наборами и вхождениями в наборы; операторы для работы с логическими структурами данных; операторы для работы с задачами; арифметические операторы и операторы интерфейса. Особо крупные либо узкоспециализированные операторы вынесены в другие разделы, в которых описывается программная реализация основных блоков логической системы.

К числу простейших реализованных на ЛОСе операторов общего характера относится перечисляющая версия оператора присвоения - "Равно(xt)". Этот оператор присваивает переменной x значение t, при этом формально является перечисляющим. Он применяется в тех случаях, когда для подсчета количества сбрасываемых внешних перечислений нужно добиться определенности в режиме работы некоторого смешанного проверочно-перечисляющего оператора. Например, оператор "альтернатива(P Равно(xt) A)", если A реализуется в режиме перечисления, оказывается перечисляющим вне зависимости от истинности условия P.

6.1 Операторы для работы с наборами и вхождениями в наборы

Чтобы получить набор, полученный из заданного набора наборов A переупорядочением элементов по невозрастанию либо неубыванию их длины (под длиной набора понимается число его разрядов), применяются операторные выражения "убываниедлин(A)"и "возрастаниедлин(A)". Результат циклического сдвига набора A на заданное число n разрядов вправо (n - в формате символьного числа) дается операторным выражением "сдвигнабора(A n)".

При замене переменных в некотором терме часто приходится прибегать к преобразованию списка A "новых"переменных к виду списка однобуквенных слов (термов), образованных этими переменными. Такое преобразование позволяет далее воспользоваться оператором "результподст(...)". Чтобы получить указанный список слов, используется операторное выражение "наборслов(A)".

При накоплении списка объектов в некотором вычислительном цикле может оказаться полезным операторное выражение "Суффикс(A t)", которое присоединяет

элемент t к концу накопителя A только в том случае, если он отсутствует в накопителе.

Если из набора A нужно исключить элементы, список вхождений которых в A есть s, причем эти вхождения не обязательно упорядочены слева направо, то применять базисное операторное выражение "исключение(A s)"нельзя. В этом случае используется выражение "Исключение(A s)".

Если нужно получить набор, возникающий при замене в наборе A объекта, расположенного по вхождению v, на объект t, то это можно делать без ссылки на набор A (как того требует выражение "заменаразряда(...) достаточно воспользоваться выражением "измпозиции(v t)".

Для получения набора объектов из набора A вхождений этих объектов применяется операторное выражение "объектывхождений(A)".

Если A - набор наборов, то выражение "Конкатенация(A)"имеет своим значением конкатенацию всех наборов набора A (в порядке их расположения в наборе A).

В тех случаях, когда нужно вставить перед заданной позицией набора не отдельный элемент, а сразу набор элементов, применяется выражение "вставканабора(A v B)". Здесь v - вхождение в набор A, перед которым выполняется вставка набора B.

Для получения номера разряда набора, для которого указано его вхождение a в набор, используется операторное выражение "номервхождения(a)". Его значением служит символьное число, равное числу вхождений, предшествующих a (не включая самого a). Для определения номера первого вхождения элемента b в набор a (нумерация начинается с 1) служит операторное выражение "номерэлемента(b a)". В том случае, когда b не входит в a, выдается 0.

В дополнение к описанным ранее операторам перечисления элементов набора, вводится оператор "четпозиция(a b)", перечисляющий вхождения a в набор b разрядов с четными номерами (нумерация начинается с 1; перечисление ведется по возрастанию номеров).

Если нужно найти первый элемент x набора A, имеющий своим заголовком логический символ s либо совпадающий с s, не перечисляя все такие элементы набора, то применяется оператор "первыйключ(A s x)".

Число разрядов набора A определяется с помощью операторных выражений "длинанабора(A)"и "Длинанабора(A)". Первое из них выражает длину набора в формате десятичного числа, второе - в виде символьного числа.

Для проверки того, что набор A имеет элемент, равный логическому символу либо символу переменной a, либо начинающийся с этого символа, используется оператор "Входит(a A)". Для проверки того, что набор состоит из символов переменных, служит оператор "Переменные(A)".

Иногда бывает нужно не просто скопировать некоторый набор A, но скопировать также и все наборы и термы, достижимые из него по цепочкам ссылок. В этом случае применяется операторное выражение "Копия(A)". Такое копирование рекомендуется применять, например, при создании вспомогательной задачи, чтобы изменения ее комментариев в процессе решения не приводили к изменениям в той задаче, из которой она возникла.

6.2 Операторы для работы с логическими структурами данных

Логические символы, встречающиеся в формулировках задач, распределены по разделам, причем названия разделов тоже суть некоторые логические символы. Справочник "содержание"позволяет получать по названию раздела список всех относящихся к нему логических символов и названий его подразделов; справочник "раздел" позволяет получать по логическому символу название раздела, к которому тот относится. Если нужно получить список всех относящихся к разделу a либо (по транзитивности) его подразделам, используется операторное выражение "Содержание(a)".

В списке символов раздела обычно принимается такое их упорядочение, при котором более "простые"в эвристическом смысле понятия и подразделы размещаются ближе к началу списка. Это упорядочение можно использовать при выборе того логического символа, за которым закрепляется некоторый прием либо теорема (в базе приемов либо теорем) - как наименее простого из встречающихся в нем символов. Здесь полезно операторное выражение "разделысимволов(a)", которое по набору логических символов a выбирает наименее простой (согласно упорядочению разделов и логических символов в разделах) символ $A1$. Значением данного выражения становится набор $(A_n \dots A_1)$, где A_{i+1} - раздел, к которому относится A_i; A_n - раздел, не имеющий надраздела. Если ни один из символов списка a не имеет надраздела, то значение выражения равно 0.

Все рассматриваемые в задачах объекты отнесены к некоторым типам, обозначенным специальными логическими символами (например, "число", "точка", "множество", "функция", и т.п.). Объекты типа A могут разбиваться на подмножества (не обязательно непересекающиеся) объектов типов $B_1, \dots, B_n$, которые считаются подтипами типа A. Например, для типа "число"имеются подтипы "целое", "рациональное", "двоичное", и т.п. Список типов логического символа A определяется с помощью справочника "тип"(обращение к нему имеет вид "справка(тип A)"). Эти типы в списке упорядочены по принципу расширения класса объектов при движении слева направо. Справочник "род"позволяет получать список всех надтипов заданного типа (при отсутствии таковых выдается 0). Справочник "типданных"позволяет по вхождению a заголовка f операции $f(A_1 \dots A_n)$ и вхождению b операнда A_i находить тип объектов, допустимых в качестве значения этого операнда. Он используется при автоматической обработке формулировки новой задачи, для пополнения ее ограничениями на область допустимых значений; обращение к нему имеет вид "справка (типданных f a b)".

6.2.1 Перечисляющие операторы

С каждым вхождением в задачу некоторого терма t связана область этого вхождения - совокупность всех утверждений задачи, в контексте истинности которых рассматривается данный терм. Если терм входит в условие задачи, то к такой области относятся все посылки задачи и все остальные ее условия; если он входит в посылку задачи, то к ней относятся все остальные посылки задачи. Кроме того, к области вхождения могут относиться подутверждения из того же условия (посылки), что и рассматриваемый терм t. Это бывает в следующих случаях:

а) t расположено внутри утверждения A_i конъюнкции "и($A_1 \dots A_n$)". Тогда к области вхождения относятся $A_1, \dots, A_{i-1}, A_{i+1}, \dots, A_n$.

б) t расположено внутри утверждения A_i из квантора общности "длялюбого(x_1 $\ldots x_m$ если $A_1 \ldots A_n$ то A_0)"; $i = 0, \ldots, n$. Тогда к области вхождения относятся утверждения A_j при всех $j = 1, \ldots, n$, отличных от i.

в) t расположено внутри выражения вида "$F(x_1 \ldots x_m$ и$(A_1 \ldots A_n)$ $T)$", где F - один из символов "отображение", "выписка", "перечисление", "суммавсех". Тогда к области вхождения относятся все A_i при $i = 1, \ldots, n$, кроме того (если такое есть), внутри которого расположено рассматриваемое вхождение t.

г) t расположено внутри выражения "вариант(A P Q) внутри P либо Q. В первом случае к области вхождения относятся все конъюнктивные члены утверждения A, во втором случае - отрицание утверждения A.

Оператор "обл(t_1 t_2 t_3 t_4 t_5)"позволяет перечислять утверждения, относящиеся к заданной области вхождения. У него t_1,t_2,t_3 - координата вхождения в задачу t_4 (т.е. вхождение подтерма в терм задачи, вхождение терма задачи во внешний набор и указатель посылки (0) - условия (1)). Выходная переменная t_5 перечисляет указанные утверждения. Возможны специальные случаи обращения: если t_4 равно 0, то перечисляются только относящиеся к тому же терму, что и t_1, вхождения первых символов утверждений из области вхождения t_1. Если t_4 равно 1, вместо вхождений здесь перечисляются сами утверждения. В обоих случаях при обращении полагается $t_2 = t_3 = 0$. Заметим, что фактически оператор "обл(...)"может включать в перечисление, кроме указанных выше, еще ряд подутверждений содержащего t терма задачи. Эти ситуации (часть из них связана с работой текстового анализатора) весьма редкие; информация о них без труда извлекается из текста программы данного оператора.

Чтобы перечислять вхождения переменных из связывающей приставки квантора либо описателя, используется оператор "связка(a b)". У него a - вхождение первого символа подтерма $f(t_1 \ldots t_n)$, представляющего собой квантор либо описатель. Выходная переменная b перечисляет слева направо вхождения операндов t_i до тех пор, пока эти операнды суть символы переменных.

Для перечисления конъюнктивных членов некоторой (возможно, вырожденной) конъюнкции используется оператор "конъюнктчлен(a b)". Здесь a - вхождение первого символа утверждения A. Выходная переменная b перечисляет: если заголовок утверждения A отличен от символа "и", то - единственное значение, равное a, иначе - вхождения первых символов операндов утверждения A. Аналогичным образом, для перечисления дизъюнктивных членов некоторой дизъюнкции служит оператор "дизъюнктчлен(a b)". Отличием здесь является то, что формат входного данного a и значения выходной переменной b - не вхождение, а терм. Выходная переменная b перечисляет: если заголовком терма a служит логический символ "или", то - все термы, являющиеся операндами этого "или", иначе - единственное значение , равное терму a.

Если нужно перечислить все максимальные (имеющие вхождение, не расположенное внутри другого подвыражения) подвыражения некоторого терма, то применяется оператор "корневоевхождение(a b)". У него a - вхождение первого символа того подтерма, в котором расположены искомые подвыражения; b - выходная переменная.

Антецеденты A_i $(i = 1, \ldots, n)$ квантора общности "длялюбого($x_1 \ldots x_m$ если A_1 $\ldots A_n$ то A_0)"можно менять местами с консеквентом A_0, переходя при этом к их отрицаниям. При идентификации этого квантора может возникнуть необходимость учитывать такие возможные перестановки, а также возможное преобразование квантора общности в отрицание квантора существования. Для этого используется опера-

тор "импликант(a b c)". У него a - вхождение первого символа квантора общности либо существования; b - 0 либо 1. Выходная переменная c перечисляет вхождения первых символов f утверждений A_i вида "$f(\ldots)$" либо "не($f(\ldots)$)", которые при $b = 0$ можно рассматривать (с точностью до указанных перестановок) как антецедент, а при $b = 1$ - как консеквент указанного квантора общности (в случае квантора существования, соответственно, рассматривать как конъюнктивный член подкванторного утверждения либо как его отрицание).

Чтобы упростить идентификацию операндов двуместной коммутативной операции либо отношения, используется оператор "Операнды(a b c)". Здесь a - вхождение указанной операции. Выходные переменные b,c поочередно становятся равны: сначала вхождениям первого и второго операндов a, затем - вхождениям второго и первого. При идентификации операций со специальной симметрией может понадобиться более сильная возможность перечисления перестановок набора операндов, обеспечиваемая оператором "сдвиг(a b c d)". У него a - вхождение подтерма $f(A_1 \ldots A_n)$; b - набор номеров операндов этого подтерма. Выходная переменная c перечисляет наборы вхождений операндов, полученные из набора вхождений $(A_1 \ldots A_n)$ циклическими перестановками внутри подмножества операндов b (на очередном шаге первый элемент подсписка операндов переносится в конец подсписка). d - величина сдвига в текущей перестановке (символьное число).

Для перечисления вхождений надтермов заданного терма имеется базисный оператор "подчинено(...)". Однако, он исключает из перечисления вхождение самого терма. Если это вхождение нужно включать в перечисление, берется оператор "Подчинено(a b)". У него a - вхождение терма; b - выходная переменная.

Если имеется набор термов либо логических символов, то для перечисления его элементов, преобразованных к формату терма, используется оператор "термнабора(a b)". Здесь a - заданный набор, b - выходная переменная.

В процедурах логического вывода теорем используется процедура унификации термов. Она получает в качестве входного данного набор пар термов $((p_1, q_1), \ldots (p_n, q_n))$ и набор переменных $(x_1, \ldots, x_m)$, входящих в эти термы. Процедура пытается найти такой набор термов $(r_1, \ldots, r_m)$, что подстановка этих термов вместо переменных $x_1, \ldots, x_m$ в термы p_i, q_i $(i = 1, \ldots, n)$ отождествляет последние. Под отождествлением понимается возможность преобразовать один терм в другой путем изменения порядка операндов коммутативных операций, а также, быть может, некоторых других простых тождественных (либо эквивалентных) преобразований, уточняемых при обращении к процедуре унификации. Например, здесь может использоваться ассоциативность операции либо тождество $f(f(x)) = x$. Допущение такого рода преобразований делает унификацию, вообще говоря, неоднозначной (в отличие от классического случая "без преобразований"); более того, при унификации могут возникать дополнительные переменные. Например, при унификации двух произведений ab,cd, где a, b, c, d - переменные, нужно рассмотреть возможность "разбиения"каждого из множителей c, d на две подгруппы множителей c_1, c_2 и d_1, d_2, с группировкой c_1, d_1 в a, а c_2, d_2 - в b. Здесь и возникают вспомогательные переменные c_1, d_1, c_2, d_2. Более подробное описание процедуры унификации, используемой в решателе, мы приведем в разделе, посвященном логическому выводу в базе теорем. Фактически, эта процедура представляет собой небольшую базу приемов и имеет до некоторой степени эвристический характер (так как некоторые результаты унификации могут оказаться чрезмерно громоздкими, а число их - весьма большим, приходится вводить правила обрыва при рассмотрении подслучаев). Реализуется она оператором "унификация(a b c)". Здесь a - набор установок на унификацию; b - набор переменных $x_1, \ldots, x_m$;

выходная переменная c перечисляет наборы термов $r_1, \ldots, r_m$. Мы приведем здесь лишь основные типы установок на унификацию, которые могут быть заданы при исходном обращении к процедуре (процедура имеет рекурсивный характер, и часть установок используется при внутренних обращениях):

а) Набор (унификация $p_i q_i K_i$) - указание пары унифицируемых термов p_i, q_i, сопровождаемых набором комментариев K_i. При исходном обращении к процедуре этот набор пуст и изменяется самой процедурой;

б) Набор (переменные X), где X - список всех переменных термов, вовлеченных в унификацию. Эта установка обычно создается самой процедурой и служит для того, чтобы новые параметры вводились не из списка X;

в) Набор (единица A) - при унификации допускается подстановка "единиц"вместо переменных списка A (используется тождество для единицы соответствующей операции, устраняющее ее из унифицируемого терма).

г) "приведение" - блокировка упрощений промежуточных унифицируемых термов, выполняемых специальной процедурой.

д) "новаяпеременная" - блокируется введение новых переменных.

Некоторые перечисляющие операторы связаны со специальными предметными областями. Так, в элементарной алгебре при формулировке решающих правил приемов, работающих с суммами дробных выражений, бывает полезен оператор "внешзнаменатель(a b)", перечисляющий по заданному вхождению a основания степени сомножителя знаменателя некоторой дроби всевозможные вхождения b знаменателей других дробей - слагаемых той же внешней суммы, что и указанная дробь, а также вхождение числителя данной дроби. Если при обращении в качестве a взять вхождение основания степени сомножителя числителя дроби, то b выдает (в режиме перечисления) вхождение знаменателя той же дроби. Другой оператор такого типа - "Слагаемое(a b c d)", у которого a - вхождение первого символа алгебраического выражения. Перечисляются слагаемые этого выражения, каждое из которых представляется в виде произведения коэффициента - простой числовой дроби - на некоторое выражение A. Выходная переменная b при этом выдает числитель коэффициента, выходная переменная c - знаменатель (оба - представленные не в виде термов, а в виде десятичных чисел), выходная переменная d - выражение A.

Оператор "слагаемое(a b)"получает в качестве значения a вхождение некоторого алгебраического выражения и перечисляет вхождения b всех его слагаемых с отброшенными знаками "минус".

Оператор "сомножитель(a b)"перечисляет вхождения b сомножителей выражения, расположенного по вхождению a.

Оператор "обобщслагаемое(a b)"перечисляет вхождения b, достижимые из вхождения a переходами к его подтермам через операции "минус", "плюс", "умножение" (сами перечисляемые подтермы не имеют своим заголовком указанные операции).

Оператор "обобщмножитель(a b)"перечисляет вхождения b, достижимые из вхождения a переходами к его подтермам через операции "минус", "умножение", "дробь", "степень"(только через основание степени), "модуль". Здесь и далее перечисляемые подтермы могут иметь своим заголовком указанные операции.

Оператор "алгебрвхождение(a b)"перечисляет вхождения b, достижимые из вхождения a переходами к его подтермам через операции "плюс","минус","умножение", "дробь","степень"(через основание степени), "модуль".

Оператор "числовойатом(a b c)"перечисляет все неконстантные атомарные числовые подвыражения c терма a (все надвыражения некоторого вхождения c имеют числовое значение, а некоторый операнд c - нечисловой). b - список утверждений, в

контексте которого распознаются числовые значения.

Обычно реализованные на ЛОСе операторы, связанные с конкретной предметной областью, используются компилятором ГЕНОЛОГа в качестве стандартных блоков; при программировании на ЛОСе вручную они практически не используются.

6.2.2 Вхождения в терм

Чтобы получить по заданному вхождению a подтерма $f(A_1 \dots A_n)$ набор вхождений подтермов $A_1, \dots, A_n$, используется операторное выражение "операнды(a)". Набор заголовков подтермов $A_1, \dots, A_n$ определяется выражением "наборсимволов(a)". Число n операндов A_i определяется операторным выражением "числооперандов(a)"(в формате десятичного числа) либо "Числооперандов(a)"(в формате символьного числа). Чтобы получить номер i операнда A_i по его вхождению v и вхождению подтерма a, служит операторное выражение "номероперанда(a v)"Этот номер выдается в формате символьного числа.

Если нужно проверить, что подтерм не находится в области действия квантора либо описателя по его свободной (без учета внешнего контекста) переменной, то применяется оператор "свобоперанд(a)"; a - вхождение первого символа этого подтерма. Для получения вхождения первого операнда A квантора либо описателя $F(x_1 \dots x_n A \dots)$, следующего за кванторной приставкой $x_1 \dots x_n$, используется оператор "конецприставки(a b)". У него a - вхождение первого символа данного квантора либо описателя; выходной переменной b присваивается вхождение A.

Для сохранения ссылки на вхождение в терм T (например, при записи его в файле) используется специальный код вхождения, называемый указателем вхождения. Этот код представляет собой терм "фикс($i_0 i_1 \dots i_p$)", у которого $i_0, i_1, \dots, i_p$ - символьные номера, определяющие выбор операндов при перемещении внутри T от заголовка к требуемому вхождению. Если T имеет вид "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)", то i_0 равно номеру того A_j, внутри которого находится вхождение ($i_0 = 0, 1, \dots, m$). Далее очередное i_k ($k = 1, \dots, p$) является просто номером того операнда (нумерация начинается с 1), внутрь которого следует переходить. Если же T не имело вида квантора общности, то указанное правило выполняется сразу, начиная с i_0. Операторное выражение "указательвхождения(a b)"позволяет получать указатель вхождения b в терм a, а операторное выражение "вхождениеуказателя(a b) находить вхождение в терм a по указателю b.

Чтобы проверять, что вхождение a располагается внутри подтерма с заголовком b, служит оператор "внешзнак(a b)".

Если кванторная импликация "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)"идентифицируется начиная не с корня, а с некоторого внутреннего вхождения, то может оказаться необходимым оператор, осуществляющий переход от вхождения некоторого A_i к вхождению корня этой импликации. При таком переходе одновременно учитывается возможность перестановки антецедента с консеквентом (которые заменяются на свои отрицания), а также возможность преобразования квантора существования в отрицание квантора общности. Этот оператор, называемый "внешнийквантор(a b c)", получает вхождение a некоторого подутверждения, а также указатель b на то, ожидается ли, что a окажется вхождением антецедента ($b = 0$) либо консеквента ($b = 1$). Если, с точностью до указанных выше преобразований, a действительно оказывается таким вхождением, то выходной переменной c присваивается вхождение внешнего квантора общности либо существования. Ослабленная версия указанного оператора относится только к случаю квантора общности (без преобразования квантора суще-

ствования к виду отрицания квантора общности) и имеет вид "внешдлялюбого(a b c)".

Можно получить вхождение антецедента A_i кванторной импликации "длялюбого ($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)"по его номеру i, используя операторное выражение "антецедентномер(a b)". Здесь a - вхождение корня указанной импликации либо она сама; b - символьный номер $i, i = 1, \dots, m$.

Операторное выражение "циклоперанд(a)"позволяет находить вхождение операнда A_j операции $F(A_1 \dots A_n)$, циклически следующего за вхождением a операнда A_i: при $i < n$ здесь $j = i+1$, а при $i = n$ - $j = 1$. Операторное выражение "Циклоперанд(a b)"аналогично предыдущему; при $b = 0$ его значение совпадает со значением предыдущего, а при $b = 1$ - берется не циклически следующее, а циклически предшествующее вхождение.

Для логических символов со специальной симметрией предусмотрена характеризация допустимых перестановок операндов с помощью справочника "схемаоперандов". Если текущий логический символ f некоммутативен, но допускает некоторые перестановки операндов, то указанный справочник возвращает древовидную структуру данных, неконцевые вершины которой соответствуют подгруппам операндов операции f и являются наборами вида $(AB_1 \dots B_n)$. Здесь A - указатель возможности перестановки подгрупп операндов, определяемых ссылками $B_1, \dots, B_n$ ($A = 0$ - перестановка недопустима, $A = 1$ - допустима). Каждая ссылка B_i - либо набор аналогичного указанному выше вида для подгруппы операндов, либо номер единственного операнда (символьное число). Для удобства использования данного справочника введено операторное выражение "схемаоперандов(a b c)". У него a - вхождение операции f со специальной симметрией; b - результат обращения к справочнику "схемаоперандов"для f; c - 0 либо набор вхождений операндов операции a. Значением выражения является результат замены в древовидной структуре данных b номеров операндов операции a на их вхождения. Значение переменной c можно брать равным 0; набор вхождений операндов операции a создается в этом случае программой самостоятельно и используется далее для рекурсивных обращений.

Иногда бывает нужно проверить наличие внешней операции заданного типа и перейти к ее рассмотрению, а если ее нет - сохранить ссылку на текущее вхождение. Например, при работе с алгебраическими выражениями часто приходится переходить от вхождения подвыражения к вхождению его знака "минус", если такой знак имеется. В этом случае полезным оказывается операторное выражение "внешсимвол(a b)". У него b - текущее вхождение в терм; a - логический символ. Если внешняя операция для b есть a, то значением выражения служит ее вхождение, иначе - сохраняется значение b.

6.2.3 Построение нового терма

Для получения утверждения, являющегося отрицанием данного, служит операторное выражение "отрицание(t_1)". Если значением t_1 служит утверждение F, заголовок которого отличен от "не", то значением указанного выражения является утверждение "не(F)". Если же значением t_1 служит утверждение вида "не(F)", то выражение имеет значение F. Аналогичное операторное выражение "заменазнака(t_1)"позволяет изменять знак у алгебраического выражения, являющегося значением t_1 (при отсутствии внешней операции "минус"вводит ее; при наличии - отбрасывает). Еще одно операторное выражение такого типа - "измзнака($t_1 t_2 t_3$)". Если значением t_3 является логический символ "0", то значением выражения является терм - значение t_1. Если

же t_3 есть "1", то значением выражения является терм, получаемый из t_1 вводом внешней операции t_2 при ее отсутствии и отбрасыванием - при ее наличии.

Если требуется построить терм с помощью заданной операции из заданного набора операндов, причем в случае одноэлементного набора эта операция вырождается, то используется операторное выражение "унисборка($t_1\, t_2$)". Если значением t_1 здесь является логический символ f, а значением t_2 - набор термов $(A_1, \ldots, A_n)$, то значением выражения является терм, совпадающий с A_1 в случае $n = 1$ и имеющий вид $f(A_1 \ldots A_n)$ при $n > 1$.

Если требуется построить терм, получаемый применением к двум заданным термам некоторой ассоциативной операции, причем операнды сами могут иметь эту операцию своим заголовком, то для устранения ее вложенных вхождений используется операторное выражение "соединение($t_1\, t_2\, t_3$)". Здесь t_1 - логический символ, обозначающий операцию f; t_2,t_3 - термы, к которым ее требуется применить. Каждый из этих термов, имеющий своим заголовком указанную операцию, сначала заменяется на набор ее операндов, а затем по объединенному (с сохранением порядка) набору термов $(A_1, \ldots, A_n)$ строится терм $f(A_1 \ldots A_n)$ - значение рассматриваемого выражения. Для устранения вложенных вхождений операций часто применяется также операторное выражение "спускоперандов($t_1\, t_2$)". Здесь t_1 - вхождение первого символа терма $f(A_1 \ldots A_n)$; t_2 - вхождение первого символа операнда A_i, имеющего вид $g(B_1 \ldots B_k)$. Значением выражения служит терм $f(A_1 \ldots A_{i-1}\, B_1 \ldots B_k\, A_{i+1} \ldots A_n)$. Преобразования по типу дистрибутивности выполняются при помощи операторного выражения "дистрибразвертка($t_1\, t_2\, t_3$)". Здесь t_1 - логический символ f; t_2 - вхождение первого символа терма $g(A_1 \ldots A_n)$; t_3- вхождение первого символа операнда A_i, имеющего вид $h(B_1 \ldots B_m)$. Значением выражения является терм $f(g(A_1 \ldots A_{i-1}\, B_1\, A_{i+1} \ldots A_n) \ldots g(A_1 \ldots A_{i-1}\, B_m\, A_{i+1} \ldots A_n))$.

Для замены операнда некоторой операции используется выражение "заменаоперанда ($t_1\, t_2\, t_3$)". Если t_1 - вхождение первого символа терма $f(A_1 \ldots A_n)$; t_2 - вхождение первого символа операнда A_i; t_3 - терм B, то значением данного выражения служит терм $f(A_1 \ldots A_{i-1}\, B\, A_{i+1} \ldots A_n)$. В случае замены двух операндов на один новый терм применяется выражение "склейкаоперандов($t_1\, t_2\, t_3\, t_4$)". Здесь t_1 - вхождение первого символа терма $f(A_1 \ldots A_n)$; t_2, t_3 - вхождения первых символов операндов A_i, A_j; $i \neq j$; t_4 - терм B. Значением выражения служит: при $n = 2$ - терм B, а при $n > 2$ - терм $f(A_1 \ldots A_{i-1}\, A_{i+1} \ldots A_{j-1} A_{j+1} \ldots A_n\, B)$. Выражение "внутрзамена($t_1\, t_2\, t_3$)"аналогично выражению "заменаоперанда(...)".Если t_1 - вхождение первого символа некоторого подтерма A; t_2 - вхождение первого символа подтерма B подтерма A и t_3 - терм C, то значением указанного выражения является терм, полученный заменой вхождения подтерма B в подтерм A на терм C.

Ранее было приведено базисное операторное выражение "заменатермов(...)", позволяющее заменять набор вхождений подтермов в заданный терм на новые термы. Его существенной особенностью являлось то, что заменяемые вхождения должны были располагаться в терме строго по порядку - слева направо. Операторное выражение "замещение($t_1\, t_2\, t_3$)"выполняет то же самое преобразование, но без указанного ограничения. У него t_1 - терм; t_2 - набор заменяемых вхождений в этот терм; t_3 - набор заменяющих термов, имеющий ту же длину, что и t_2. В отличие от операторного выражения "заменатермов(...)", у которого значением t_2 мог быть не набор вхождений, а единственное заменяемое вхождение, здесь значение t_2 - обязательно набор. Заметим, что хотя выражение "замещение(...)"свободно от ограничения на упорядоченность заменяемых вхождений, оно реализуется существенно более трудоемким образом, чем выражение "заменатермов(...)", так что применение его целесообразно

лишь в случаях крайней необходимости.

Для переобозначения переменных связывающей приставки служит операторное выражение "переобозначениесвязок(a b)". У него a - терм вид $F(x_1 \dots x_n t_1 \dots t_m)$; $x_1, \dots, x_n$ - переменные, а t_1 - не переменная. b - некоторый набор переменных. Значением этого операторного выражения является результат переобозначения в a переменных $x_1, \dots, x_n$ на попарно различные переменные, не входящие в набор b и не имеющие вхождений в a. Усиленный вариант переобозначения связанных переменных дает выражение "новыесвязки(a b)". Здесь a - терм (произвольного вида); b - набор термов. Значением выражения является результат переобозначения в a связанных переменных на переменные, не встречающиеся в термах из b.

Чтобы получить отрицание квантора общности либо существования a (знак квантора заменяется на противоположный и соответственно корректируются подкванторные утверждения), применяется операторное выражение "отрицаниеквантора(a)". Для преобразования утверждения a к виду "или($A_1 \dots A_n$)", где каждое A_i имеет вид "и($B_{i1} \dots B_{im_i}$)" ;$1 \leq m_i$; $1 \leq n$, применяется операторное выражение "днф(a)". Это преобразование использует только дистрибутивность, коммутативность и ассоциативность логических связок "и","или".

Если нужно осуществить одновременную замену в терме a согласно некоторому списку b пар термов (заменяемый образец - заменяющий образец), то может быть использовано операторное выражение "мультизамена(a b)". Предпринимается выделение в a непересекающихся друг с другом вхождений первых элементов пар из b и замена их на вторые элементы этих пар. Предпочтения при выделении вхождений - согласно порядку следования пар в b.

Для лексикографического упорядочения операндов коммутативных операций и отношений в терме a, а также операций и отношений со специальной симметрией (например, квантора общности с его антецедентами) служат несколько отличающихся друг от друга во второстепенных деталях операторных выражений: "стандупорядочение(a)", "станд(a)", "стандтерм(a)". В приемах решателя обычно используется первое из них; в логическом выводе теорем - второе и третье. Для перехода к стандартной перегруппировке операндов при прорисовке формулы на экране используется еще одно выражение аналогичного типа - "нормформ(a)".

Операторное выражение "метатерм(t_1)"имеет своим значением операторное константное выражение, имеющее своим значением терм - значение операторного выражения t_1. Это выражение применяется при автоматическом формировании программ на ЛОСе.

6.2.4 Свойства терма и отношения между термами

Оператор "элементарно(t_1)"истинен, если t_1 - терм, не содержащий кванторов, описателей "класс", "отображение", а также логических связок "и", "или", "эквивалентно"(допускаются вхождения этих символов, после которых не идет открывающая скобка). Оператор "простаяимпликация(t_1)"истинен, если значением t_1 служит утверждение вида "длялюбого($x_1 \dots x_k$ если $A_1 \dots A_n$ то A_0)", где $k > 0, n \geq 0$, причем все утверждения $A_0, A_1, \dots, A_n$ - элементарны в смысле оператора "элементарно(...)". Оператор "константа(t_1)"истинен, если терм t_1 не имеет переменных.

Оператор "бесповторно(a)"проверяет отсутствие в терме a повторных вхождений одной и той же переменной. Операторное выражение "числоповторений(a)"дает число повторных вхождений переменных в терм a (равное разности числа вхождений переменных в этот терм и числом переменных терма).

Операторное выражение "длинанабора(a)"выдает число символов в записи терма a либо длину набора a (десятичное число). Выражение "глубина(a)"имеет своим значением глубину терма a (десятичное число); глубина переменной либо константы здесь полагается равной 0. Выражение "глуб(p b)"дает наибольшую глубину вхождений переменной p в терм a. При определении глубины не учитываются одноместные операции, которые могут быть перегруппированы внутрь многоместной операции таким образом, чтобы они не лежали на путях от вхождений переменной p к корню терма (например, "минус"перед произведением может быть отнесен к сомножителю без p). Выражение "числочленов(a f)"равно: если a имеет вид $f(A_1 \dots A_n)$, то числу (десятичному) n, иначе оно равно 1. Выражение "числоконстант(a)"имеет своим значением число вхождений в терм a логических символов, после которых не идет открывающая скобка.

Операторное выражение "симметричныепары(a)"определяет по терму a набор лексикографически упорядоченных пар его переменных, перестановка которых не изменяет терма (с точностью до изменения порядка операндов коммутативных операций и отношений).

Ответы задач на описание часто предствляют собой дизъюнктивно-конъюнктивные логические конструкции a, в которых значения неизвестных определяются с разбором подслучаев. Чтобы определить, что во всех таких подслучаях значение некоторой переменной x определяется через значения переменных списка p при помощи явного равенства, служит оператор "опредзначение(a x p)".

Список связанных переменных квантора либо описателя A опре деляется с помощью операторного выражения "связприставка(A)". Здесь A - вхождение первого символа этого квантора либо описателя.

Оператор "внутрзаголовок(a b c)"получает в качестве входного данного вхождение a первого символа некоторого терма. Этот терм представляется в виде $f_1(f_2(\dots f_n(t)\dots))$, где $f_1, \dots, f_n$ - одноместные операции; t - терм, заголовком которого служит более чем одноместная операция (при отсутствии такого предствления оператор ложен). Значением выходной переменной b становится список операций $f_1, \dots, f_n$ (без повторений), значением выходной переменной c - заголовок терма t.

Оператор "подобныетермы(a b)"истинен, если термы a и b получаются друг из друга переобозначением без отождествлений переменных и перестановкой операндов коммутативных операций и отношений. Оператор "подобныеимпликации(a b)"проверяет, что кванторные импликации a и b, не имеющие внутри своих антецедентов и консеквента связывающих символов, могут быть получены друг из друга переобозначением без отождествлений связанных переменных, а также изменением порядка операндов в коммутативных операциях и операциях со специальной симметрией. Если требуется не только установить подобие импликаций указанного вида, но и найти соответствующее переобозначение связанных переменных, используется оператор "Подобныеимпликации(a b c d)". Здесь a,b - две кванторных импликации без свободных переменных; c - набор переменных первой из них. Если b может быть получено из a переобозначением без отождествлений переменных списка c, а также изменением порядка операндов в коммутативных операциях, то переменной d присваивается (без перечисления) набор однобуквенных термов, определяющих новые переменные для c.

Чтобы определить подстановку, переводящую один терм в другой, с точностью до изменения порядка операндов в коммутативных операциях, используется оператор "усмподст(a b c d)". У него a,b - термы; c - набор переменных, вместо которых делается подстановка в терм a. Выходной переменной d присваивается набор термов,

подставляемых вместо переменных набора c. Так как результат определен, вообще говоря, неоднозначно, оператор работает в режиме перечисления.

Оператор "вычерк(a b)"проверяет, что терм b получен из терма a вычеркиванием части операндов некоторых операций.

Оператор "сравнтермов(a b c d)"сравнивает два терма a,b начиная с их корней и определяет в каждом из них такой наименьший подтерм, что a,b отличаются только внутри этого подтерма. Выходным переменным c,d присваиваются вхождения корней этих подтермов (в соответствующем порядке).

Ряд операторов был реализован на ЛОСе с целью упростить описание вида стандартных теорем, по которым происходит синтез приемов того или иного типа.

Оператор "квазипеременная(a b)"проверяет, что по вхождению a располагается символ переменной b либо одноместная операция от этой переменной; если значение b на момент обращения не определено, то ему присваивается указанная переменная. Оператор "унитерм(a b)"проверяет, что терм a имеет единственное вхождение переменной; эта переменная присваивается выходному значению b.

Оператор "бинарнаяоперация(a b c d)"проверяет, что по вхождению a расположена бинарная операция. Если это так, то переменной b присваивается символ данной операции, а переменные c,d перечисляют вхождения различных операндов этой операции: сначала в прямом порядке, а затем в обратном. Оператор "Бинарнаяоперация (a b c d e f)"проверяет, что по вхождению a располагается бинарная операция, быть может, внутри одноместной операции. Переменной b присваивается символ бинарной операции, переменной c - символ одноместной операции (если ее нет, то 0), переменной d - указатель перестановки операндов двуместной операции : 0 - без перестановки, 1 - с перестановкой. Переменные e,f перечисляют вхождения самих операндов (сначала без перестановки, затем с перестановкой). Оператор "двойнаяоперация(a b c d)"проверяет, что по вхождению a расположена одноместная операция, примененная к двуместной. Переменной b присваивается символ двуместной операции, а c,d перечисляют вхождения ее операндов: сначала - без перестановки, затем с перестановкой.

Оператор "унарноеотношение(a b)"проверяет, что утверждение, вхождением первого символа которого служит a, имеет вид $P(A)$ либо $P(Ac)$, где c - логический символ. Переменной b присваивается вхождение первого символа выражения A. Оператор "простоеусловие(a b)"проверяет, что утверждение a имеет вид $P(y)$, либо "не ($P(y)$)", либо $Q(ya)$, либо "не($Q(ya)$)", где a - логический символ, y - переменная (быть может, под знаком одноместной операции). В этих случаях b присваивается переменная y.

Оператор "простоеотношение(a b c)"проверяет, что утверждение a имеет один из видов: $P(AB)$, "не($P(AB)$)", $P(f(AB)d)$, "не($P(f(AB)d)$)", где d - логический символ; A, B - различные переменные x, y, быть может, находящиеся под знаком одноместных операций. b, c присваиваются значения x, y. Оператор "Простоеотношение(a b c)"аналогичен предыдущему, но A, B могут быть только переменными. Оператор "бинарноеотношение(a b c)"проверяет, что утверждение, вхождение первого символа которого есть a, с точностью до отбрасывания внешнего отрицания, имеет вид $P(AB)$ либо $P(f(AB)d)$; d - логический символ. Переменным b,c присваиваются вхождения первых символов выражений A, B - сначала без перестановки, затем с перестановкой.

6.2.5 Списки термов

Для лексикографического упорядочения термов набора a используется оператор "лексупорядочение(a b)". Если набор a уже был лексикографически упорядочен, то выходной переменной b присваивается логический символ 1, иначе ей присваивается результат лексикографического упорядочения данного набора. Оператор "циклупорядочение(a b)"находит такую циклическую перестановку b термов набора a, быть может, сопровождаемую изменением порядка термов на противоположный, которая является наименьшей в лексикографическом порядке.

Операторное выражение "компонента(a b c)"используется для нахождения наибольшего подмножества множества термов b, зависящих от переменных пересечения списков a,c либо достижимых из них путем переходов от одного терма списка b к другому, имеющему с ним общую переменную из c. Оно обычно используется при разбиении списка условий задачи на описание на независимые компоненты либо при разбиении подкванторных утверждений на независимые подгруппы.

Оператор "симметрично(a b c)"проверяет, что термы a,b входят в термы набора c симметричным образом (с точностью до допустимых перестановок операндов).

Оператор "подборзначений(a b c d e)"позволяет находить такой набор термов d, подстановка которого вместо переменных списка c в термы набора a переводит последние в некоторые термы набора b (с точностью до допустимых перестановок операндов). Выходной переменной e присваивается набор тех термов списка b, в которые были переведены термы списка a. Оператор работает в режиме перечисления. При обращении к нему требуется, чтобы переменные списка c не входили в термы списка b. Оператор "подборнеизвестных(a b c d e)"несколько усиливает предыдущий оператор. У него a, b - списки утверждений (рассматриваемые, соответственно, как посылки и условия); c - список переменных ("неизвестных"), не встречающихся в посылках. Оператор перечисляет всевозможные наборы d значений неизвестных, при которых все условия оказываются истинными, сопровождая каждый такой набор списком e использованных посылок. Отличие от оператора "подборзначений"заключается в том, что непосредственно в списке посылок реализуется лишь часть условий, а остальные условия, по мере того, как оказываются определены значения входящих в них неизвестных, проверяются путем решения вспомогательных задач на доказательство. Оператор "Реализация(a b c d)"совпадает с оператором "подборзначений (a b c d)", но не требует невхождения переменных списка c в термы набора b.

Операторное выражение "выражения(a)"выдает набор максимальных по включению выражений, содержащихся в терме a. Операторное выражение "утверждения (a)"выдает набор атомарных утверждений, из которых утверждение a построено при помощи логических связок "не", "и", "или", "эквивалентно".

Если терм b имеет своим заголовком логический символ a, то значением выражения "наборчленов(a b)"является набор корневых операндов терма b, иначе этим значением является одноэлементный набор, состоящий из b. Набор антецедентов $A_1, \ldots, A_n$ кванторной импликации "длялюбого($x_1 \ldots x_m$ если $A_1 \ldots A_n$ то A_0"дается операторным выражением "наборантецедентов(a)". Здесь a - сама кванторная импликация либо вхождение ее первого символа.

6.3 Операторы для работы с задачами

6.3.1 Элементы задачи и связи ее с другими задачами

Если нужно преобразовать терм, встречающийся в задаче, то для обоснования корректности этого действий приходится создавать список A всех утверждений, относящихся к области рассматриваемого вхождения, и решать вспомогательные задачи на проверку условий корректности относительно A как списка посылок. Для получения A используется операторное выражение "облвхожд(a b c d)", у которого (a, b, c) - координата вхождения в задачу d. Если замена затрагивает не один, а несколько термов задачи (например, несколько условий заменяются на одно новое условие), то при обосновании корректности все эти термы должны исключаться из списков посылок вспомогательных задач. Здесь бывает полезно операторное выражение "Облвхожд(a b c d e)", отличающееся от предыдущего тем, что из результирующего списка исключаются все утверждения, указанные в списке e. Еще один вариант выражения такого типа - "конъюнктконтекст(a b c d e)". У него (a, b, c) - координата вхождения в задачу d конъюнкции либо кванторной импликации; e - набор вхождений в конъюнктивные члены либо в антецеденты и консеквент терма T по вхождению a. Значением выражения служит область вхождения (a, b, c), пополненная не содержащими вхождений из e конъюнктивными членами либо антецедентами терма T. Это выражение используется при проверке корректности одновременной замены группы утверждений из конъюнкции либо кванторной импликации. Если рассматриваемое вхождение в терм задачи - корневое либо не расположено внутри таких логических конструкций, которые могут добавить что-либо к его области, то можно применять упрощенную версию операторного выражения "областьвхождения" - выражение "подобл(a b c)". У него a - вхождение в список условий задачи c при $b = 1$ и в список посылок при $b = 0$ некоторого терма. Значением выражения является: при $b = 0$ - набор всех отличных от a посылок задачи, а при $b = 1$ - набор всех посылок задачи и всех ее условий, отличных от a.

Для проверки того, что терм a не содержит неизвестных задачи b, используется оператор "известно(a b)". Строго говоря, его следует применять только для задач на описание и исследование, либо для редко втречающихся задач на преобразование, снабженных целью "неизвестные $x_1 \ldots$". Однако, предусмотрено его использование и в случае задач на доказательство - терм считается "известным"в контексте такой задачи, если он либо не содержит переменных, либо задача на доказательство имеет комментарий "известно $a_1 \ldots a_n$", среди переменных $a_1, \ldots, a_n$ которого содержатся все переменные данного терма. Оператор "известны(a b)"проверяет, что ни один из термов набора a не содержит неизвестных задачи b.

Оператор "независит(a b)"проверяет, что терм a не имеет свободных переменных, встречающихся в цели "независит $x_1 \ldots x_n$"задачи на описание b. Напомним, что такая цель вводится в тех случаях, когда ответ не должен зависеть от ряда параметров, встречающихся в списке посылок. Оператор "параметр(a b)"проверяет, что переменная a встречается в цели "параметры $x_1 \ldots x_n$"задачи на описание b (т.е. a - несущественная неизвестная задачи, для которой достаточно установить существование значения, не указывая его в ответе явным образом).

Оператор "родобъекта(a b c Z v t K)"позволяет находить список t типов значения выражения по вхождению v, усматриваемых в контексте вхождения в задачу Z, имеющего координату (a,b,c). Выходной переменной K при этом присваивается список утверждений данного контекста, использованных при определении типа.

Оператор "контрольвывода(a b)"проверяет, что каждое атомарное выражение, встречающееся в утверждении b, уже имеется в посылках либо условиях задачи a. Этот оператор бывает полезен для решающих правил приемов, оценивающих целесообразность добавления нового утверждения b к посылкам либо условиям задачи.

Оператор "зона(a b c Z T d e)"получает координату (a, b, c) вхождения в задачу Z. Если это вхождение - в условие U, то он перечисляет все отличные от U условия T задачи, а если в посылку U - то также и все отличные от U посылки T. При этом d,e - второй и третий разряды координаты вхождения первого символа утверждения T в задачу. Если U - посылка, то перечисление начинается со списка посылок. Данный оператор фактически перечисляет все термы задачи, к области вхождений которых относится тот терм, в котором выделено вхождение (a, b, c).

Если в посылках задачи имеется некоторое множество утверждений $F(AB)$ для транзитивного, рефлексивного и симметричного отношения F, то может понадобиться перечисление с помощью этих утверждений всех термов, F - эквивалентных заданному терму T. В этом случае используется оператор "транзитоперанд(a b c)". У него a - вхождение операнда A_i утверждения вида $F(A_1A_2)$; b - список посылок. c перечисляет следующие вхождения: а) вхождение операнда, противоположного операнду A_i; б) если $i = 1$, то - вхождения всех операндов A_3 утверждений $F(A_3A_2)$ из списка b. Здесь используется тот факт, что перед перечислением решатель стандартизирует группу относящихся к одному и тому же "классу эквивалентности"утверждений вида $F(AB)$ так, что все они имеют одинаковые вторые операнды B. Оператор используется в идентифицирующей части приемов, где при $i = 2$ бывает достаточно ограничиться лишь указанием на противоположный операнд A_1.

Оператор "внешусловие(a b)"перечисляет все условия b той задачи на описание, блоком анализа которой служит задача на исследование a. Операторное выражение "внешописать(a)"выдает ту задачу на описание, блоком анализа которой является задача на исследование a. Операторное выражение "внешняязадача(a)"выдает задачу, предшествующую в цепи задач текущей задаче. Его входная переменная a - фиктивная, ее значение несущественно.

Оператор "вхождениевзадачу(a b c d)"проверяет, является ли терм b условием либо посылкой задачи a. Если является, то переменной c присваивается вхождение этого терма в соответствующий список задачи; переменной d - 0 случае посылки и 1 в случае условия.

Связанные переменные встречающихся в задаче кванторов и описателей переобозначаются специальными приемами так, чтобы они не имели свободных вхождений во внешних по отношению к данному квантору либо описателю термах. Нарушение этого требования может повлечь логические ошибки. Для данного переобозначения используется оператор "нормализациясвязок(a b c d e)", у которого (a, b, c) - координата вхождения первого символа квантора либо описателя $F(x_1 \ldots x_k a_1 \ldots a_n)$ в задачу d. Если использование связанных переменных $x_1 \ldots x_n$ в данной ситуации допустимо, то выходной переменной e присваивается 1, иначе ей присваивается результат необходимого переобозначения связанных переменных в рассматриваемом кванторе либо описателе.

6.3.2 Преобразования задачи

Преобразования условий и посылок задачи выполняются приемами при помощи небольшого числа стандартных операторов. Это позволяет вводить дополнительный общий контроль за ходом решения задачи, локализуя его в указанных нескольких

операторах. Операторы преобразования задачи, кроме собственно изменения условий и посылок, выполняют большое число сопутствующих действий. Они корректируют сопровождение термов задачи условиями на область допустимых значений при их преобразованиях; учитывают общие установки на блокировку срабатываний приемов; корректируют комментарии задачи; изменяют веса условий и посылок; обнаруживают зацикливание, и т.д. Подробное описание устройства этих операторов будет приведено впоследствии; здесь мы ограничимся их перечислением и общей характеризацией.

Наиболее часто используемый и наиболее сложный оператор преобразования задачи - "заменавхождения($t_1t_2t_3t_4t_5t_6$)", используемый для тождественных и эквивалентных замен. У него (t_1, t_2, t_3) - координата вхождения в задачу t_4 заменяемого подтерма; t_5 - заменяющий терм либо логический символ. t_6 - набор информационных элементов, определяющих сопутствующие действия. Перечислим основные типы этих элементов. Прежде всего, к их числу относится ссылка на примененный прием. Она вводится только в случае приемов, реализованных на ГЕНОЛОГе, и представляет собой набор (прием a b c), формат которого будет описан в разделах, посвященных ГЕНОЛОГу. Наличие такой ссылки позволяет реагировать заданным образом на срабатывание приема (например, регистрировать статистику срабатываний по приемам; блокировать применение избранных приемов, и т.п.). Информационный элемент (выводимо A) используется для учета списка A утверждений, использованных для обоснования корректности замены. Таких элементов в наборе t_6 может быть несколько. Список A может содержать нулевые разряды; если само A есть 0, то замена блокируется. Информационный элемент (замечание A) указывает, что при выполнении замены вводится комментарий (для задачи на исследование - комментарий посылок) A. Элемент (комментариипосылок A) указывает, что при выполнении замены вводится комментарий посылок A. Элемент (примечание A) перечисляет в наборе A все комментарии, которыми сопровождается изменяемый терм задачи. Ввод новых комментариев можно было бы выполнять и до обращения к оператору "заменавхождения(...)". Однако, этот оператор может отменить срабатывание приема, и тогда произошло бы рассогласование между комментариями, указывающими (прямо либо косвенно), что срабатывание произошло, и фактическим состоянием задачи. Элементы (удалениеусловия A) и (удалениепосылки A) указывают на удаляемое после выполнения замены условие (посылку) A, если только она не используется как утверждение, указывающее область допустимых значений для некоторого терма задачи. Элемент (посылка AB) указывает на новую посылку A, заносимую в список посылок при условии, что все утверждения, использованные при обосновании корректности замены, являлись посылками; B - список комментариев, которыми сопровождается эта посылка.

Иногда применяется групповая замена - сразу все вхождения в задачу некоторого терма заменяются на новый терм. Это позволяет ускорять действия решателя в тех случаях, когда правдоподобным является многократное появление в задаче одного и того же терма. Разумеется, выполнение замены в каждом конкретном случае требует дополнительной проверки того, что все использованные при обосновании корректности замены утверждения попадают в область заменяемого вхождения. Для групповой замены служит оператор "заменавхождений(a b c d)". У него a - задача; b - вхождение первого символа заменяемого терма T; c - заменяющий терм; d - набор информационных элементов, определяющих сопутствующие действия. Типы этих элементов - те же, что у оператора "заменавхождения(...)"(их подробный список можно найти в справочной информации для логического символа "заменавхождения"). Оператор

находит все вхождения в задачу терма T, замена которых на c является допустимой согласно элементам (выводимо ...) из d, и осуществляет для них одновременную замену. Если не было произведено ни одной замены, то оператор ложен.

Для вывода следствий в списке посылок служит оператор "вывод(a b c)". У него a - задача; b - утверждение, присоединяемое к ее списку посылок; c - набор информационных элементов, определяющих сопутствующие действия (типы их - те же, что и выше). При регистрации новой посылки выполняется лексикографическое упорядочение операндов для коммутативных операций (либо операций со специальной симметрией). Если результат уже содержался в списке посылок, либо являлся константой "истина", либо представлял собой кванторную импликацию, полученную из уже имеющейся в посылках переобозначением без отождествлений связанных переменных, то преобразование блокируется. При блокировке преобразования оператор оказывается ложным, если c не содержит логического символа "выход", и истинным в противном случае. В случае вывода следствий для условий задачи на описание применяется оператор "выводусловия(a b c)", аналогичный вышеуказанному.

Если нужно заменить сразу несколько посылок либо несколько условий на одно новое утверждение, применяется оператор "заменагруппы(a b c d)". Здесь a - набор посылок при $b = 0$ или набор условий при $b = 1$; c - задача; d - заменяющее утверждение. e - набор информационных элементов, определяющих сопутствующие действия (типы их те же, что и выше).

Удаление условий либо посылок задачи практически никогда не выполняется как основное преобразование приема; обычно оно сопровождает другие действия. Тем не менее, для этих целей также возникли специальные операторы. В первую очередь, это оператор "удалениепосылок(a b)", позволяющий исключать все посылки задачи a, упоминаемые в наборе b. При удалении корректируются некоторые структуры данных. Оператор "исключениеусловия(a b c)"позволяет удалить условие задачи a на описание либо доказательство, вхождение которого есть b. c - набор информационных элементов для сопутствующих действий. Если удаляемое условие было единственным, оно заменяется на константу "истина".

Этим, по существу, и исчерпываются основные типовые блоки преобразования задач, используемые в приемах. Приведем далее ряд операторов, используемых для вспомогательных преобразований задачи.

Оператор "учетнеизвестных(a b)"позволяет пополнить список неизвестных задачи на описание либо исследование a переменными, входящими в список b.

Оператор "одз(a)"используется в самом начале решения задачи a. Он выполняет расширение ее списков условий и посылок, добавляя к ним необходимые для указания на область допустимых значений встречающихся в задаче операций и отношений утверждения. Такие утверждения могут заноситься не только в виде новых посылок и условий, но и вставляться внутрь сложных логических конструкций (например, в качестве антецедентов кванторных импликаций).

Оператор "внимание(a b)"используется для переключения внимания, связанного с некоторым термом либо логическим символом b. Он находит все условия и посылки задачи a, содержащие b, и заменяет их веса на 0. Впрочем, этот оператор фактически используется редко - обычно указания на переключение внимания (путем определения специальных условий и посылок и уменьшения их веса до заданной величины), имеющиеся в описании приема на ГЕНОЛОГе, компилируются в кванторные операторы ЛОСа, непосредственно просматривающие списки посылок и условий.

Комментарии задачи часто имеют вид (sA), где логический символ s указывает тип комментария, A - накопитель объектов, пополняемый по ходу решения зада-

чи. Для удобства занесения в этот накопитель новых элементов введены операторы "Примечание(Z s t)", "Замечание(Z s t)", "Примечпосылки(Z v s t)". Каждый из них проверяет наличие комментария вида (sA): в первом случае - к посылкам задачи Z, во втором случае - к самой задаче, и в последнем случае - к посылке задачи, имеющей вхождение v. При наличии данного комментария в накопитель A заносится, если его там не было, элемент t; при отсутствии комментария - он создается, с накопителем, состоящим из t.

6.3.3 Создание и решение вспомогательных задач

Оператор "заменавхождения"осуществляет необратимое преобразование задачи. В тех случаях, когда есть опасения, что необратимая замена заведет процесс решения в тупик, используется версия замены с переходом к вспомогательной задаче - копии текущей задачи. Это осуществляет оператор "попытказамены(t_1 t_2 t_3 t_4 t_5)". У него ($t_1t_2$1) - координата вхождения в задачу t_3 заменяемого подтерма, t_4 - заменяющий терм, t_5 - набор информационных элементов, определяющих сопутствующие действия. Замена реализуется в копии текущей задачи; эта копия решается до получения результата. Если получен ответ A , отличный от символа "отказ", то он передается текущей задаче; для передачи используется комментарий (ответ A) к последней. При получении отказа оператор ложен, и в этом случае действия по решению текущей задачи продолжаются независимо от предпринятой попытки замены. Заметим, что при развитии автоматики принятия решений в предметной области обычно удается обойтись без использования оператора "попытказамены"; обращение к нему - исключительно редкая ситуация.

При решении задачи на доказательство либо на описание (если не нужно получить полного описания для всех допустимых значений неизвестных) может быть применен прием, заменяющий условие A на некоторое утверждение B, логическим следствием которого является A. Так как эта замена, вообще говоря, не является эквивалентной, то она не должна необратимым образом изменять текущей задачи (возможно, при откате придется к ней возвращаться). Поэтому для дальнейших действий вводится копия текущей задачи, в которой выполняется указанная замена и осуществлены необходимые коррекции комментариев. В случае задачи на описание, во избежании нежелательных склеек, здесь должен быть скопирован также и блок анализа. Перечисленные действия выполняются операторным выражением "спуск(Z v A)", у которого v - вхождение в список условий задачи Z (для задачи на доказательство - в саму эту задачу)заменяемого условия; A - заменяющее утверждение.

Ряд процедур, формирующих вспомогательные задачи и осуществляющих обращение к их решению, используется компилятором ГЕНОЛОГа в качестве стандартных блоков. Перечислим основные такие процедуры.

Указанная выше попытка решить задачу на доказательство либо на описание путем замены ее условия A на некоторое (вообще говоря, не эквивалентное A) утверждение B, следствием которого является A, реализуется оператором "попыткаспуска(Z v B d)". У него Z - задача на доказательство либо на описание; v - вхождение условия A в задачу (если Z - задача на доказательство) либо в список условий; d - набор информационных элементов, определяющих сопутствующие преобразования. Типы этих элементов - такие же, как у оператора "заменавхождения". Оператор создает, используя операторное выражение "спуск", вспомогательную задачу, и предпринимает попытку ее решить. Если это удается, то все условия текущей задачи заменяются на ответ вспомогательной задачи (в случае задачи на описание - на

группу конъюнктивных членов ответа). Иначе - оператор оказывается ложным.

Для решения вспомогательной задачи на преобразование служит оператор "вспомпреобразование(t_1 t_2 t_3 t_4 t_5)", у которого t_1 - преобразуемое выражение, t_2 - список посылок, относительно которых должно выполняться преобразование; t_3 - список целей вспомогательной задачи на преобразование. Выходной переменной t_4 присваивается ответ задачи на преобразование, выходной переменной t_5 - список использованных при решении посылок из t_2. Если в t_3 не входит логический символ "смпосылка", то веса посылок вспомогательной задачи полагаются равными 20 - сканирование посылок блокируется для ускорения вычислений. Ряд элементов списка t_3 в действительности не становятся целями вспомогательной задачи, а лишь используются для уточнения способа ее построения. Так, элементы (комментарий A) используются для указания комментариев A к вспомогательной задаче; элемент (уровеньобращения A) устанавливает максимальный уровень A для ее решения. Заметим, что при отсутствии такого элемента непосредственно до обращения к оператору должен быть применен оператор "уровеньобращения(A) иначе вспомогательная задача будет решаться лишь до текущего уровня текущей задачи. Результаты обращения к данному оператору сохраняются в специальном буфере (длина его равна 20), и при повторном идентичном обращении извлекаются непосредственно из буфера. Роль такого буфера играет комментарий (вспомпреобразование $A_1A_2A_3$) к посылкам исходной задачи. У него A_1 - набор троек ($t_1t_2t_3$) для входных данных обращений; A_2 - набор такой же длины, что и A_1, сохраняющий результаты обращений; A_3 - текущая (очередная изменяемая) позиция в наборе A_1. Эта текущая позиция перемещается циклически; в исходной ситуации наборы A_1, A_2 заполнены нулями.

Оператор "вспомпреобразование(...)"обычно используется внутри другой вспомогательной процедуры (как правило, внутри так называемого пакетного оператора, см. описание ГЕНОЛОГа). Для обращения к вспомогательной задаче на преобразование из приема, инициированного при сканировании задачи, служит оператор "преобразование(A t_1 t_2 t_3 t_4 t_5 Z)". У него A - базовая задача (обычно, текущая решаемая задача), из которой извлекается дополнительная информация, необходимая для формирования вспомогательной задачи на преобразование выражения t_1 относительно списка посылок t_2 со списком целей t_3. Выходные переменные t_4, t_5 - такие же, как у оператора "вспомпреобразование". Выходной переменной Z присваивается сама вспомогательная задача. В дополнение к решению задачи Z, оператор "преобразование" обеспечивает обратную передачу задаче A целого ряда комментариев задачи Z (их отбор и модификация обеспечиваются справочником "преобразование"). Отбираются комментарии, содержащие полезную для последующего информацию о новых термах, возникших в ответе вспомогательной задачи. При обращении к задаче Z может быть установлено ограничение на трудоемкость ее решения, по превышении которого оператор обрывается как ложный. Это ограничение задается элементом "лимит(N)"в наборе t_3; N - допустимая трудоемкость в числе шагов интерпретатора ЛОСа. Как и для оператора"вспомпреобразование", предусмотрен буфер. Однако, он используется и пополняется лишь при наличии в t_3 элемента "буфер". Технически буфер представляет собой комментарий (преобразование B) к посылкам исходной задачи, у которого B - набор троек (версия задачи Z до решения - версия задачи Z после ее решения - ответ на задачу Z). В буфере хранятся 5 последних результатов.

Для проверки истинности утверждения в заданном контексте имеется целый ряд вспомогательных операторов, отличающихся глубиной предпринимаемых ими попыток. Выбор конкретной версии оператора предпринимается из соображений целесообразности тех или иных затрат трудоемкости на проверку. Первый из таких опера-

торов - "извлекается(a b c)". У него a - проверяемое утверждение; b - список посылок, относительно которых выполняется проверка. Вспомогательная задача на доказательство решается здесь до максимального уровня 4, причем веса ее посылок изначально полагаются равными 20 - это означает отключение сканирования посылок. Оператор истинен, если удается усмотреть истинность a, причем в этом случае переменной c присваивается список фактически использованных посылок. Реже используется оператор "сильноизвлекается(a b c)", отличие которого от предыдущего состоит только в более высоком максимальном уровне вспомогательной задачи - 7 вместо 4. Оператор "выводимо(a b c)"отличается от оператора "извлекается"тем, что исходные веса его посылок равны 0, а максимальный уровень обращения N должен устанавливаться заранее с помощью оператора "уровеньобращения(N)"(иначе вспомогательная задача будет решаться до текущего уровня текущей задачи). Наиболее употребительным для вспомогательных проверок при отсутствии сильных ограничений на привлекаемые средства оказался оператор "следствие(a b c d)". У него a, b - проверяемое утверждение и список посылок; c - набор комментариев, передаваемых вспомогательной задаче. Если требуется указать также комментарий K к конкретной посылке P из b, то в c заносится элемент (комментарийпосылки P K) (сам он не включается в список комментариев формируемой вспомогательной задачи). Как и в остальных случаях, выходной переменной присваивается список использованных при проверке посылок. Максимальный уровень вспомогательной задачи у этого оператора, как и у предыдущего, должен устанавливаться заблаговременно с помощью оператора "уровеньобращения".

Заметим, что обращение к вспомогательной задаче на доказательство для проверки истинности утверждения обычно является значительно более трудоемким, чем использование специализированной проверочной процедуры для утверждений заданного вида. Такие процедуры, называемые проверочными операторами, легко реализуются на ГЕНОЛОГе, и стандартом при компиляции приема является обращение для проверки истинности условий именно к ним. Существует способ обратиться к пакетному оператору, не зная его заголовка - для этого служит оператор "легковидеть(a b c d)". Здесь a - проверяемое утверждение, b - список посылок, c - набор комментариев, передаваемый проверочному оператору. Выходной переменной d присваивается список использованных посылок. При отсутствии специализированного проверочного оператора для утверждения a оператор "легковидеть"оказывается ложным.

Если прием, реализуемый в контексте некоторой текущей задачи Z, должен обратиться к решению вспомогательной задачи на описание, то используется оператор "вспомописание(Z a b c x y z)". Здесь a - утверждение, представляющее собой конъюнкцию условий вспомогательной задачи на описание A; b - список утверждений, из которых извлекаются необходимые для сопровождения a дополнительные условия этой задачи; c - список целей. Посылками задачи A становятся все не зависящие от ее неизвестных утверждения набора b. Комментарии к вспомогательной задаче извлекаются (быть может, модифицированные с помощью специального справочника) из задачи Z. Максимальный уровень N задачи A устанавливается при помощи элемента (уровеньобращения N) из c. Выходной переменной x присваивается ответ задачи A; y - список использованных утверждений из b; переменной z присваивается сама задача A (версия ее, возникшая по окончании решения). Как и в случае задач на преобразование, можно сохранить результат решения в буфере для извлечения его оттуда в случае повторного обращения. Для подключения буфера в c должен быть занесен элемент "буфер". Роль буфера играет комментарий (вспомописание B) к посылкам исходной задачи; у него B - набор троек (исходная версия задачи -

результирующая ее версия - ответ). Длина этого набора не более 5.

Если нужно решить вспомогательную задачу на описание безотносительно к контексту какой-либо внешней задачи, можно использовать оператор "быстроописание(a b c x)". Здесь a - конъюнкция условий вспомогательной задачи; b - список посылок; c - список целей. Переменной x присваивается ответ задачи. Веса посылок здесь полагаются равными 20 (блокировка сканирования посылок); элементы (уровень-обращения N) и (комментарий K) из c определяют максимальный уровень задачи и ее комментарии K.

6.4 Арифметические операторы

6.4.1 Десятичные числа

Напомним, что под десятичным числом в ЛОСе понимается либо цифра (логический символ от 0 до 9), либо не начинающийся с 0 набор цифр, возможно, с запятой (логический символ ",") внутри. Для отрицательных чисел первым элементом набора служит логический символ "минус". Базисные арифметические операторы ЛОСа позволяют работать с десятичными записями вплоть до 1000 знаков; в основном, реализованные на ЛОСе вспомогательные арфметические процедуры не используют такой возможности и ориентированы на меньшее (хотя тоже значительное) число знаков.

Кроме уже упоминавшихся ранее базисных операторных выражений ЛОСа, для работы с десятичными числами созданы операторные выражения "модуль(a)", "минимум(a b)", "максимум(a b)", "целаячасть(a)". Для возведения числа a в целую неотрицательную степень b служит выражение "возведениевстепень(a b)". Для проверки того, что число a больше числа b, имеется оператор "больше(a b)". Заметим, что для проверки нестрогих неравенств операторы не созданы (быстрее реализуется отрицание строгого неравенства).

Выражение "неполноечастное(a b)"позволяет находить целую часть от деления неотрицательного числа a на положительное число b.

Для перечисления в порядке возрастания всех целых чисел x, не меньших целого числа m и не больших целого числа n, служит оператор "Номера(m n x)". Бесконечное перечисление всех целых чисел x в порядке возрастания их модуля: $0, 1, -1, 2, -2, \ldots$ - реализуется оператором "Целое(x)". Перечисление обрывается, например, с помощью оператора "сброс"либо "обрыв".

Для усмотрения целых, натуральных и четных чисел созданы операторы "целое(a)", "натуральное(a)", "четное(a)".

Деление с остатком целого числа a на отличное от нуля целое b выполняется оператором "деление(a b m n)"; здесь m - неполное частное, n - неотрицательный остаток. Оператор "частное(m n k)"проверяет делимость целого числа m на ненулевое целое n и выдает частное k.

Выражение "нок(m n)"дает наименьшее общее кратное двух натуральных чисел m, n; выражение "нод(m n) наибольший общий делитель целых чисел m, n.

Оператор "извлечениекорня(a n x)"перечисляет результаты x извлечения из целого числа a корня степени n. Если корень не извлекается, то оператор ложен; для четного n выдаются два значения, отличающиеся знаком, для нечетного - одно.

Разложение натурального числа a в произведение степеней простых чисел выполняется оператором "множители(a b c)". Переменной b присваивается набор простых

чисел, переменной c - набор той же длины, у которого на соответствующих местах находятся показатели степени данных простых чисел в разложении a на множители.

Оператор "простыеделители(a b)"перечисляет все простые делители b натурального числа a. Оператор "делители(a b)"перечисляет все целочисленные (положительные и отрицательные) делители b целого отличного от нуля числа a. Оператор "делитель(a b c)"получает в качестве входной информации набор простых чисел a и набор b показателей степеней этих чисел. Переменная c перечисляет все натуральные делители произведения данных степеней.

Операторное выражение "числоделителей(a)"выдает число натуральных делителей целого числа a.

Оператор "степеньделителя(a b c d)"определяет для целого a и целого не равного по модулю единице b такое наибольшее натуральное c, что a делится на b в степени c (если a не делится на b, то оператор ложен). Переменной d присваивается частное от деления a на указанную степень b.

Ряд операторов обеспечивают приближенные вычисления с десятичными числами. Эти операторы реализованы так, чтобы получать гарантированные оценки снизу либо сверху - нужное направление указывается при обращении. В качестве меры точности задается число знаков после запятой, к которому приводится результат (сами эти знаки, в отличие от направления оценки, не гарантируются; при необходимости получить заданное число точных знаков следует использовать последовательность встречных оценок с увеличивающейся точностью).

Для приближенного деления десятичных чисел используется оператор "десделение(a b c d e)". У него a, b - делимое и делитель; c - число знаков после запятой, к которому приводится результат. d - результат деления; $e = 0$, если этот результат точный, иначе $e = 1$. Результат берется с недостатком в смысле модуля. Альтернативный способ деления - применение операторного выражения "десчастное(a b c d)". Здесь a, b - делимое и делитель; c - число знаков после запятой в результате; d - указатель направления округления. Логический символ "меньше"означает округление в сторону уменьшения значения; логический символ "не в сторону увеличения. Округление к ближайшему выполняется при указателе, равном 0. Эти же соглашения об округлении используются и в последующих операторах.

Для округления числа a применяется операторное выражение "округление(a b c)", у которого b - число знаков после запятой, до которого происходит округление; c - указатель направления округления.

Для извлечения квадратного корня из числа a служит операторное выражение "квадркорень(a b c)". Здесь b - указатель направления округления; c - число знаков после запятой. Аналогичного вида выражение "экспонента(a b c)"дает значение экспоненты.

Операторное выражение "натурлог(a b c)"позволяет вычислять натуральный логарифм a. Указатель направления округления c и число знаков после запятой b здесь переставлены местами.

Для вычисления значений тригонометрических операций (синус, косинус, тангенс, котангенс, арксинус, арккосинус, арктангенс) используется оператор "тригоценка(a b c d e)". Здесь a - значение аргумента; b - логический символ для тригонометрической операции; c - число знаков после запятой; d - указатель направления округления. Переменной e присваивается результат.

Объединяет отдельные процедуры для приближенных вычислений с десятичными числами оператор "числоценка(a b c d)". Он позволяет находить приближенное значение константного числового терма, вхождение первого символа которого есть

a. Этот терм построен из числовых констант, символов "е"и "пи"с помощью арифметических операций, возведения в степень, логарифмов, модулей и указанных в операторе "тригоценка"тригонометрических функций. b - число знаков после запятой; c - указатель направления округления. Результат присваивается переменной d.

6.4.2 Термы, представляющие числовые константы

Ряд операторов выполняют арифметические вычисления, получая входные данные в виде термов для числовых констант и возвращая результат в том же формате. Напомним, что для представления неотрицательного десятичного числа в виде терма имеются две возможности: однобуквенный терм, состоящий из логического символа-цифры, и терм вида "величина($a_1 \dots a_n$)". где $a_1, \dots, a_n$ - цифры, быть может, с единственным символом ",". Для представления отрицательного числа используется терм вида "минус(A)", где A - одного из указанных выше типов. Термы, представляющие указанным образом десятичные числа, называем десятичными записями.

Операторное выражение "сложить(a)"получает выражение "плюс($t_1 t_2$)"для десятичных записей t_1, t_2 и выдает десятичную запись (терм), значением которого служит сумма t_1, t_2. Операторное выражение "сложитьдроби(a)"получает выражение аналогичного вида, но каждое слагаемое - либо десятичная запись, либо (с точностью до знака) отношение десятичных записей натуральных чисел. Значением служит терм - десятичная запись либо простая дробь, полученная в результате сложения t_1, t_2. Операторное выражение "умножить(a)"получает выражение "умножить(t_1 t_2)"для десятичных записей t_1, t_2. Значением его является десятичная запись произведения. Операторное выражение "натурстепень(a)"получает выражение "степень($t_1 t_2$)", где t_1 - десятичная запись, t_2 - десятичная запись целого неотрицательного числа. Значением выражения является десятичная запись результата возведения t_1 в степень t_2.

Для проверки того, что v - вхождение первого символа десятичной записи, используется оператор "десзапись(v)". Чтобы перейти от вхождения v первого символа десятичной записи к десятичному числу, служит операторное выражение "числзначение(v)". Для получения десятичной записи десятичного числа a служит операторное выражение "десзапись(a)".

Оператор "деспорядок(a b c)"позволяет по вхождению a первого символа десятичной записи положительного числа получать десятичную запись b натурального числа, полученного из a домножением на наименьшую неотрицательную целую степень десяти, и показатель c этой степени (в формате десятичного числа). Операторное выражение "сдвигзапятой(a b)"получает вхождение a первого символа десятичной записи и натуральное десятичное число b. В a предпринимается сдвиг запятой на b разрядов влево; значением выражения служит десятичная запись результата сдвига.

Оператор "стандстепень(a b c x)"получает выражение a, имеющее вид дробной степени натурального числа. b, c - числитель и знаменатель показателя этой степени (в формате десятичных чисел). Переменной x присваивается набор выражений - сомножителей стандартного представления a в виде произведения дробных степеней различных простых чисел.

Для усмотрения возможности представления числа в виде простой дроби служит оператор "простаядробь(a b c)". У него a - вхождение первого символа выражения T. Если это выражение является десятичной записью либо имеет вид "дробь(A B)"; "минус(дробь(A B))", где A,B - десятичные записи натуральных чисел, то находятся

целое b и натуральное c, такие, что дробь b/c несократима и равна T. В противном случае оператор ложен. Здесь b, c выдаются в формате десятичных чисел. Операторное выражение "десдробь(a b)"получает целое и натуральное десятичные числа a, b. Осуществляется сокращение дроби a/b; если знаменатель результата есть степень десяти, то выполняется переход к десятичной дроби. Значением выражения служит терм для результата сокращения дроби. Операторное выражение "сокращдроби(a)"получает терм a вида A/B, где A, B - десятичные записи. Его значением является выражение - результат приведения a к виду простой несократимой дроби (возможно, со знаком минус).

Операторное выражение "дробнаязапись(a)"получает десятичную запись a, имеющую символ запятой. Значением его служит терм - результат преобразования a к виду простой несократимой дроби (возможно, со знаком "минус"). Операторное выражение "составнаядробь(a)"получает выражение a вида "дробь(A B)", где A, B - десятичные записи натуральных чисел, первое из которых не меньше второго. Значением его служит терм - результат представления a в виде суммы натурального числа и простой несократимой дроби, числитель которой меньше знаменателя.

Оператор "констцелое(a)"получает терм либо вхождение первого символа терма a. Он истинен, если этот терм является десятичной записью целого числа. Оператор "констдробь(a)"получает аналогичную ссылку a на терм и является истинным, если этот терм имеет вид дроби с натуральными числителем и знаменателем либо получен из такой дроби добавлением внешнего знака "минус". Оператор "дробнаявеличина(a)"получает вхождение a первого символа некоторого выражения. Он истинен, если данное выражение является десятичной записью либо (с точностью до знака "минус") отношением десятичных записей.

6.4.3 Вычисления с плавающей запятой

Для ускоренного вычисления значения выражения в формате с плавающей запятой предусмотрен базисный оператор "Выч(программа A)", реализующий микропрограмму A. Оператор "вычпрог(a b c)"позволяет создавать такую микропрограмму b для вычисления значения выражения a. Выходной переменной c присваивается набор вспомогательных констант, используемых данной микропрограммой. Вход-выходной набор микропрограммы b состоит из набора значений свободных переменных выражения a, упорядоченных по возрастанию их номеров (именно так эти переменные упорядочиваются операторами "параметры", "списокпеременных"); далее идет набор констант c, и в конце размещается результирующее значение.

Операторное выражение "вычконст(a)"позволяет определять в формате с плавающей запятой приближенное значение константного терма a. Этот способ вычислений несколько более медленный, чем с использованием микропрограммы - здесь используется рекурсия по построению выражения a. Впрочем, он на порядки быстрее вычислений с десятичными числами.

Оператор "вычпроизв(a b c)"представляет собой усиленный вариант оператора "вычпрог". Он создает микропрограмму b для одновременного нахождения значения выражения a и всех первых частных производных этого выражения по его параметрам. При обращении к ней данные располагаются так же, как у оператора "вычпрог", однако после найденного значения a добавляются значения частных производных по параметрам a.

6.5 Общие процедуры интерфейса

Роль интерфейса в логической системе, обучаемой путем интенсивного интерактивного диалога, чрезвычайно велика. Чтобы сделать легко воспринимаемой информацию, объясняющую процесс решения задачи, упростить процесс программирования приемов, организовать работу с базой теорем, развивать комплекс средств автоматического синтеза приемов, и т.п. - приходится постоянно пополнять интерфейс системы новыми элементами. Здесь мы перечислим ряд наиболее часто используемых операторов интерфейса, указывая сведения, достаточные для их использования, но не приводя сколь-нибудь детального описания внутреннего устройства оператора. В наиболее значительных случаях (например, формульный и текстовый редакторы) такое описание будет приведено отдельно.

6.5.1 Клавиатура, мышь и меню

Обычная организация диалогового режима заключается в обращении к некоторому оператору ввода информации (например, базисному оператору ЛОСа "клавиатура (a)") и последующей обработке поступившего логического символа a с помощью цепочки проверок "равно(a S_i) иначе j". После каждого такого оператора проверки размещается программа процедуры P_i, выполняющей те или иные действия по нажатию клавиши с кодом S_i. Чтобы можно было обратиться к процедуре P_i из другой точки программы, имеется две возможности - либо создать для этой процедуры отдельный оператор, к которому и обращаться независимо из разных точек программы, либо обеспечить возможность автоматического переключения из другой точки программы (например, из другого диалогового режима) в данный диалоговый режим, и далее - устроить фиктивный ввод с клавиатуры символа S_i. В логической системе, как правило, реализуется именно вторая возможность, хотя для нее пришлось создать некоторую специальную технику, обеспечивающую "прыжки"из одного диалогового режима в другой через общую корневую точку интерфейса (ее роль играет обработчик команд от главного меню).

Прежде всего, для этого используется оператор "автоклавиатура(a)". Он проверяет наличие комментария (автоклавиатура A) к посылкам исходной задачи. Если такой комментарий имеется, то A - набор логических символов. Из этого набора извлекается первый символ s и передается выходной переменной a. Набор A укорачивается на единицу, и если становится пустым, то комментарий (автоклавиатура A) удаляется. Если данного комментария нет, то оператор работает так же, как оператор "клавиатура". Если организовать диалоговый режим с помощью оператора "автоклавиатура", то появляется возможность автоматически запускать отдельные его процедуры - для этого достаточно заблаговременно ввести комментарий (автоклавиатура ...) к посылкам исходной задачи и обеспечить вход в начальную точку диалогового режима. Если перед оператором "автоклавиатура"поместить оператор "повторение"(что обычно и делается), то можно программировать указанным образом выполнение заданных последовательностей операций диалогового режима.

Усиленный вариант оператора "автоклавиатура(a)" - оператор "автоменю(a)". Он позволяет вводить символ a одновременно с клавиатуры, от мыши и от меню в верхней части экрана. Использование им комментария (автоклавиатура ...) - такое же, как выше.

Для создания нового меню в верхней части экрана либо изменения ранее созданного меню предусмотрен специальный интерфейс. Чтобы войти в него, следует пе-

рейти к главному меню логической системы и либо нажать клавишу "м"(кир.), либо выбрать пункт "Ресурсы и установки", а затем - подпункт "Изменение меню". Тогда появится раздел оглавления, перечисляющего различные используемые в логической системе меню (тот, который рассматривался при предыдущем обращении к оглавлению). Это оглавление устроено таким образом, что каждое меню соответствует некоторой его ветви: в корне ветви находится пункт с текстом, объясняющим предназначение меню, а внутри ветви - выводимые на экран при прорисовке меню тексты. Непосредственно к корню ветви примыкает подраздел оглавления, тексты пунктов которого (как правило, состоящие из одного слова) выводятся в верхней части экрана. Неконцевым пунктам этого подраздела соответствует "выпадающее"подменю (в свою очередь, они тоже могут иметь подменю). Чтобы указать, на какую клавишу должен реагировать концевой пункт меню, следует выбрать соответствующий концевой пункт оглавления и нажать "курсор вправо". Тогда в верхней части экрана перерисовывается текст выбранного пункта, а под ним размещается (отделенный горизонтальной чертой) логический символ - код нужной клавиши. Для изменения этого кода (а при его отсутствии - ввода нового), далее нажимается "Enter" - появляется курсор текстового редактора. Чтобы найти логический символ - код нужной клавиши, здесь достаточно нажать F8 и сразу за этим - данную клавишу. По окончании набора символа снова нажимается "Enter". Если буква, при нажатии на клавишу которой срабатывает пункт меню, встречается в его тексте, то ее можно выделить подчеркиванием - для этого в тексте пункта оглавления перед буквой помещается знак &. Каждое представленное в оглавлении меню имеет свой адрес - набор номеров пунктов оглавления, перемещаясь по которым от корня оглавления, приходим к корню ветви меню. Номера здесь имеются в виду символьные (т.е. 10 - логический символ "десять", и т.д.). Эти адреса жестко закрепляются за меню для всех последующих обращений к ним из различных точек программы. Поэтому переставлять местами пункты оглавления меню (кроме пунктов внутри отдельного меню) нельзя.

Для вызова (прорисовка в верхней части экрана и активизация) меню, имеющего адрес a, используется оператор "сборкаменю(a)". Этот оператор создает комментарий (сборкаменю a) к посылкам исходной задачи, обеспечивающий возможность узнать впоследствии тип активизированного меню. При обращении к вспомогательным процедурам иногда приходится менять используемое меню. Для удобства восстановления меню при выходе из процедуры введены операторы "сохранениеменю(b)", "восстановлениеменю(b)". Первый из них находит комментарий (сборкаменю a) и создает на его основе комментарий (сохранениеменю a), а второй - находит такой комментарий и восстанавливает указанное в нем меню. Входная переменная b у обоих операторов - фиктивная (например, можно полагать ее равной 0).

6.5.2 Текстовый редактор

В системе используется текстовый редактор, интерфейс которого был описан выше (см. "Интерфейс ввода задач"). Он реализуется оператором "текстредактор(t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8)". Текстовый редактор использует для работы некоторый отрезок буфера текстов (массив, хранящий символы с указаним их цвета и цвета фона; см. описание базисного оператора "видео(...)"). t_1 есть номер первой ячейки этого отрезка (нумерация ячеек буфера текстов начинается с 0). t_2, t_3, t_4, t_5 суть координаты прямоугольной рамки, в которой осуществляется редактирование. Здесь t_2, t_3 - символьные номера столбца и строки для левого верхнего угла рамки, t_3, t_4 - символьные номера

столбца и строки правого нижнего угла. Как и обычно, отсчет строк ведется сверху вниз; нумерация строк и столбцов начинается с 0. t_6, t_7 - коды цвета фона и символов, изначально устанавливаемых при редактировании (как уже отмечалось, они могут изменяться в процессе редактирования). Выходной переменной t_8 присваивается номер последней занятой ячейки буфера текстов (пустая часть экрана в конце текста автоматически отбрасывается). Перед началом редактирования происходит расчистка используемой области. Чтобы сохранить для редактирования старое изображение, вместо указания на цвет символов t_7 берется равным 0. В этом случае изначальный цвет символов будет черным. В обоих случаях буфер текстов перед редактированием не расчищается. Если его специально не расчистить, то при перемещении курсора по пустому пространству рамки будут "проявляться"сохраненные в буфере символы. Поэтому перед обращением к текстовому редактору рекомендуется расчищать буфер, например, обращением к "видео(0)"(кроме случаев, когда внутри рамки и в буфере уже имеются согласованные изображения и редактирование выполняется с целью их коррекции). В действительности процедура "текстредактор"представляет собой нечто большее, чем просто текстовый редактор - в нее встроена часть интерфейса других блоков логической системы.

Текстовый редактор достаточно часто используется для набора логических символов, термов либо списков логических символов и термов. В этом случае после набора применяется базовый оператор "кодтекста(a b c)", который анализирует содержимое буфера текстов и либо выдает результат b - при $c = 0$, либо, при $c \neq 0$, выдает информацию (b,c) о допущенной ошибке. Чтобы прорисовать сообщение о такой ошибке в нижней правой части экрана, используется оператор "указатель-ошибки(b c $t_1t_2t_3t_4$)". Здесь t_1, t_2, t_3, t_4 - координаты рамки оператора "текстредактор", используемые для определения места ошибки в этой рамке. Данные об этом месте пересылаются в комментарий (теквхожд A B) к посылкам исходной задачи. Если после обнаружения ошибки реализуется откат к повторному редактированию (с сохранением ранее набранного текста), то в начале его текстовый редактор автоматически переместит курсор к месту ошибки.

Для сравнения двух фрагментов буфера текстов, расположенных между позициями a, b и c, d, служит оператор "равнтекст(a b c d)". Оператор "большиебуквы(a b)"заменяет все маленькие русские буквы, расположенные в буфере текстов начиная с позиции a и кончая позицией b, на большие русские буквы. Операторное выражение "малбуква(a)"по коду a большой буквы (латинской либо кириллической) выдает код соответствующей малой буквы. Операторное выражение "болбуква(a)"выполняет обратный переход. Если логический символ a в двух последних случаях не удовлетворяет указанным условиям, то возвращается значение a.

6.5.3 Формульный редактор

Интерфейс формульного редактора был описан выше ("интерфейс ввода задач"). Этот редактор реализуется оператором "формредактор($t_1t_2t_3t_4t_5t_6t_7t_8t_9t_{10}$)". Данный оператор используется в двух различных режимах - либо для ввода нового терма, либо для создания структуры данных, позволяющей прорисовать на экране уже имеющийся терм. У него t_1, t_2, t_3, t_4 - координаты рамки редактирования (в таком же формате, как у текстового редактора); t_5 - код цвета фона; t_6 - либо код цвета символов (при ручном вводе нового терма), либо ранее созданный терм, который нужно прорисовать на экране. Выходной переменной t_7 присваивается так называемое дерево указателей - структура данных, позволяющая непосредственно прорисовать

терм на экране; выходной переменной t_8 - введенный при редактировании терм; t_9 - символьный номер для правого столбца собственной рамки набранного терма; t_{10} присваивается пара (N_1N_2) символьных номеров верхней и нижней строк собственной рамки терма. Если оператор реализуется во втором режиме (то есть t_6 - терм), то он ничего не прорисовывает на экране, а лишь создает структуру данных t_7.

Для прорисовки терма по заданному дереву указателей, а также для выполнения ряда других действий с этим деревом используется оператор "блокредактора(n a_1 $\ldots$ a_m)". Фактически он представляет собой семейство независимых подоператоров, причем n - номер (символьный) подоператора этого семейства. Не уточняя здесь формата самого дерева указателей (это будет сделано в разделе, специально посвященном программной реализации формульного редактора), приведем краткое описание некоторых из подоператоров оператора "блокредактора".

При $n = 2$ подоператор позволяет корректировать дерево указателей для параллельного сдвига изображения. У него a_1 - пара (d_1d_2), указывающая направление d_1 (0 - влево, 1 - вправо) и величину d_2 (число пикселей, символьный номер) сдвига по горизонтали; a_2 - пара (e_1e_2), указываюшая направление e_1 (0 - вверх, 1 - вниз)и величину e_2 сдвига по вертикали. a_3 есть максимально допустимая при сдвиге величина номера строки изображения (при превышении ее оператор ложен), a_4 - корректируемое дерево указателей.

При $n = 3$ подоператор осуществляет прорисовку терма по его дереву указателей. В этом случае a_1, a_2 - цвет фона и символов; a_3 - дерево указателей. a_4 - либо 1 (обычная прорисовка), либо 0 (виртуальная прорисовка, при которой на экране ничего не изменяется и которая служит для определения позиции курсора после прорисовки терма), либо символьный номер $n + 1$, где n - номер строки, начиная с которой происходит явная прорисовка (вышележащая часть терма на экран не выводится). Последние две возможности являются техническими и используются внутри программы формульного редактора. Выходным переменным a_5, a_6 присваиваются символьные номера столбца и строки, на которых размещается курсор по окончании прорисовки терма.

При $n = 8$ подоператор осуществляет перевод дерева указателей, полученного при ручном вводе терма, в терм. Здесь a_1 - дерево указателей; выходной переменной a_2 присваивается терм. При обнаружении ошибки в наборе терма оператор ложен.

При $n = 9$ подоператор осуществляет определение самой верхней из строк, занятой символами подтерма. У него a_1 - дерево указателей; a_2 - выходная переменная, которой присваивается символьный номер указанной строки.

Формульный редактор использует некоторое фиксированное сопоставление переменным их обозначений. Переменные от первой до двадцать пятой обозначаются последовательными малыми латинскими буквами; начиная с двадцать шестой идут большие латинские буквы; далее - те же латинские буквы с индексами. Оператор "видпеременной(a b c)"позволяет находить по переменной ее обозначение и обратно. Если $b = 0$, то он определяет по переменной a пару c, первый элемент которой есть логический символ - код латинской буквы, второй элемент - 0 либо индекс в формате десятичного числа. Если $b = 1$, то он по паре c указанного вида выдает соответствующую переменную a. Чтобы прорисовать заданную переменную в заданном месте экрана так, как это делает формульный редактор, служит оператор "записьпеременной(t_1 t_2 t_3 t_4 t_5 t_6 t_7)". У него t_1, t_2 - символьные номера столбца и строки для прорисовки; t_3 - переменная; t_4, t_5 - цвет фона и символов. Выходным переменным t_6, t_7 присваиваются символьные номера столбца и строки позиции, расположенной после прорисованной переменной. В некоторых ситуациях

(например, в редакторе приемов ГЕНОЛОГа) приходится одновременно работать с формульным и текстовым редактором, где переменные обозначаются как xi, и тогда возникает необходимость в таблице-подсказке, указывающей соответствие между разными обозначениями одной и той же переменной. Такая таблица прорисовывается оператором "обозначпеременных(a b c)". У него a - терм либо набор термов; b - номер первой строки на экране, начиная с которой будет прорисовываться соответствие между формульными и текстовыми обозначениями. Таблица представляет собой список разделенных запятыми пар "формульное обозначение переменной - текстовое ее обозначение"; она прорисовывается голубым цветом. Выходной переменной c присваивается символьный номер первой строки под таблицей.

Чтобы можно было выделять подтермы прорисованного терма, перекрашивая их в другой цвет, нужно определять соответствие между вхождениями в терм и поддеревьями дерева указателей. Для этого используется оператор "формподтерм(a b c d)". Здесь a - терм; b - его дерево указателей; c - вхождение подтерма в a. Выходной переменной d присваивается поддерево дерева b, необходимое для перекраски c. Перекраска выполняется оператором "блокредактора(3 ...)", в котором выбран требуемый цвет символов. Заметим, что не для всех вхождений в терм такое выделение поддерева указателя возможно; если оно невозможно, то оператор ложен. Приведем еще один оператор аналогичного типа (он используется оператором "формподтерм") - "формоперанды(a b c)". Здесь a - дерево указателей; b - вхождение терма, которому соответствует a. Переменная c перечисляет пары "дерево указателей для максимального допускающего отдельную прорисовку подтерма терма b - вхождение корня этого подтерма".

При решении задач иногда приходится вводить новые переменные. Чтобы при прорисовке этих переменных формульным редактором придерживаться обычных соглашений, принятых для обозначения объектов заданного типа (целые числа - буквы $m, n, k, \ldots$; точки плоскости - $A, B, C, \ldots$; функции - $f, g, h, \ldots$, и т.п.), можно использовать оператор "обозначения(a b c)". У него a - указатель способа выбора новой переменной; b - набор информационных элементов, уточняющих этот способ. Выходной переменной c присваивается новая переменная. Рассматриваются следующие способы выбора новой переменной:

1. a = "задача". В этом случае b - тройка (задача Z - переменная x - набор переменных A). Выбирается такая новая переменная y, которая будет замещать в задаче Z переменную x, и при этом не входит в набор A. Здесь оператор применяется в ситуации, когда обозначение переменной, быть может, неудачное, уже было введено, и требуется выполнить его коррекцию.

2. a = "неизвестная". В этом случае b - список переменных, от которых должна отличаться новая неизвестная.

3. a = "теорема". В этом случае b - тройка (терм T - переменная x - набор переменных A). Выбирается такая новая переменная y, которая будет замещать в терме T переменную x, и при этом не входит в набор A. Как и в случае 1), здесь имеет место коррекция уже введенного, быть может, неудачного, обозначения для переменной x теоремы T из базы теорем.

4. a = "целое". В этом случае b - список переменных, от которых должен отличаться новый целочисленный параметр.

5. a = "функция". В этом случае b - задача, для которой выбирается новый символ функциональной переменной.

6. a = "точка". В этом случае b - список переменных, от которых должно отличаться обозначение новой точки.

При обучении системы и расширении ее логического языка приходится постоянно расширять формульный редактор. Для тех случаев, когда достаточно использовать в "формульной"записи слово, обозначающее новый логический символ, эта операция автоматизирована - см. разделы, посвященные языку ГЕНОЛОГ. Если же нужно вводить новую символику, то для расширения формульного редактора может понадобиться более глубокое знакомство с его архитектурой. Эта архитектура описывается далее в специальном разделе.

6.5.4 Геометрический редактор

Интерфейс геометрического редактора был описан в разделе, посвященном редактированию задач. Этот редактор предназначен для построения простейших чертежей в задачах по планиметрии. Он реализуется оператором "геомредактор(t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8)". Здесь t_1, t_2, t_3, t_4 - координаты рамки редактирования (столбец и строка верхнего левого угла; столбец и строка правого нижнего угла); t_5, t_6 - цвет фона и цвет линий. t_7 - структура данных чертежа, представляющая собой пару (A_1, A_2). A_1 есть набор пятерок $(xa_1a_2b_1b_2)$, указывающих координаты (a_1, a_2) различных точек x чертежа и координаты (b_1, b_2) позиций, на которых размещаются обозначающие точки переменные. x - переменная, обозначающая точку; a_1, a_2, b_1, b_2 - символьные номера. В координате сначала идет номер столбца, затем - номер строки. A_2 есть набор информационных элементов, определяющих прорисовку. Типы этих элементов таковы: (точка x) - прорисовка точки x; (отрезок x_1 x_j) - прорисовка отрезка с концами x_i, x_j; (прямая x_i x_j) - прорисовка прямой, проходящей через точки x_i, x_j (до границ рамки); (окружность x_i a) - прорисовка окружности с центром в точке x_i и радиусом a. Здесь a - символьный номер, определяющий величину радиуса в пикселях. Оператор "геомредактор"сначала выполняет прорисовку чертежа; если $t_8 = 0$, то на этом его работа завершается. Если же $t_8 = 1$, то после прорисовки реализуется интерфейс редактирования чертежа. При создании нового чертежа вводится структура данных с пустыми списками A_1, A_2.

Оператор "блокчертежа($na_1 \ldots a_m$)"представляет собой семейство вспомогательных подоператоров для работы со структурой данных чертежа. Приведем здесь несколько наиболее важных подоператоров.

При $n = 1$ значением a_1 служит структура данных чертежа; выходная переменная a_2 выдает номер нижней строки, занятой чертежом.

При $n = 2$ реализуется коррекция структуры данных чертежа, соответствующая его параллельному переносу. Значением a_1 здесь служит набор (геомредактор A_1 A_2 A_3), обычно используемый в качестве комментария к посылкам задачи, указывающего на сопровождающий задачу чертеж. A_1 - структура данных чертежа; A_2, A_3 - символьные номера верхней и нижней строк чертежа. Сдвиг чертежа по горизонтали определяется парой $a_2 = (b_1, b_2)$; в случае $b_1 = 0$ происходит сдвиг влево, при $b_1 = 1$ - сдвиг вправо; b_2 - величина сдвига (символьное число). Сдвиг по вертикали определяется парой $a_3 = (c_1, c_2)$. $c_1 = 0$ - сдвиг вверх, $c_1 = 1$ - вниз; c_2 - величина сдвига.

При $n = 5$ значением a_1 является структура данных чертежа; переменной a_2 присваивается символьный номер верхней строки чертежа.

Чтобы выбирать на чертеже позицию для размещения буквенного обозначения заданной точки, служит оператор "выборпозиции(t_1 t_2 t_3 t_4 t_5 t_6)". У него t_1 - структура данных чертежа; t_2, t_3 - столбец и строка для положения новой точки на чертеже. Позиция для прорисовки обозначения этой точки выбирается исходя из соображений

наименьшего наложения этого обозначения на другие элементы чертежа. Переменным t_4, t_5 присваиваются столбец и строка для прорисовки; t_6 - оценка степени наложения. При определении последней используется оператор "оценкапозиции(t_1 t_2 t_3 t_4 t_5 t_6)". У него t_1 - структура данных чертежа; t_2, t_3 - координаты для прорисовки обозначения точки; t_4, t_5 - координаты точки. Переменной t_6 присваивается оценка степени наложения.

Для сохранения чертежа в информационном блоке он перекодируется в набор термов, который и заносится в некоторый логический терминал. Используются термы следующих типов: "переменная(x $a_1a_2b_1b_2$)" - код пятерки ($xa_1a_2b_1b_2$) из A_1; "точка(x)"; "отрезок(x_ix_j)"; "прямая(x_ix_j)"; "окружность(x_ir)" - коды элементов A_2; "начало(n_1)"; "конец(n_2)"; "левыйкрай(n_3)"; "правыйкрай(n_4)" - указатели на верхнюю, нижнюю, левую и правую границы рамки чертежа. n_1, n_2, n_3, n_4 суть символьные номера; указатели левой и правой границ могут отсутствовать. Кроме того, могут использоваться некоторые дополнительные пометки (например, логический символ "чертеж" - указатель на то, что чертеж был построен решателем самостоятельно).

Оператор "регистрациячертежа(a b c d)"выполняет запись чертежа в активый информационный блок. У него a - указатель-список данного блока; b - структура данных чертежа; c - пара (верхняя строка рамки чертежа - нижняя строка рамки); d - метка, по которой описание чертежа заносится в указатель a. Оператор "чтениечертежа(a b c d)"получает в качестве входных данных указатель-список a активного информационного блока и метку b, по которой от a имеется переход к логическому терминалу с описанием чертежа. Переменной c присваивается структура данных чертежа, переменной d - пара (верхняя строка рамки - нижняя строка рамки) либо (если в логическом терминале имелись элементы "левыйкрай(...)", "правыйкрай(...)") четверка (верхняя строка - левый столбец - правый столбец - нижняя строка).

6.5.5 Просмотр термов и списков

Оператор "просмотртерма(a b c d)"реализует интерфейс выбора подтерма в терме, прорисованном текстовым редактором (то есть в скобочной записи). Перед обращением к этому оператору терм уже должен быть прорисован на экране. a,b - символьные номера столбца и строки для первой буквы терма, c - исходная позиция в буфере текстов, начиная с которой был размещен при прорисовке этот терм, d - пара (терм - указатель заранее выделенного подтерма). Последний указатель при отсутствии выделения равен 0, иначе он представляет собой вхождение первого символа выделенного подтерма. Прежде всего оператор перекрашивает фон терма в светло-голубой, меняет цвет символов выделенного подтерма (если он имелся) в светло-пурпурный, и выделяет желтым цветом фона первый корневой операнд всего терма. Далее включается интерфейс выбора подтерма: нажатия клавиш "курсор влево - курсор вправо"позволяют перемещаться в группе операндов одной и той же операции. Для исходной ситуации это означает возможность выбора, путем перекраски фона в желтый цвет, нужно корневого операнда терма. Если нужно выбрать операнд текущей выбранной операции, то нажимается клавиша "курсор вниз это приводит к перекраске в особый цвет фона первого операнда этой операции; далее клавишами "курсор влево-курсор вправо"выбирается нужный ее операнд. Цвета фона при перемещении вглубь терма изменяются циклически: светло-голубой, желтый, светло-зеленый, светло-пурпурный, светло-серый. Если нужно вернуться от операндов к внешней операции, то нажимается клавиша "курсор вверх". Нажатие ее для уровня корневых операндов означает выход из процедуры (оператор при этом будет истин-

ным). Для немедленного выхода из процедуры нажимается "Esc"(оператор в этом случае истинен). Можно передавать указатель выбранного вхождения (терм вида "фикс(...)"внешним процедурам. Для этого перед обращением к оператору следует ввести комментарий (указательвхождения 0) к посылкам исходной задачи. Нажатие любой клавиши, отличной от перечисленных выше, вызовет (при наличии такого комментария) обрыв просмотра и замену последнего разряда комментария на указатель "фикс(...)". Оператор в этом случае будет ложен. При отсутствии комментария (указательвхождения A) оператор выполняет ряд дополнительных действий. В частности, выбор подтерма может быть выполнен однократным нажатием левой клавиши мыши при размещении ее курсора на начальной позиции подтерма. Нажатие правой клавиши мыши вызовет переход к просмотру справочной информации о выделенном (мышью либо клавиатурой) логическом символе. Такой же переход вызывает нажатие клавиши F3. Наконец, ряд специальных функций оператора "просмотртерма"связан с конкретными блоками системы, использующими этот оператор (например, с отладчиком ЛОСа). Оператор "просмотртерма"можно использовать для прорисовки на экране всей многоцветной схемы выделения заданного подтерма (помимо одноцветного выделения через d) - для этого перед обращения к нему достаточно создать комментарий (автоклавиатура A), у которого A - последовательность нажатий клавиш, необходимых для входа в указанный подтерм. Если после прорисовки нужно сразу выйти из оператора, то в конце A добавляется нажатие клавиши "Esc". Создавать комментарий (автоклавиатура A) в указанной ситуации можно с помощью оператора "входвтерм(a b c)". У него b - то вхождение в терм a, которое нужно выделить. Если $c = 0$, то в конце A добавляется код клавиши "Esc".

Чтобы определять вхождение того подтерма, который выделяется курсором мыши, можно применять оператор "вхождениемыши(t_1 t_2 t_3 t_4 t_5 t_6 t_7)". У него t_1, t_2 - символьные номера столбца и строки для начальной позиции терма t_4, выданного на экран в скобочной записи; t_3 - исходная позиция в буфере текстов для этого терма; t_5, t_6 - символьные номера столбца и строки для текущей позиции курсора мыши. Если этот курсор расположен в области прорисовки терма, то t_7 присваивается соответствующее вхождение в терм, иначе оператор ложен.

Для просмотра заданного списка термов и выбора нужного терма служит оператор "просмотртермов(a b c d e)". У него a - набор термов; b - символьный номер изначально выделенного терма в наборе a (нумерация начинается с 1); c - 0 либо структура данных для прорисовки списка термов, которая создается самим оператором. Если при просмотре был выбран некоторый терм (курсор вправо на выделенном терме), то выходной переменной d присваивается символьный номер выбранного терма, выходной переменной e - результирующая структура данных для прорисовки списка термов. Эта структура данных при повторном обращении к оператору может использоваться как входной параметр c и ускорить очередную прорисовку. Если выход из просмотра терма произошел по нажатии клавиши "Esc", либо "курсор влево", либо по нажатии правой клавиши мыши, то оператор ложен. При прорисовке термы нумеруются; если это возможно, происходит их выдача в формульном виде, иначе - в скобочном. Номер выделенного терма перекрашен в голубой цвет. Смена выбранного пункта - либо клавиши "курсор вверх", "курсор вниз", либо постраничный шаг - "Page Up", "Page Down".

Для просмотра и (частично) редактирования списка объекта произвольных типов используется оператор "просмотрсписка(a)". У него a - структура данных просмотра - набор информационных элементов, определяющих просматриваемый список и режим просмотра; типы этих элементов приводятся ниже. Интерфейс данного опе-

ратора развит в значительно большей степени, чем у оператора просмотра списка термов. Помимо прорисовки термов (в формульной либо скобочной записи), происходит прорисовка отдельных логических символов, наборов, вхождений в наборы и термы, чисел в формате с плавающей запятой. Каждый элемент просматриваемого списка выдается после своего номера начиная с отдельной строки. Если элемент - логический символ, после номера идет закрывающая скобка, иначе - точка. При прорисовке набора те его элементы, которые суть логические символы и символы переменных, изображаются непосредственно; прочие элементы обозначаются одной из букв: "т" - терм, "в" - вхождение, "н" - набор. При прорисовке вхождения в набор либо терм происходит выделение разряда набора либо подтерма по заданному вхождению голубым цветом. Число в формате с плавающей запятой выдается как терм "набор(M P)"; M, P - десятичные записи, где обычно P равно 0. Число, определяемое таким термом, есть произведение M на десять в степени P. Номер текущего просматриваемого элемента выделен голубым цветом; смена этого элемента происходит с помощью клавиш "курсор вверх" - "курсор вниз"и (постраничная прокрутка) "Page Up" - "Page Down". Переход к первому либо последнему пункту части списка, прорисованной на экране - "Ctr-курсор вверх"и "Ctr-курсор вниз". Оператор имеет два буфера - для единичного выбранного элемента либо группы элементов. Занесение текущего элемента в первый буфер (информационный элемент (терм T)в структуре данных просмотра) с исключением из него ранее имевшегося элемента происходит при нажатии "Ctr - ы"; занесение текущего элемента во второй буфер (информационный элемент (буфер B)) либо исключение из него - при нажатии "б". Номера элементов, занесенных во второй буфер, выделяются красным цветом. Для перехода к общему формульному режиму просмотра (выбран по умолчанию) нажимается "м", для перехода к скобочному режиму - "с". Выход из просмотра осуществляется следующими способами: (а) нажатием клавиши "курсор вправо"на текущем пункте - в этом случае оператор истинен; (б) нажатием клавиш "Esc"либо "пробел"либо "курсор влево в этом случае оператор ложен. Оператор предоставляет возможность изменять просматриваемый список. Если нужно добавить в конце списка новый терм формульным редактором, нажимается клавиша "Enter"; если это нужно сделать текстовым редактором, клавиша "т". При добавлении в конце списка набора логических символов и термов также используется текстовый редактор, но здесь должна быть нажата клавиша "н". Чтобы добавить в конце списка элемент T из буфера (терм T), служит клавиша "ы". Аналогичные операции, выполняемые не в конце списка, а перед текущим его элементом, инициируются клавишами "Ф", "Т", "Н", "Ы". Если нужно изменить текущий элемент, используются клавиши "Ctr-ф"(формульный редактор), "Ctr-т", "Ctr-н"(текстовый редактор для термов и наборов). Исключение текущего элемента списка - "Ctr-Del". Для просмотра соответствия обозначений использованных в списке переменных при прорисовке их формульным и текстовым редакторами нажимается клавиша "х". По завершении просмотра и редактирования, информация о выбранных элементах (в том числе о последней текущей позиции списка) и об имевших место изменениях извлекается из соответствующих информационных элементов структуры данных просмотра. Перечислим типы этих элементов:

1.(набор A) - A есть просматриваемый набор;

2.(позиция N) - N есть символьный номер текущего просматриваемого разряда набора A (нумерация начинается с 1);

3.(область h_1 h_2) - h_1, h_2 суть символьные номера верхней и нижней строк полосы, отведенной для просмотра списка;

4."текстредактор" - термы и переменные выдаются на экран при помощи тексто-

вого редактора;

5."сквознойпросмотр" - выход по нажатии клавиши "курсор вправо"осуществляется только на позициях списка, являющихся наборами либо вхождениями в наборы;

6."выход" - немедленный выход после прорисовки начала списка, без обращения к клавиатуре;

7."номера" - блокировка прорисовки номеров пунктов;

8.(терм T) - буфер для сохранения выбранного терма; при нажатии клавиши "курсор вправо"на текущем терме происходит замена T на этот терм.

9."обл" - при нажатии клавиши "пробел"выход из процедуры происходит по значению "истина".

10."текстоваязадача" - выход по нажатии клавиши "Enter";

11. (примечание K) - K есть набор примечаний к просматриваемому набору;

12. (см S) - S есть набор сопровождающих термы основного набора объектов. При удалении элемента основного набора такое же удаление выполняются в наборе S. Если элемент набора S - набор, содержащий логический символ "плюс", то при прорисовке на экране после номера соответствующего пункта ставится знак +; если элемент набора S - набор, содержащий логический символ "минус", то после этого номера ставится знак $-$. Если элемент набора S - набор, содержащий логический символ "см", то после номера ставится знак вопроса; если элемент набора S - набор, содержащий логический символ "См", то после номера ставится восклицательный знак.

13. (буфер B) - в накопителе B содержится список номеров выделенных при просмотре списка элементов.

Обязательными при обращении к оператору являются элементы типов 1-3, к которым должен быть добавлен также набор (видео пустоеслово).

Оператор "просмотрсписка"не позволяет просматривать компоненты наборов своего списка. Чтобы такой просмотр стал возможен, применяется оператор "сквознойпросмотр(a)". У него a - та же структура данных просмотра, что и у оператора "просмотрсписка". Интерфейс отличается от интерфейса последнего оператора лишь тем, что по нажатии на текущем пункте клавиши "курсор вправо", если этот пункт представлял собой набор либо вхождение в набор, происходит переход к просмотру этого набора. Если текущий пункт не являлся набором, то нажатие клавиши "курсор вправо"игнорируется. Для возвращения в просмотр внешнего набора нажимается любая из клавиш "курсор влево", "Esc", "пробел", "End".

6.5.6 Текстформульный редактор

Для создания и просмотра форматированных текстов с вставленными в них формулами и чертежами применяется текстформульный редактор. В системе с его помощью реализованы общий справочник и (частично) интерфейс просмотра задач, в том числе интерфейс редактирования задач на текстовый анализ. Текст составляется из элементов, создаваемых текстовым, формульным и геометрическим редакторами. Между этими элементами могут вставляться указатели абзаца, пустой строки и пропуска заданного числа позиций (без перехода на новую строку). Реализуется текстформульный редактор оператором "текстформ(a b)", который получает в качестве входного данного текстформульную структуру a и набор b информационных элементов, определяющих режим редактирования. Текстформульная структура содержит полное описание редактируемого текстформульного массива, причем это описание

хранится не в файлах, а в оперативной памяти. Поэтому после применения оператора "текстформ"необходимо сохранить в файлах имевшие место изменения - такие действия предусматриваются особо в тех блоках интерфейса, которые обращаются к текстформульному редактору.

Начнем с описания интерфейса текстформульного редактора. Завершение редактирования происходит единственным способом - по нажатии клавиши "End"; в этом случае результаты редактирования будут сохранены. Для отмены редактирования можно использовать клавишу "курсор влево"либо "Esc естественно, без сохранения результатов редактирования. При просмотре текстформульного массива можно применять обычную прокрутку с помощью клавиш "курсор вверх", "курсор вниз"и клавиш "PageUp", "PageDown".

В различных операциях используется выделение элемента текстформульного массива. Такое выделение осуществляется мышью: ее курсор подводится в поле рассматриваемого элемента, и нажимается левая клавиша мыши. Выделенный элемент перекрашивается в синий цвет. Повторное нажатие левой клавиши мыши отменяет выделение элемента. При выделении нового элемента старое выделение автоматически сбрасывается. Выделение элементов требует осторожности, так как оно подготавливает выполнение различных операций, и случайное нажатие клавиши может привести к нежелательным последствиям.

Для ввода нового текста, формулы либо чертежа используются, соответственно, уже описанные ранее текстовый, формульный либо геометрический редакторы. Если добавление нового элемента происходит в конце текстформульного массива, то нужно с помощью прокрутки освободить в нижней части экрана достаточное место и нажать одну из следующих клавиш:

а) Ввод нового текста - либо нажать "Enter", либо нажать "т".

б) Ввод новой формулы формульным редактором - нажать "ф" маленькое.

в) Ввод новой формулы текстовым редактором - нажать "Ф" большое.

г) Ввод чертежа - нажать "ч".

Это обеспечивает вход в необходимый редактор; по завершении редактирования элемент добавляется непосредственно после предыдущего (если оставалось место в строке, занятой предыдущим элементом, то - начиная с первой свободной позиции этой строки; исключение составляет лишь чертеж, который всегда помещается начиная с новой строки). Возможна отмена редактирования (Esc).

Если нужно вставить новый элемент перед ранее созданным элементом, то этот элемент сначала выделяется, и затем выполняются указанные выше действия а) - г).

Если нужно, чтобы новый элемент начинался с новой строки, то перед его вводом нажимается клавиша "а"(абзац). Если элемент уже был введен, и требуется сделать его начинающимся с новой строки, то этот элемент выделяется левой клавишей мыши и нажимается "а". Аналогичным образом вводится пропуск строки - перед набором нового элемента нажимается "с"либо после выделения ранее набранного элемента нажимается "с".

Для вставки пробелов перед выделенным элементом следует выбрать пункт меню "Пробелы"либо нажать клавишу пробела, после чего нажать левую клавишу мыши на той позиции, до которой следует сдвинуть начало выделенного элемента. Сдвиг осуществляется только в рамках одной строки, без перехода на следующую строку.

Для удаление введенных перед выделенным элементом пробела, абзаца либо пропуска строки следует нажать клавишу "Ctr-п".

Для изменения текста, формулы либо чертежа предпринимается нажатие правой клавиши мыши в области данного элемента. Если нужно изменить формулу фор-

мульным редактором, то предварительно эта формула выделяется. Иначе ее изменение будет выполняться текстовым редактором.

Для удаления элемента он сначала выделяется, а затем нажимается "Ctr-Del". Пробелы, абзацы и пропуски перед удаленным элементом сохраняются. Поэтому перед удалением концевого элемента целесообразно сначала удалить предшествующие ему пробелы и пропуски.

Для перестановки выделенного элемента на новую позицию курсор мыши подводится в область того элемента, перед которым требуется расположить выделенный элемент. Затем нажимается правая клавиша мыши. Если правая клавиша нажимается вне областей элементов массива, то выделенный элемент переносится в конец массива.

Текстформульный редактор использует для временного хранения в оперативной памяти текстовых фрагментов так называемый массив текстов (см. выше базисные операторы ЛОСа "видео(начало)", "видео(конец)", "видео(образ ...)", "видео(прообраз ...), "видео(сброс ...)", "видео(заменавхождения ...)", "видео(см ...)", "видео (посылка ...)", "видео(вычеркивание ...)"). Этот массив позволяет сохранять до 56000 букв; текстовые фрагменты в него заносятся из буфера текстов и возвращаются тоже в буфер текстов.

Как уже отмечалось выше, входными данными оператора "текстформ(a b)"являются текстформульная структура a и набор информационных элементов b. Текстформульная структура представляет собой набор (S) длины 1, состоящий из набора S элементов, определяющих последовательно расположенные текстформульные фрагменты. Типы этих элементов таковы:

1) (слово A_1 A_2) - A_1 и A_2 суть символьные номера начала и конца текстового фрагмента в массиве текстов;

2) (терм A) - A есть терм;

3) (позиция A) - A есть указатель форматирования. Если $A = 1$, то следующий фрагмент прорисовывается с новой строки. Если $A = 2$, то выполняется пропуск строки. Если $A = (3B)$, то перед следующим фрагментом создается пробел в B пикселей без перехода на следующую строку; B - символьное число.

4) (геомредактор A_1 A_2 A_3) - A_1 есть описание чертежа в формате оператора "геомредактор"; A_2, A_3 - верхняя и нижняя строки рамки чертежа.

Информационные элементы набора b, определяющего режим работы текстформульного редактора, бывают следующих типов:

1) (начало A) - A есть символьный номер первой строки для прорисовки (по умолчанию 0).

2) (изменение A) - A есть индикатор изменений текстформульной структуры: $A = 1$ - были изменения, $A = 0$ - не было изменений (анализируется после выхода из оператора "текстформ"для принятия решения о сохранении изменений в файлах).

3) (высотаполосы A) - прорисовки не происходит; A становится равно высоте полосы, необходимой для прорисовки.

4) "выход" - выход из процедуры сразу после прорисовки.

5) (конец A) - A есть нижняя строка полосы прорисовки (для аналиаз после выхода из оператора).

6) (цветпункта A) - A есть цвет символов для прорисовки.

7) "решение выход из оператора при нажатии любой клавиши, кроме клавиш прокрутки.

8) (видео A_1 A_2 A_3) - указатель на экранную структуру A_1 (см. ниже), символьный номер A_2 первой свободной строки для прорисовки и номер A_3 полосы экранной

структуры A_1, с которой начинается прорисовка. Экранная структура определяет размещение на экране фактически прорисовываемой части текстформульного массива. Она создается автоматически самим текстформульным редактором, однако ее можно сохранить и использовать, для ускорения повторной прорисовки, при следующем обращении к текстформульному редактору. Экранная структура представляет собой набор $(P_1 \dots P_n)$ наборов P_i, представляющих собой экранные полосы. Каждое P_i имеет вид $(B_1\ B_2\ B_3\ B_4)$, где B_1 - символьный номер первой строки полосы, B_2 - высота полосы в пикселях (символьное число), B_3 - символьный номер несущей строки полосы (на ней размещаются буквы, если из-за выступающих вверх частей формул, расположенных на той же строке, приходится смещать текстовую часть вниз); B_4 - набор $(C_1 \dots C_k)$ элементов полосы. Каждое C_i имеет один из следующих видов:

а) Текстовый элемент - набор (слово $D_1\ D_2\ D_3\ D_4\ D_5$), где D_1, D_2 - ссылки на начало и конец в массиве текстов, D_3,D_4 - символьные номера левого и правого столбцов элемента; D_5 - вхождение левого края соответствующего элемента текстформульной структуры (к одному и тому же элементу текстформульной структуры может относиться несколько информационных элементов экранной структуры, размещенных на соседних строках).

б) Скобочный элемент - набор (терм $D_1\ D_2\ D_3\ D_4$), где D_1 - терм, прорисовываемый в скобочной записи; D_2, D_3 - левый и правый столбцы элемента; D_4 - вхождение левого края соответствующего информационного элемента текстформульной структуры.

в) Формульный элемент - набор (формредактор $D_1\ D_2\ D_3\ D_4\ D_5$), где D_1 - терм, прорисовываемый формульным редактором; D_2 - дерево указателей формульного редактора для D_1; D_3, D_4 - левый и правый столбцы элемента; D_5 - вхождение левого края соответствующего элемента текстформульной структуры.

г) Элемент форматирования - набор (пусто $D_1\ D_2\ D_3$), где D_1, D_2 - левый и правый столбцы пустой зоны полосы; D_3 - вхождение левого края соответствующего элемента текстформульной структуры.

д) Геометрический элемент - набор (геомредактор $D_1\ D_2\ D_3\ D_4$), где D_1 - структура данных описания чертежа; D_2, D_3 - левый и правый столбцы чертежа; D_4 - вхождение левого края соответствующего элемента текстформульной структуры.

В информационном блоке текстформульная структура сохраняется в виде текстформульной ветви - цепочки указателей - списков, переходы между которыми осуществляются по метке "продолжение". Каждый такой указатель - список хранит несколько последовательных элементов текстформульной структуры. Переходы от него к логическим либо текстовым терминалам с этими элементами происходят по меткам "1","2", ... Если логический терминал хранит единственный терм, не начинающийся с логического символа "фикс", то этот терм определяет элемент (терм ...) текстформульной структуры; если он хранит терм "фикс(...)", то операнды данного терма образуют установку на форматирование текста - элемент (позиция ...). Если логический терминал хранит более одного терма, то он содержит описание чертежа.

Оператор "записьтекста($a\ b\ c$)"позволяет записывать текстформульную структуру c в виде текстформульной ветви активного информационного блока, к которой от указателя -списка a имеется переход по метке b. Если из a по метке b ранее выходила другая ветвь, то она удаляется. Оператор "чтениетекста($a\ b\ c$)"получает ссылку на указатель-список a активного информационного блока и метку b, по которой от него имеется переход к текстформульной ветви. Переменной c присваивается соответствующая текстформульная структура. Используемые к этому моменту фрагменты

массива текстов сохраняются.

6.5.7 Оглавления

Оглавления чрезвычайно важны в интерфейсе логической системы. Задачник системы, база приемов, база теорем, справочник по системе, комментарии к ЛОС-программам основных блоков, различные вспомогательные справочники - организованы с помощью оглавлений. Фактически оглавления играют не просто роль древовидного классификатора для быстрого поиска объектов того или иного типа, но и роль функционального интерфейса, позволяющего инициировать различные процедуры.

Структура данных оглавления представляет собой древовидную систему указателей-списков в каком-либо информационном блоке. Эти указатели-списки далее называем меню оглавления. Каждое меню имеет метки следующих типов:

1) Номера пунктов меню - логические символы $1, 2, \ldots$. По такой метке i имеется переход к паре объектов. Первый элемент данной пары - текстовый терминал, определяющий текст i - го пункта меню. Второй элемент пары - либо соответствующее i -му пункту подменю оглавления, либо логический терминал. В последнем случае данный логический терминал может содержать косвенную ссылку на подменю оглавления - терм "логсимвол(A_1A_2)", где A_1 - логический символ, по которому требуется осуществить переход в корневом каталоге текущего информационного блока к некоторому указателю - списку S; A_2 - логический символ, по которому нужно перейти далее из S к корню искомого подменю оглавления. Если логический терминал не содержит терма вида "логсимвол(...)", то он соответствует концевому пункту оглавления. Косвенные ссылки нужны в очень больших оглавлениях, которые целиком не умещаются в одном файле информационного блока. Заметим, что в отсутствии косвенных ссылок все переходы внутри оглавления (как переходы между указателями - списками) происходят внутри одного такого файла. Чтобы можно было использовать другой файл того же информационного блока, нужно выбрать логический символ A_1, отличный от того, к ветви которого относится оглавление (все объекты ветви одного и того же логического символа размещаются в одном файле), и организовать косвенную ссылку на ветвь оглавления, выносимую таким образом в отдельный файл. Обычно в качестве символа A_2 при этом выбирается логический символ "оглавление". Для "разрезания"оглавления с применением косвенных ссылок в интерфейсе оглавлений предусмотрена специальная операция (см. ниже).

2) По метке "позиция" - переход к логическому терминалу, хранящему терм "число(i j)". Здесь i - номер пункта оглавления, выбранного в последний раз при работе с данным меню (либо 0); j - номер первого пункта, с которого будет происходить выдача меню на экран. Информация, сохраняющаяся в терминалах "позиция", позволяет прослеживать в оглавлении "текущий путь"от корня, и при повторном входе в оглавление сразу попадать в ту его точку, из которой имел место последний выход. Изменяя с помощью специального оператора текущий путь в оглавлении, можно обеспечивать автоматические переходы к различным его пунктам.

3) По метке "замечание" - переход к логическому терминалу, хранящему дополнительную информацию о ветви оглавления. Эти терминалы фактически относятся к структуре данных использующей оглавление внешней процедуры. Например, в случае базы теорем они содержат информацию о разделах, к которым относятся ветви оглавления, позволяющую логической системе самостоятельно ориентироваться в оглавлении, изначально предназначенном для работы с ним человека - оператора.

4) По метке "примечание" - переход к текстовому терминалу, хранящему сопроводительный текст для ветви оглавления.

Корни оглавлений размещаются в информационных блоках с использованием следующего соглашения. Берется указатель-список, достижимый из корневого каталога по метке "оглавление". Из этого указателя по меткам - логическим символам, рассматриваемым далее как названия типов оглавлений, и выполняются переходы к корням оглавлений. Типы используемых в системе оглавлений и дополнительные их особенности будем указывать в описании тех блоков, которые используют эти оглавления (кроме того, см. перечень типов оглавлений в справочной информации для логического символа "вид").

Для работы с оглавлениями используется оператор "оглавление(a b)". У него a - ссылка на корневой указатель - список оглавления. Оператор реализует интерфейс выбора концевого пункта оглавления и изменения самого оглавления. При обращении к нему сразу осуществляется переход к последнему ранее рассматривавшемуся при работе с данным оглавлением пункту (концевому либо промежуточному). Выходной переменной b присваивается пара (ссылка на текстовый терминал выбранного концевого пункта - ссылка на его логический терминал). Если концевой пункт выбран не был (выход из оглавления по "Esc"и т.п.), то оператор ложен.

Хотя вкратце интерфейс оглавлений уже был описан выше (применительно к оглавлению задачника), приведем эдесь более полное его описание.

При прорисовке меню оглавления пункты нумеруются последовательными натуральными числами. Если пункт является концевым, то после его номера идет точка. Если же пункт оглавления сам является меню, то после его номера идет правая скобка.

Текущий пункт меню оглавления выделен синим цветом. Для смены этого пункта используются клавиши "курсор вверх"и "курсор вниз". В случае большого меню, не умещающегося целиком на экране, можно использовать также клавиши "PageUp"и "PageDown".

Для входа в подменю либо в интерфейс, обслуживающий выбранный концевой пункт оглавления, нажимается клавиша "курсор вправо"либо (что менее предпочтительно, так как более мобильным является перемещение с помощью четырех клавиш курсора) "Enter".

Для возвращения в надменю нажимается клавиша "курсор влево" либо "Esc". Нажатие "Esc" в корневом меню выводит из оглавления во внешнюю процедуру, обратившуюся к его просмотру; нажатие "курсор влево" в корневом меню игнорируется.

Для одношагового выхода из оглавления (в главное меню либо в другую внешнюю процедуру интерфейса), минуя все промежуточные надменю, используется клавиша "End".

Для одношагового перехода к нижнему из прорисованных на экране пункту меню используется клавиша "Ctr-курсор вниз"(одношаговый переход вверх не предусмотрен).

При нажатии клавиши для одной из цифр $0, 1, \ldots, 9$ происходит переход к первому из прорисованных на экране пунктов меню, имеющему данную последнюю цифру.

Возможно перемещение по оглавлению с помощью мыши. Если подвести курсор мыши к нужному пункту меню и нажать левую клавишу мыши, то происходит переход к подменю (либо концевому интерфейсу) этого пункта. При нажатии правой клавиши мыши происходит возвращение в надменю.

Оглавление можно изменять непосредственно в процессе работы с ним. Если нуж-

но создать новый пункт оглавления для подменю, то нажимается клавиша "м"либо "М". В первом случае пункт будет создан в конце меню, и для него необходимо сначала расчистить место прокруткой оглавления вверх. Во втором случае новый пункт будет создан непосредственно перед текущим (выделенным синим цветом) пунктом оглавления, и окно текстового редактора, в котором следует набирать текст для нового пункта, разместится непосредственно поверх текста текущего и следующих за ним пунктов. После нажатия указанной клавиши появляется окно текстового редактора, с помощью которого вводится текст пункта. По завершении ввода (клавиша "Enter") прорисовывается модифицированное меню. Если во время ввода нажать "Esc", то будет восстановлен исходный вид меню.

Если нужно создать новый концевой пункт оглавления, то нажимается клавиша "к"либо "К". В первом случае пунтк будет создан в конце меню, во втором случае - непосредственно перед текущим пунктом оглавления. В остальном действия те же, что и при вводе нового пункта для подменю.

Для перенесения группы пунктов одного или нескольких меню на другое место используется буфер оглавления. Для отбора текущего пункта в этот буфер нажимается клавиша "б". В результате цвет пункта становится красным (если переместить указатель текущего пункта в другое место, то отобранный в буфер пункт сохраняет малиновую окраску). Чтобы исключить из буфера ранее занесенный в него текущий пункт, снова нажимается клавиша "б".

Чтобы перенести отобранные в буфер пункты на новое место, либо нажимается клавиша "и" - тогда отобранные пункты будут исключены со своих исходных позиций и занесены в конец текущего меню в том же порядке, в каком они заносились в буфер, либо нажимается клавиша "И" - тогда отобранные пункты после исключения с исходных позиций регистрируются непосредственно перед текущим пунктом текущего меню.

В особых случаях перенесение пункта оглавления на другое место оказывается невозможным. Это происходит, если при разрезании оглавления на фрагменты, отнесенные к различным логическим символам, старая и новая позиции пункта оказались относящимися к различным файлам информационного блока.

Для удаления текущего пункта меню следует нажать клавишу "Ctr-Del". Если этот пункт представлял собой подменю, то удаление его произойдет только в случае пустого подменю.

Для изменения текста текущего пункта следует нажать клавишу "р"(кир.). Тогда цвет текста становится черным и возникает курсор текстового редактора. По окончании редактирования (клавиша "Enter"в текстовом редакторе) текст пункта модифицируется. Если в текстовом редакторе нажать "Esc", то восстановится исходный вид пункта.

В процессе развития оглавления его ветвь может оказаться чрезмерно большой для сохранения ее в одном файле информационного блока (размеры таких файлов ограничены, ввиду особенностей двоичного кодирования адресов в используемой версии интерпретатора ЛОСа, 512 килобайтами). Тогда к данной ветви применяется операция разрезания оглавления. Выбирается некоторый логический символ "A по возможности, не чрезмерно загруженный в рассматриваемом информационном блоке, и данная ветвь закрепляется за этим символом. Фактически выполняются следующие действия: выбирается текущий пункт, определяющий переход к отрезаемой ветви; нажимается "Ctr-F6"; в появляющемся диалоговом окне (после слов "Косвенная ссылка:") текстовым редактором набирается терм "логсимвол(A оглавление)", и нажимается "Enter". Не рекомендуется применять данную операцию без крайней

необходимости.

Кроме перечисленных выше, процедура "оглавление(...)"реализует ряд других операций, тесно связанных с внешними процедурами, обращающимися к оглавлениям. Эти операции будут указаны далее в разделах, посвященным таким процедурам.

Приведем ряд операторов, часто используемых при работе с оглавлениями. Использование их предполагает, что содержащий оглавление информационный блок является активным.

Чтобы получить последовательность ссылок на пункты текущего пути оглавления, используется оператор "узлыоглавления(a b)". У него a - тип оглавления; выходной переменной b присваивается набор ссылок. Каждая такая ссылка - пара (ссылка на текстовый терминал пункта - ссылка на сам пункт,т.е. на указатель либо терминал). Начало набора b соответствует корню оглавления, причем вместо ссылки на текстовый терминал (отсутствующий у корня) берется 0.

Если нужно получить последовательность ссылок на пункты произвольного (не обязательно текущего) пути оглавления, применяется оператор "пунктыоглавления(a b c)". У него a - тип оглавления; b - набор меток перехода (символьных номеров пунктов) вдоль требуемого пути. Выходной переменной c присваивается набор ссылок (текстовый терминал пункта - пункт), аналогичный выходному набору оператора "узлыоглавления". Единственное отличие - в наборе c ссылка на корень оглавления отброшена. Иногда нужно получить ссылки не на все пункты пути, а только на последний пункт. Тогда можно применять оператор "пунктоглавления(a b c)". Входные данные у него те же, что у оператора "пунктыоглавления", но выходной переменной c присваивается пара ссылок (текстовый терминал последнего пункта пути - последний пункт).

Чтобы сделать текущим путем оглавления путь, определяемый заданной последовательностью b символьных номеров пунктов, используется оператор "путьвоглавлении(a b c)". Здесь a - тип олгавления. Выходной переменной c присваивается пара ссылок (текстовый терминал концевого пункта пути b - концевой пункт этого пути). Чтобы получить набор символьный номеров пунктов, определяющих текущий путь в оглавлении, применяется оператор "цепьвоглавлении(a b)". У него a - тип оглавления; выходной переменной b присваивается искомый набор.

Чтобы перечислить ссылки на все пункты заданного меню оглавления, используется оператор "подпункт(a b c d)". У него a - ссылка на представляющий меню указатель - список. Выходной переменной b присваивается метка перехода к очередному пункту меню; c - ссылка на текстовый терминал этого пункта; d - ссылка на сам пункт.

Оператор "смоглавление(a b)"перечисляет ссылки на все пункты оглавления (концевые и неконцевые), достижимые из заданного пункта. a - ссылка на исходный пункт (указатель либо логический терминал). b - пара (ссылка на текстовый терминал пункта - пункт) для текущего перечисляемого пункта. Исходный пункт a из перечисления исключается.

Чтобы просмотреть все концевые пункты оглавления, расположенные в некоторой его ветви, может быть полезен оператор "следтерминал(a b c d e)". Он осуществляет переход к очередному концевому пункту внутри заданной ветви. a - тип оглавления; b - путь (набор символьных номеров выбираемых пунктов) от корня оглавления к корню его ветви; c - путь от корня оглавления к его логическому терминалу, расположенному внутри ветви. Если удается найти логический терминал, расположенный в ветви после заданного, то переменной d присваивается путь к этому терминалу от корня оглавления; переменной e - пара (ссылка на текстовый терминал, сопровожда-

ющий найденный логический терминал - ссылка на логический терминал). Иначе оператор ложен.

Оператор "ветвьоглавления(a b c d)"перечисляет все концевые пункты оглавления, достижимые из некоторого его меню, причем через пункты этого меню с номерами, не меньшими заданного. У него a - ссылка на указатель- список, представляющий данное меню; b - путь к меню от корня оглавления; c - символьный номер того пункта меню, с которого начинается перечисление. Выходная переменная d перечисляет тройки (содержимое логического терминала очередного концевого пункта - пара (ссылка на текстовый терминал, сопровождающий логический терминал - ссылка на логический терминал) - путь к концевому пункту от корня оглавления). Заметим, что этот оператор обслуживает интерфейс оглавлений для коррекции его структур данных при перестановках, вставках и удалениях пунктов; поэтому, кроме указанных выше, им выполняются также некоторые дополнительные действия (обработка комментария (оглавление A) к посылкам исходной задачи - буфера оглавления, который нужно корректировать при указанных выше операциях).

6.5.8 Информационные блоки

Если информационный блок имеет корневой указатель-каталог, то обычно он организуется по следующей схеме. По меткам - логическим символам A из его корневого каталога происходят переходы к указателям-спискам. Ветвь такого указателя-списка называется статьей логического символа A, а сам указатель - корнем статьи. С ним связывается некоторая древовидная система указателей, переходы в которой осуществляются по меткам - цифрам. Концевые вершины данной древовидной системы указателей называются узлами статьи символа A; последовательность цифр на пути от корня к узлу образует десятичную запись номера узла. Сохраняемые в информационном блоке объекты (приемы, теоремы и т.п.) распределяются между узлами статей логических символов, и ссылка на узел - пара (логический символ - номер узла) так или иначе включается в ссылку на этот объект.

Последовательность цифр, образующая номер узла, может начинаться с одного или нескольких нулей, но должна иметь хотя бы один ненулевой элемент. Из корня статьи логического символа по метке "длина"имеется переход к логическому терминалу, хранящему длину этой последовательности (одну и ту же для всех узлов). Номера узлов, представленные в дереве, не обязательно образуют сплошной отрезок натуральных чисел; между ними допустимы разрывы. Для ускоренного поиска неиспользуемого номера при вводе нового узла в дереве предусмотрена специальная разметка. Если вершина A дерева номеров узлов относится не к последнему и не к предпоследнему ярусам этого дерева, причем некоторые из ветвей этой вершины имеют неиспользуемые номера, то логический терминал, достижимый из A по метке "новыйузел", хранит все корневые метки указанных ветвей.

Для ввода нового узла статьи логического символа S служит оператор "новыйузел(a b c)". У него a - ссылка на корень статьи логического символа S. Выходной переменной b присваивается ссылка на новый узел, а переменной c - номер этого узла (десятичное число). Для удаления узла применяется оператор "удалениеузла(a b)". Здесь a - ссылка на корень статьи логического символа; b - номер удаляемого узла. Для получения ссылки c на узел статьи логического символа a, имеющий номер (десятичному числу) b, служит оператор "адресузла(a b c)".

Оператор "узелстатьи(a b c)"перечисляет все узлы статьи логического символа a. Переменной b при этом присваивается ссылка на узел, c - номер этого узла. Чтобы

просматривать узлы статьи некоторого логического символа, можно использовать также оператор "следузел(a b c d)", который позволяет находить первый узел, номер которого больше заданного числа b. Здесь a - корень статьи рассматриваемого логического символа. Переменной c присваивается ссылка на найденный узел, переменной d - номер этого узла. Если узла с номером, большим b, не существует, оператор ложен.

Если нужно скопировать целую ветвь активного информационного блока, не выходя за рамки этого блока (но, быть может, в другом его файле), то применяется оператор "новаяветвь(a b)". У него a - ссылка на указатель-список, являющийся корнем копируемой ветви; b - указатель-список, начиная с которого будет добавляться копия. Если в b уже имелись переходы по тем же меткам, что в a, то происходит добавление ссылок на подключаемые копии объектов из a. Предполагается, что в копируемой ветви имеются только указатели-списки, логические терминалы и текстовые терминалы без указателей цвета. При копировании кратные ссылки на терминалы не воспроизводятся, и по каждой такой ссылке создается новая копия терминала.

6.5.9 Диалоговые блоки

Для организации специализированных диалогов, в том числе с использованием логических структур данных, можно создавать диалоговые блоки. Эти блоки служат для отображения различной статистической информации, выбора параметров, прорисовки и редактирования термов и логических символов. Ввиду специфических функциональных особенностей, они реализованы внутренними средствами логической системы и не используют стандартных диалоговых средств. Диалоговый блок обычно занимает весь экран и состоит из группы прямоугольников, называемых элементами блока диалога. Каждый элемент блока диалога сопровождается группой информационных элементов, уточняющих списоб его использования. Активизация элемента диалога предпринимается нажатием левой клавиши мыши в его области либо нажатием клавиши клавиатуры, код которой является меткой элемента (обычно название такой клавиши указывается в прямоугольнике элемента либо в примыкающих к нему прямоугольниках). Предусмотрен простой интерфейс, позволяющий создавать и редактировать блоки диалога. При обращении к диалоговому блоку создается структура данных диалога - набор информационных элементов для заполнения его окон при первоначальной прорисовке и для сохранения введенной в процессе диалога информации.

Прежде всего, перечислим типы информационных элементов, характеризующих использование элемента блока диалога (каждый такой элемент - логический символ либо терм):

1) "метка(A)" логический символ A является меткой элемента.

2) "См(A)" прямоугольная рамка элемента прорисовывается в цвете A. В случае белого цвета (A = "шестнадцать") рамка не видна.

3) "формблок" при активизации элемента инициируется набор терма формульным редактором. По окончании набора в структуре данных диалога находится информационный элемент "формблок A B", где A - метка элемента диалога, у которого B заменяется на набранный терм.

4) "текстредактор" при активизации элемента инициируется вход в текстовый редактор. Начало B используемого отрезка буфера текстов при этом берется из информационного элемента "текстредактор A B"структуры данных диалога; A - метка элемента диалога.

5) "терм" при активизации элемента инициируется набор терма текстовым редактором. По окончании набора в структуре данных диалога находится информационный элемент "терм A B", где A - метка элемента диалога, у которого B заменяется на набранный терм.

6) "логсимвол" при активизации элемента инициируется набор логического символа текстовым редактором. По окончании набора в структуре данных диалога находится информационный элемент "логсимвол A B", где A - метка элемента диалога, у которого B заменяется на набранный логический символ.

7) "новыйсимвол" при активизации элемента инициируется набор логического символа текстовым редактором. Если символа с таким названием не было, то он вводится. По окончании набора в структуре данных диалога находится информационный элемент "новыйсимвол A B", где A - метка элемента диалога, у которого B заменяется на набранный логический символ.

8) "серия" при активизации элемента инициируется набор группы термов текстовым редактором. По окончании набора в структуре данных диалога находится информационный элемент "серия A B", где A - метка элемента диалога, у которого B заменяется на список набранных термов.

Шаблоны используемых в системе диалоговых блоков хранятся в 5-м информационном блоке. Для хранения отдельного шаблона используется узел A статьи какого-либо логического символа, так что пара (логический символ - номер узла) является ссылкой на этот шаблон. Из узла A по меткам "команды"и "нормистина" (первоначально - логический символ "блокдиалога", который впоследствии был переименован в название вспомогательной процедуры) имеется переход к указателю - списку B, из которого по меткам $1, 2, \ldots$ - переходы к указателям-спискам $C_1, C_2, \ldots$, представляющим элементы блока диалога. Из C_i по метке "область переход к логическому терминалу, хранящему четверку логических символов D_1, D_2, D_3, D_4. D_1, D_2 суть символьные номера столбца и строки верхнего левого угла прямоугольника, занимаемого на экране элементом; D_3, D_4 - символьные номера столбца и строки правого нижнего угла. По метке "слово"(может отсутствовать) из C_i - переход к текстовому терминалу, содержимое которого выводится в области элемента. По метке "нормистина"из C_i - переход к логическому терминалу, содержащему информационные элементы, характеризующие использование элемента диалога (типы их приведены выше).

В том же 5-м информационном блоке содержится оглавление шаблонов диалоговых блоков; переход к его корню от корневого каталога 5-го информационного блока осуществляется по меткам "оглавление", "нормистина"). Концевые логические терминалы данного оглавления хранят термы "прием(A_1A_2 нормистина)", где A_2 - номер узла статьи символа A_1, хранящего описание шаблона.

Для инициализации диалога по уже имеющемуся шаблону диалогового блока используется оператор "началодиалога(a b c)". У него b - номер узла статьи логического символа a, хранящего описание шаблона блока. Оператор осуществляет прорисовку диалогового блока с незаполненными (кроме текстов, введенных непосредственно при создании блока) элементами и возвращает структуру данных диалога c, используемую, в частности, как накопитель результатов диалога. Приведем перечень типов элементов в этой структуре данных:

1) (формблок A B) - B есть терм, прорисованный формульным редактором внутри элемента диалога с меткой A.

2) (терм A B) - B есть терм, прорисованный текстовым редактором внутри элемента с меткой A.

3) (логсимвол A B) - B есть логический символ, прорисованный внутри элемента с меткой A.

4) (текстредактор A B) - B есть исходная позиция в буфере текстов, начиная с которой размещается текст, вводимый внутри элемента с меткой A.

5) (прямоугольник A B) - B есть четверка символьных номеров, определяющая границы прямоугольника для элемента диалога с меткой A.

6) (новыйсимвол A B) - B есть новый логический символ, введенный в элементе с меткой A.

7) (серия A B) - B есть набор термов, введенных с помощью текстового редактора в элементе диалога с меткой A.

В тех случаях, когда параметр B изменяется при диалоге, он изначально инициализируется какой-либо переменной.

После того, как оператор "началодиалога"прорисовал бланк блока диалога, происходит заполнение необходимым содержимым прямоугольников его элементов. Для этого служит оператор "рис(a b c)". У него a - структура данных диалога; b - метка элемента диалога. Осуществляется прорисовка внутри прямоугольника этого элемента информации, определяемой указателем прорисовки c. Используются указатели прорисовки следующих типов:

1) (формблок A) - прорисовка терма A формульным редактором;

2) (терм A) - прорисовка терма A в скобочной записи;

3) (логсимвол A) - прорисовка логического символа A;

4) (текстредактор A B C) - прорисовка содержимого буфера текстов начиная с позиции A до позиции B. C - символьный номер строки, с которой начинается прорисовка (относительно верхней части прямоугольника элемента диалога);

5) "пустоеслово расчистка прямоугольника элемента;

6) (серия A) - прорисовка набора термов A в скобочной записи.

Для создания нового шаблона блока диалога служит оператор "новыйдиалог (a b)". Перед обращением к нему должен быть создан узел статьи некоторого логического символа a в 5-м информационном блоке, в котором предполагается сохранить шаблон, и из этого узла должен быть создан переход по метке "команды"к новому указателю - списку. b - номер данного узла. Оператор реализует интерфейс редактирования шаблона; обращение к нему возможно из редактора программ ЛОСа (см. следующий раздел), так что выбор a, b и создание узла выполняетя автоматически.

Перечислим операции, применяемые при редактировании шаблона.

Для создания нового элемента диалога нажимается "c"(кир.), либо курсор мыши подводится к пункту меню "Создание"и нажимается левая клавиша мыши. Затем левая клавиша мыши нажимается сначала в позиции левого верхнего угла прямоугольника, затем - в позиции правого нижнего угла.

Для удаления элемента диалога сначала активизируется пункт меню "Удаление", и затем левая клавиша мыши нажимается в области удаляемого элемента.

Для ввода текста внутри прямоугольника элемента активизируется пункт меню "Текст", и левая клавиша мыши нажимается в области элемента. Тогда появляется курсор текстового редактора, позволяющий ввести текст. По завершении редактирования ("Enter") текст сохраняется, иначе, по "Esc" - редактирование отменяется.

Для редактирования списка сопровождающих элемент диалога информационных элементов, уточняющих его использование, выбирается пункт меню "Поведение", и левая клавиша мыши нажимается в поле элемента. Тогда шаблон диалога исчезает, а в верхнем левом угла экрана появляется курсор текстового редактора (если уже

имелись сопровождающие информационные элементы, то они прорисовываются). По завершении редактирование изображение шаблона восстанавливается.

Чтобы получить ссылку a, b (логический символ - номер узла) на редактируемый шаблон, выбирается пункт меню "Информация" - тогда в верхнем левом углу прорисовываются a и b. По нажатии любой клавиши далее восстанавливается изображение шаблона.

Чтобы передвинуть рамку прямоугольника элемента, левая клавиша мыши нажимается на одной из его сторон, и далее удерживанием клавиши курсора можно добиться медленного перемещения этой стороны (для вертикальных сторон нажимаются клавиши "курсор вправо - влево", для горизонтальных - "курсор вверх- вниз"). Пока выбранная сторона не будет сдвинута хотя бы на один пиксел, прочие команды редактором игнорируются.

Для перемещения всего элемента нужно нажать левую клавишу мыши в области этого элемента и выбрать пункт меню "Сдвиг", после чего использовать клавиши курсора.

Выход из редактирования - по "Esc"; при этом результаты редактирования сохраняются.

6.5.10 Разное

Если нужно сохранить в текстовом терминале последовательность букв, определенных кодирующими их (т.е. являющихся кодами соответствующих клавиш) логическими символами, то используется оператор "текстовыйтерминал(a b c d)". У него a - последовательность кодирующих логических символов; b - ссылка на указатель активного информационного блока, в котором по метке c регистрируется текстовый терминал. Переменной d присваивается ссылка на созданный терминал.

Справочная информация о логических символах хранится в 3-м информационном блоке. Из корневого каталога этого блока по метке - логическому символу S происходит переход к указателю, из которого по метке "оператор переход к группе текстовых терминалов, образующих справочную информацию для S. Каждый терминал этой группы соответствует одной странице; упорядочение страниц - от конца набора терминалов к началу (так как каждый новый объект по заданной метке добавляется в начало списка достижимых по этой метке объектов). Некоторые справочные тексты общего характера размещены в 1-м информационном блоке. Для просмотра и редактирования справочных текстов перечисленных типов используется оператор "помощь(a b)". У него a - либо логический символ 0, либо логический символ 1, либо ссылка на указатель-список 1-го информационного блока; b - логический символ. Если $a = 0$, то при обращении к оператору возникает окно диалога - под текстом "Логический символ:"появляется курсор текстового редактора, и далее вводится тот логический символ, для которого нужна справка. Если $a = 1$, то просматривается и редактируется информация о логическом символе b. В остальных случаях просматриваются и редактируются текстовые терминалы, достижимые из указателя a по метке b. После обращения к оператору активным остается тот информационный блок (1 либо 3), из которого извлекалась справочная информация.

Интерфейс просмотра справочной информации состоит из следующих операций. Для смены страницы служат клавиши "Page Up" "Page Down"либо нажатие левой клавиши мыши на стрелках в меню. Чтобы внести изменения в текст страницы, достаточно нажать "Enter тогда появляется курсор текстового редактора, и предпринимается редактирование текста (по завершении редактирования через

"Enter"изменения сохраняются, по завершении редактирования через "Esc" - сохраняется первоначальный текст). Для удаления текущей страницы нажимается "Ctr-Del". Если нужно ввести новую страницу непосредственно после последней, нажимается "Ctr-Enter". Нажатие любых других клавиш (в том числе клавиш курсора) приводит к выходу из просмотра информации.

Общий справочник по логической системе размещается в 7-м информационном блоке. Он организован с помощью оглавления, корень которого достижим из корневого каталога по меткам "оглавление", "текстформ". Каждый концевой логический терминал данного оглавления содержит терм "прием(A_1A_2)", ссылающийся на узел статьи логического символа A_1, имеющий номер A_2. По метке "текстформ"из этого узла выходит текстформульная ветвь, соответствующая концевому пункту оглавления. Для просмотра и редактирования общего справочника по решателю служит оператор "Помощь(a)". Здесь a указывает на режим использования массива текстов: если $a = 0$, то при выходе из оператора этот массив полностью расчищается. Интерфейс работы со справочником, по существу, распадается на интерфейс оглавлений и интерфейс текстформульного редактора.

Для работы с графиками функций одной переменной используется оператор "график(a b)". У него a - набор выражений; b - набор информационных элементов, испоьзуемых для вычисления значений набора a. Реализуется интерфейс графического анализа выражений набора a - выбор варьируемого параметра и границ его изменения, с построением на общем чертеже кривых для выражений данного набора. Набор b называем блоком графика; он имеет информационные элементы следующих типов:

1) (выч i A_1 A_2) - A_1 есть микропрограмма для оператора "Выч", позволяющая вычислять значение i - го выражения набора a; A_2 - набор вспомогательных констант для этой микропрограммы. Нумерация элементов набора a начинается с 1.

2) (переменная x) - x есть варьируемая переменная, для которой строится график.

3) (от t_1) - t_1 есть минимальное значение варьируемой переменной.

4) (до t_2) - t_2 есть максимальное значение варьируемой переменной.

5) (значение y r) - r есть численное значение параметра y, отличного от варьируемой переменной.

6) (отрезок A_1 A_2) - A_1 есть нижняя граница устанавливаемого вручную диапазона значений для отображения на экране; A_2 - верхняя граница этого диапазона. A_1, A_2 - десятичные числа.

7) (левыйкрай A) - A есть символьный номер столбца, по которому проходит вертикальная шкала.

8) (конец A) - A есть символьный номер строки для прорисовки горизонтальной шкалы.

9) (прямоугольник A_1 A_2 A_3 A_4) - указатель на верхнюю левую (A_1, A_2) и правую нижнюю (A_3, A_4) вершины прямоугольника для выделенного фрагмента графика.

10) (откат B) - B есть набор четверок $(A_1A_2A_3A_4)$, указывающих границы прямоугольников просмотра графика, предшествовавшие выбору текущего прямоугольника. Начало набора B - последний из таких надпрямоугольников (наименьший).

11) (точка A_1 A_2) - A_1 есть символьный номер столбца для выделенной точки графика; A_2 - символьный номер кривой графика.

Интерфейс работы с графиками был описан в разделе "Диалоговые задачи" (заметим, что пока в этом интерфейсе используются лишь возможности оператора "график(...)", связанные с рассмотрением единственного выражения).

Глава 7

Редактор программ ЛОСа

7.1 Названия логических символов

При программировании на ЛОСе необходимо знать названия логических символов, используемых в логическом языке системы либо являющихся заголовками операторов ЛОСа. Иногда приходится вводить названия новых логических символов либо изменять названия уже введенных символов. Для просмотра имеющихся на текущий момент символов и простейших операций с ними предусмотрены следующие возможности.

7.1.1 Просмотр названий логических символов

Чтобы просмотреть все используемые в системе названия логических символов, следует в главном меню системы выбрать нажатием левой кнопки мыши пункт "Логические символы"либо нажать клавишу "с"(кир.). Тогда на экране появляются расположенные в два столбца названия первых 46 логических символов (первая страница словаря). Для перелистывания страниц используются клавиши "PgDown"и "PgUp". Перед названием логического символа помещен его номер. На текущей странице синим цветом выделяется текущий логический символ. Смена этого символа осуществляется (только в рамках страницы) клавишами "курсор вверх" - "курсор вниз". Кроме того, клавиши "курсор вправо" курсор влево позволяют менять столбец.

Для просмотра справочной информации, связанной с текущим логическим символом, следует нажать клавишу "и тогда появляется первая страница такой информации. Перелистывание этих страниц (если их несколько) - при помощи клавиш "PageDown", "PageUp". Если нужно внести дополнения либо изменения в справочную страницу, то нажимается "Enter" - появляется курсор текстового редактора. По завершении редактирования нажимается "Enter", что приводит к немедленному изменению страницы. Если во время редактирования нажать "Esc", то изменение страницы отменяется. Новая страница (после последней) вводится нажатием клавиши "Ctr - Enter". Для удаления текущей страницы используется "Ctr - Del". Возвращение в списки логических символов - по нажатию любой не используемой для специальных целей клавише (например, по клавише "пробел"либо "Esc").

Если после названия логического символа помещен знак "?", то отсутствует программа этого символа. Если после него помещен знак "&", то отсутствует связанная с символом справочная информация. Наконец, если вместо названия символа помещен знак "!", то само название символа вообще не введено (логический символ с соответствующим номером все же существует, но фактически не используется).

7.1.2 Изменение словаря и получение дополнительной информации о символе

Операции по изменению словаря и получению информации, связанной с номером символа, вынесены в пункт "Ресурсы и установки" главного меню системы. После выбора этого пункта (левой кнопкой мыши либо нажатием клавиши "р") появляется меню операций с ресурсами и установками. Левая часть этого меню (кроме нижнего пункта) относится к операциям со словарем логических символов. Здесь предусмотрены следующие действия:

1) Ввод названия нового логического символа (клавиша "н", которую можно нажимать не только из данного меню, но и непосредственно из главного меню). После выбора пункта появляется диалогове окно, в котором вводится необходимое название. В качестве названия может выступать произвольная последовательность букв, цифр и других прорисовываемых текстовым редактором на экране знаков, имеющая длину не более 24. Большие и маленькие буквы различаются. Если введенное название уже было использовано, то выдается сообщение "Логический символ с таким названием уже имеется", после чего нажатие любой клавиши возвращает в меню, и попытку ввода нового символа следует повторить, скорректировав название. Если название было новым, то находится первый неиспользуемый (не имеющий названия) номер логического символа, и ему присваивается данное название.

2) Изменение названия логического символа (клавиша "Ctr-н", которую можно нажимать непосредственно из главного меню). После выбора пункта появляется диалоговое окно, в котором вводится старое название логического символа. После того, как оно было введено и нажата клавиша "Enter", появляется предложение ввести новое название символа. При вводе нового названия может оказаться, что это название уже было использовано. Тогда выдается соответствующее сообщение, и происходит откат к повторному вводу нового названия. Если в этом случае отказаться от ввода нового названия, то старое будет сохранено. Следует учесть (!), что если имелся логический символ без названия, номер которого меньше номера изменяемого символа, то при указанном выше откате символ потеряет название, а его старое название будет присвоено первому не имевшему названия символу.

3) Определение номера логического символа по его названию (клавиша "с"). После выбора пункта возникает диалоговое окно, в котором вводится название логического символа. По нажатии клавиши "Enter"под введенным названием появляется номер символа.

4) Определение логического символа по его номеру (клавиша "Ctr-с"). После выбора пункта возникает диалоговое окно, в котором набирается номер логического символа. По нажатии клавиши "Enter"под введенным номером появляется название символа. Заметим, что данный и предыдущий пункты практически не используются собственно при программировании на ЛОСе, а бывают нужны лишь при отладке интерпретатора ЛОСа, периодически возникающей в связи с различными расширениями ЛОСа.

5) Удаление названия логического символа (клавиша "Ctr - у", кириллица). После выбора пункта возникает диалоговое окно, в котором набирается удаляемое название символа. Следует заметить (!), что удаление названия логического символа, используемого в программах ЛОСа. нежелательно: при выдаче на экран текстов таких программ название символа не будет прорисовываться, и при попытке редактировать программу она будет искажена.

7.1.3 Перечень названий операторов ЛОСа

Для поиска нужного оператора ЛОСа создано оглавление, в котором базисные и основные реализованные на ЛОСе операторы и операторные выражения ЛОСа упорядочены по своему назначению. Это оглавление достижимо из главного меню ЛОСа при нажатии клавиши "о"(кир.); соответствующий пункт находится в правом нижнем углу. В него также можно войти из редактора программ ЛОСа (см. ниже). Через концевой пункт оглавления осуществляется выход (нажатием клавиши "курсор вправо") в справочную информацию о логическом символе, являющемся заголовком соответствующего оператора либо операторного выражения. Интерфейс работы с такой справочной информацией уже был описан выше. Заметим, что в данном случае появляется возможность перехода к просмотру корневого фрагмента программы текущего логического символа - для этого достаточно нажать клавишу "Home". Для возвращения из просмотра программы нажимается "End".

Оглавление операторов ЛОСа можно пополнять самостоятельно, используя для этого следующие действия. После создания программы для нового оператора либо операторного выражения следует найти подходящий раздел оглавления либо ввести новый такой раздел; в нем создать концевой пункт с краткой характеристикой действий, выполняемых новой программой. Далее - нажать клавишу "Enter". Текст пункта окажется перерисованным в верхней части экрана; под ним будет проведена горизонтальная линия. После повторного нажатия "Enter"под данной линией возникнет курсор текстового редактора, с помощью которого следует ввести название нового оператора (операторного выражения). После завершения ввода (еще одно "Enter") нажимается "курсор влево", и новый концевой пункт создан.

7.2 Интерфейс редактора программ

Как уже говорилось выше, программа на ЛОСе представляет собой дерево фрагментов, переходы между которыми выполняются с помощью операторов перехода "ветвь N", "иначе N". Каждый фрагмент является последовательностью операторов ЛОСа. Редактор программ ЛОСа осуществляет постраничный показ фрагментов программы, так что каждый фрагмент целиком умещается на экране. Никаких прокруток изображения на экране при этом не предусмотрено. Если фрагмент слишком большой и целиком на экране не помещается, то он разрезается на части, между которыми имеются линейные переходы: в конце очередной части помещаются операторы "ветвь N", "продолжение", где N - номер перехода к следующей части. Операторы размещаются в программе, отделенные друг от друга пробелами; никаких других разделяющих знаков быть не должно. Их последовательность не форматируется; при автоматической перерисовке операторы идут друг за другом с промежутками ровно в один пробел. Если строка заканчивается, то текст продолжается с начала следующей строки без каких-либо пробелов и знаков переноса. Разумеется, такой стиль изображения программы отличается от общепринятого. Однако, плотный режим размещения операторов имеет свои достоинства: удается разместить больший объем информации на меньшем пространстве, что упрощает анализ контекста при написании программы и ее чтении. Решающим фактором, сделавшим плотный режим размещения операторов практически приемлемым и даже удобным, оказалась специальная многоцветная указка, которая, по существу, заменяет форматирование, позволяя концентрировать внимание на нужном участке текста и определяя архитектуру его включения в целый текст. Разумеется, возможно создать более традицион-

ные способы изображения ЛОС-программы на экране - если они обеспечат большую эффективность программирования, чем настоящая версия. Впрочем, уже сейчас подавляющее большинство приемов записывается не на ЛОСе, а на языке значительно более высокого уровня - ГЕНОЛОГе. Поэтому более перспективным направлением, чем совершенствование интерфейса редактора программ ЛОСа, представляется (естественное для логической системы) развитие средств автоматического создания ЛОС - программ.

7.2.1 Вход в редактор программ ЛОСа

Вход в редактор программ ЛОСа через главное меню обычно осуществляется одним из двух способов - с помощью явного указания логического символа, программу которого нужно просматривать либо редактировать, либо через оглавление программ ЛОСа, в котором представлены основные блоки системы или большие вспомогательные процедуры общего назначения.

Для входа в корневой фрагмент программы логического символа A нужно либо нажать клавишу "п", либо нажать левую клавишу мыши, переведя ее курсор в окно главного меню "Просмотр программы логического символа". В обоих случаях в синей рамке, расположенной внутри этого окна, появится курсор текстового редактора. Далее нужно набрать название символа A и нажать "Enter". При наборе последовательности букв, не являющейся названием логического символа, синяя рамка снова становится пустой, и набор следует повторить либо нажать "Esc". При правильном наборе символа на экране появится изображение корневого фрагмента программы символа A. Если такой программы еще не было создано, то в верхней части экрана будет перерисовано название символа A, а остальная часть экрана будет пустой.

Для входа в оглавление основных программ ЛОСа из главного меню нужно либо нажать клавишу "л", либо переместить курсор мыши в окно "Оглавление программ"и нажать левую клавишу мыши. После этого на экране возникает подменю оглавления программ, которое рассматривалось перед последним выходом из этого оглавления. Неконцевые пункты оглавления программ соответствуют основным блокам либо вспомогательным процедурам системы, или подблокам таких блоков и процедур. Фактически ветвь оглавления программ является вынесенной в отдельную структуру данных иерархической системой комментариев к программе. Концевые пункты этой ветви жестко связаны с конкретными точками в программе; выйдя на такой пункт и нажав клавишу "курсор вправо", можно оказаться в соответствующей контрольной точке программы. Эта точка будет обозначена в прорисованном на экране фрагменте программы выделенным синим цветом фиктивным оператором "прием(N)"; N - последовательность цифр, образующая номер контрольной точки. Нумерация контрольных точек - своя для каждого логического символа. Для возвращения в оглавление из просмотра фрагмента программы следует нажать "End". Для перехода к просмотру надфрагмента либо подфрагментов, либо для изменения фрагмента нужно выйти из режима просмотра подтермов операторов фрагмента, в котором мы оказываемся после нажатия "курсор вправо", нажав клавишу "о". Заметим, что возвращение в оглавление может быть осуществлено из любого фрагмента просматриваемой программы (причем в тот концевой пункт оглавления, из которого был сделан переход к рассмотрению программы). Однако, для такого возвращения нужно сначала восстановить режим просмотра подтермов операторов (снова нажатием клавиши "о"). Способ регистрации в оглавлении программ ЛОСа новых контрольных точек описывается ниже.

В принципе, возможен еще один способ входа в просмотр фрагментов программы - через оглавление операторов ЛОСа. Это оглавление (как уже говорилось выше) достижимо из главного меню по клавише "o"(выбор пункта в правом нижнем углу экрана). После перехода в концевой пункт такого оглавления и нажатия клавиши "курсор вправо"появляется справочная информация о логическом символе - заголовке оператора. При нажатии "Home"здесь осуществляется переход к просмотру корневого фрагмента программы данного логического символа (но уже не в режим просмотра подтермов операторов, а в обычный режим просмотра). После просмотра и редактирования этого либо других фрагментов той же программы возможно возвращение в оглавление операторов - по нажатии клавиши "End".

Наконец, имеется возможность входа в просмотр программы приема, заданного на ГЕНОЛО́Ге - об этом будет сказано в разделе, посвященном редактору ГЕНОЛОГа.

7.2.2 Просмотр фрагментов программы

Перемещение по дереву фрагментов программы

При просмотре фрагмента программы переходы из этого фрагмента в подфрагменты занумерованы числами $1, 2, \ldots$. Каждый такой переход представляет собой оператор "ветвь N"либо "иначе N", где N - номер перехода. Один из этих переходов ("текущий") выделен желтым цветом. Используя клавиши "курсор вправо" (к следующему переходу) и "курсор влево" (к предыдущему переходу), можно выбрать нужный текущий переход. После этого нажатие клавиши "курсор вниз" переводит в просмотр подфрагмента, к которому ведет данный переход. Последующее нажатие клавиши "курсор вверх" вызовет возвращение к исходному фрагменту.

Если клавиша "курсор вверх" нажимается в корневом фрагменте, то происходит возвращение в главное меню (именно, в режим ввода логического символа, программу которого требуется просмотреть). Нажатие клавиши "PageUp" из произвольного фрагмента программы логического символа также позволяет (причем за один шаг) попасть в указанный режим главного меню. Обычно эти возможности используются для перехода к просмотру программы другого логического символа.

Если вход в редактор программ ЛОСа имел место через указание логического символа в главном меню, а не из какого-либо оглавления или из редактора ГЕНОЛОГа, то нажатие клавиши "End"при просмотре любого фрагмента программы переводит в корневой фрагмент этой программы.

Для перемещения по дереву фрагментов программы с помощью мыши следует подвести курсор мыши к нужному оператору перехода в подфрагмент и нажать правую кнопку (нажатие левой кнопки переведет в режим просмотра подтермов). Чтобы вернуться из подфрагмента в надфрагмент, следует перевести курсор мыши в зону под текстом программы и также нажать правую кнопку.

Режим просмотра подтермов операторов фрагмента программы

Многие операторы ЛОСа представляют собой сложные многоэтажные логические конструкции, текст которых занимает иногда значительную часть экрана. Для того, чтобы облегчить концентрацию внимания на отдельном фрагменте такого оператора и, более того, сделать видимой его "архитектуру", используется специальная многоцветная указка - режим просмотра подтермов операторов. Вход в режим просмотра подтермов обеспечивается нажатием клавиши "o"(кириллица). При этом пропадает

выделение желтым цветом текущего перехода к подфрагменту, но первый оператор фрагмента программы окрашивается в голубой цвет. Используя клавиши "курсор вправо"и "курсор влево", можно последовательно выделять все операторы текущего фрагмента.

После выбора отдельного, представляющего интерес, оператора, можно выделить новым (отличным от голубого) цветом один из его корневых операндов. Для этого достаточно нажать сначала на клавишу "курсор вниз", и далее клавишами "курсор вправо"и "курсор влево"обеспечить выбор требуемого корневого операнда. Повторяя эту операцию для уже выделенного на некотором уровне операнда (снова нажимая клавишу "курсор вниз", и т.д.), можно войти в просмотр подоперандов этого операнда. Чтобы избежать смешения цветов, они чередуются в соответствии с некоторым фиксированным списком цветов (по достижении последнего в списке цвета снова используется первый).

Нажатие клавиши "курсор вверх"возвращает из просмотра подоперандов к внешней операции. Если уже достигнут уровень просмотра всего оператора в целом, то нажатие данной клавиши игнорируется.

Для возвращения из режима просмотра подтермов операторов к обычному режиму просмотра фрагмента программы достаточно повторно нажать клавишу "o".

Можно войти в режим просмотра подтермов, переведя курсор мыши к корню представляющего интерес подтерма и нажав левую кнопку мыши. Если нужно перейти к выделению другого подтерма - операция повторяется. Для выхода из режим просмотра подтермов можно перевести курсор мыши в зону под текстом программы и нажать левую кнопку.

Если при выделенном текущем подтерме, заголовком которого служит логический символ, нажать клавишу F3, то на экране появится справочная информация об этом логическом символе (интерфейс работы с ней уже был описан). Выход из просмотра этой информации - по любой клавише, кроме используемых для перелистывания и редактирования справочных текстов (например, по нажатию пробела).

Если нужно перейти к программе логического символа, выделенного при просмотре подтермов в текущем фрагменте программы, то нажимается клавиша "c". После этого появляется корневой фрагмент программы данного логического символа. Из этого фрагмента, далее, можно повторно перейти указанным образом к новому фрагменту, и т.д. По этой цепочке переходов можно возвращаться, нажимая клавишу "End" (при каждом возвращении устанавливается корневой фрагмент программы, а не тот, из которого имел место переход; восстановлению исходного фрагмента для первого элемента цепочки здесь иногда помогает нажатие клавиши F8 - например, если переход к нему был из оглавления программ ЛОСа либо из редактора ГЕНОЛОГа).

Справочная информация о логических символах

Чтобы при просмотре программы получить информацию о том, что делает тот или иной оператор, или что означает некоторое понятие, с которым работает прием, можно пользоваться справочной информацией о логических символах. Она размещена в 3-м информационном блоке, и фактически представляет собой записную книжку, в которой собрана вся необходимая техническая информация о логических символах - описания операторов, смысл применяемых в задачах обозначений, пояснения к использованию целей и комментариев задач, заголовком которых является данный логический символ, и т.п. По мере ввода новых символов и новых операторов,

эту записную книжку следует регулярно пополнять новыми данными. Интерфейс просмотра и редактирования справочной информации о логическом символе уже описан выше в подразделе "Разное"раздела "Общие операторы интерфейса". Перечислим возможные способы обращения к этой информации из просмотра фрагмента программы.

Если требуется получить информацию о логическом символе, являющемся заголовком данной программы, то нажимается клавиша F3. Если нужна информация о каком-либо другом логическом символе, то следует нажать клавишу F2. Далее возникает окно диалога, в котором следует набрать название нужного символа и нажать "Enter".

Для получения справочной информации о логическом символе, встречающемся в текущем фрагменте программы, можно подвести курсор мыши к этому символу и нажать левую кнопку мыши. Тогда произойдет переход в режим просмотра подтермов операторов, в котором текущим окажется подтерм с корнем в выбранном логическом символе. Далее нажимается правая кнопка мыши, и на экране возникает справочная информация о символе. Для выхода из ее просмотра достаточно нажать любую кнопку мыши (кроме нажатия этой кнопки на пунктах меню $\rightarrow, \leftarrow$, которое вызовет перелистывание справочной информации). После этого можно либо убрать режим просмотра подтермов, выведя курсор мыши за рамки текста программы и нажав левую кнопку, либо перевести курсор к другому представляющему интерес логическому символу и нажать левую кнопку, чтобы соответствующий подтерм стал текущим.

Кроме справочной информации о логических символах, при просмотре фрагмента программы можно войти в оглавление операторов ЛОСа. Для этого достаточно нажать "Ctr-л". Напомним, что из главного меню к этому же оглавлению переход происходил по нажатию клавиши "о".

Поиск заданного терма в ветви программы логического символа

Для того, чтобы найти все вхождения в операторы текущей ветви программы заданного терма либо логического символа, следует нажать клавишу F4 и далее набрать в окне диалога текстовым редактором требуемый терм либо логический символ. После нажатия "Enter"будет либо прорисовано многоцветной указкой первое найденное вхождение, либо (если таких вхождений в ветви вообще нет) возникнет пустой экран; по нажатию любой клавиши от него - возвращение в просмотр фрагмента. При поиске вхождений, кроме текущего фрагмента программы, просматриваются все достижимые из него фрагменты. Здесь используется принцип "сначала вглубь", причем непосредственные подфрагменты любого фрагмента упорядочиваются "от начала к концу" (т.е. по убыванию локальных номеров).

Для перехода к следующему вхождению искомого терма (символа) повторно нажимается F4. Если найдено представляющее интерес вхождение, то можно (как обычно, нажатием клавиши "о") выйти из режима просмотра подтермов, автоматически установленного при поиске, и выполнить необходимые операции (например, по редактированию программы). При этом установка на поиск заданного терма (символа) сохраняется, однако при повторном нажатии F4 будет выполняться поиск только в той ветви, в которой это нажатие (первое в возобновленном цикле поиска) было осуществлено. Поэтому перед возобновлением цикла поиска следует сначала выйти на корень необходимой ветви программы.

Если после очередного нажатия F4 возникает пустой экран, то поиск завершен.

Только после этого установка на заданный терм (символ) сбрасывается. Далее при нажатии любой клавиши - переход в обычный режим просмотра фрагментов программы.

Последовательный просмотр программ логических символов

Можно последовательно просматривать программы всех логических символов, начиная с заданного - в соответствии с нумерацией логических символов. Для перехода к корню программы очередного логического символа (первого после текущего, для которого создана программа) нажимается клавиша "ш". Кроме этого простейшего средства полного просмотра программ "вручную", имеется возможность автоматического просмотра всех программ системы, связанная с использованием специального оператора "смпрог". При полном просмотре всех программ и входящих в них операторов, ему передается информация о каждом текущем подоператоре. В зависимости от того, как будет запрограммирован оператор "смпрог", при этом может происходить поиск всех мест в программе, удовлетворяющих заданному условию, и изменение их по заданному принципу. Кроме того, будет выполняться фоновый (не зависящий от оператора "смпрог") поиск типовых ошибок в программах. Наличие такого режима автоматического просмотра сделало пока излишним развитие других средств поиска и контроля в программах ЛОСа; оно позволило обращаться со всем многообразием программ системы как с единым целым и вводить в них при необходимости любые изменения (в том числе, связанные с развитием языка ЛОС). Подробнее о режиме автоматического просмотра программы будет сказано ниже, после изложения необходимой для изменения оператора "смпрог"техники редактирования программ.

7.2.3 Редактирование текущего фрагмента программы

Для начала редактирования текущего просматриваемого (в обычном режиме) фрагмента программы следует нажать клавишу "р"(кир.). При этом в правом верхнем углу экрана появляется красный прямоугольник, предупреждающий о том, что в случае нажатия клавиши "Enter"новая версия текста фрагмента будет немедленно записана в файлы блока программ, а старая пропадет (в этом случае, все же, при необходимости можно будет извлечь из резервной директории SLCOPY все файлы ранее сохраненной в ней версии решателя). Редактирование осуществляется текстовым редактором.

Для завершения редактирования нажимается клавиша "Enter"; для отмены изменений фрагмента - "Esc". Чтобы изменения фрагмента были сохранены, в тексте не должно иметься незавершенных или неправильно набранных операторов. Попытка сохранить такой некорректный текст приводит обычно к появлению сообщения об ошибке, прорисовываемого внутри красного прямоугольника в правом верхнем углу. Курсор текстового редактора при этом перемещается либо в начало текста, либо к той точке, около которой произошла ошибка. После устранения ошибки попытка сохранения текста фрагмента может быть повторена. Предусмотрены следующие типы сообщений об ошибках:

1) "Незакрытая левая скобка" - число левых скобок в тексте больше числа правых. Курсор перемещается в начало текста.

2) "Избыточная правая скобка" - число правых скобок в некоторой точке превысило число предшествующих левых. Курсор перемещается к этой точке.

3) "Ошибка в логическом символе" - при наборе логического символа допущена ошибка либо произошла склейка идущих подряд названий символов или переменных. Курсор перемещается к началу неправильно набранного символа.

4) "Одинаковые метки" - несколько различных операторов перехода из фрагмента имеют один и тот же номер перехода. Курсор перемещается в начало текста.

5) "Недопустимая переменная" - номер переменной больше 512. Курсор перемещается в начало записи этой переменной.

Ссылки из фрагмента на подфрагменты нумеруются при редактировании произвольным образом натуральными числами. Впоследствии, при перерисовках фрагмента, они будут автоматически переобозначены на $1, 2, 3, \ldots$. Если изменяется старая версия фрагмента, то исключение оператора перехода, ссылающегося на подфрагмент, либо изменение номера его ссылки на новый, не использовавшийся ранее в фрагменте, приведут к полной потере всей ветви этого оператора.

Если в редактируемом фрагменте возникли новые номера перехода, то после нажатия клавиши "Enter" и регистрации в файлах блока программ новой версии фрагмента на экране появляется в верхнем левом углу один из новых номеров перехода (первый при движении от конца фрагмента к началу), а остальная часть экрана расчищается (остается только курсор текстового редактора). Это - приглашение к вводу подфрагмента с указанным номером. От него можно отказаться нажатием клавиши "Esc", а можно воспользоваться, введя новый подфрагмент. Этот подфрагмент, в свою очередь, может содержать ссылки на подфрагменты, и тогда снова появится приглашение к вводу последнего из них, и т.п. В результате, если не нажимать "Esc", будут по принципу "сначала вглубь" перечислены все ссылки на новые подфрагменты. Здесь важно не использовать дважды одного и того же номера ссылки (в процессе перечисления новых подфрагментов нумерация ссылок - глобальная, но по завершении перечисления эти номера теряются и вводится локальная нумерация).

Если при наборе подфрагментов нажималась клавиша "Esc", то останутся недоопределенные переходы. Лучше всего доопределить их сразу, по завершении указанного выше перечисления подфрагментов. Для создания фрагмента по недоопределенному переходу, следует нажать на этом переходе клавишу "курсор вниз"(экран расчистится) и войти в его редактирование повторным нажатием "p". Вообще, предпочтительнее не нажимать при перечислении подфрагментов клавишу "Esc", а вводить какой-либо простой оператор для последующего возвращения к этому подфрагменту и фактического его набора (например, оператор "продолжение").

Исходный фрагмент программы любого оператора, справочника либо операторного выражения должен начинаться с оператора "метка(икс(N))", где N - символьный номер первой программной переменной, не определенной при обращении и не являющейся выходной переменной. Этот оператор должен размещаться сразу после операторов "программа"либо "обращение(t)", указывающих тип обращения к программе. В случае сканирования задачи (т.е. после оператора "решить") оператор "метка(икс(...))" не используется.

Чтобы получить справочную информацию о логическом символе непосредственно в процессе редактирования, нажимается клавиша F4 и в диалоговом окне вводится название этого символа. После просмотра информации восстанавливается изображение на момент прерывания редактирования. Если при наборе текста нужно ввести некоторый символьный номер N, больший двадцати (номера от 0 до 20 обозначены логическими символами "0", ..., "9", "десять", ..., "двадцать"), то вводится запись "логсимвол(N)". По окончании набора текста, после нажатия "Enter", эта запись будет автоматически переведена в требуемый символьный номер, и впоследствии

в данном месте программы он будет прорисовываться в явном виде. Если нужно вставить в программу логический символ - код клавиатуры, то нажимается клавиша F8, после чего нажимается та клавиша, для которой нужен код. Этот код будет прорисован автоматически, начиная с текущей позиции курсора. Если нужен код для режима латинских букв, то предварительно нажимается F12; для возвращения к режиму кириллицы нажимается F11.

Для копирования части фрагмента программы в другие фрагменты можно использовать буфер текстового редактора: войти в режим редактирования фрагмента, содержащего нужный текст (клавиша "p"); занести этот текст в буфер; выйти из режима редактирования, не изменяя программы (клавиша "Esc"); войти в редактирование того фрагмента, в котором нужно вставить копию, и извлечь ее из буфера текстового редактора. При этом, однако, не будут скопированы те фрагменты, к которым из выделенного текста имелись переходы - их придется создавать повторно. Чтобы скопировать всю ветвь программы, предусмотрена специальная операция (см. ниже).

Если в процессе набора программы возникла необходимость получить информацию, недоступную из текстового редактора, то следует донабрать (произвольным образом, но со строгим соблюдением синтаксических правил для операторов со связанными переменными) текущий оператор, сохранить набранную часть фрагмента нажатием "Enter", выйти в просмотр необходимых данных, а затем вернуться в прерванное редактирование.

При редактировании программы в блоке программ обычно возникают "пустоты" - неиспользуемые остатки предыдущих версий фрагментов программ. Эти пустоты рекомендуется периодически устранять, выбирая в главном меню пункт "Уплотнение измененных файлов". Кроме исключения неиспользуемых фрагментов, операция уплотнения осуществляет контроль за корректностью общей структуры программ. Если по причине ошибки или машинного сбоя в блоке программ обнаруживается дефект, то при выборе указанного пункта либо пункта "Сохранение копий файлов" (тоже осуществляющего данную проверку) произойдет выход из программы в операционную систему. В этом случае следует воспользоваться сохраненной в резервной директории SLCOPY копией программы. Такую копию следует постоянно обновлять выбором в главном меню пункта "Сохранение копий файлов". Рекомендуемый режим работы - регулярный выбор пунктов "Уплотнение...", "Сохранение..." после любых изменений в программе и информационных блоках.

7.2.4 Сдвиг номеров программных переменных

При программировании на ЛОСе следует придерживаться принципа последовательной нумерации новых переменных - это связано в первую очередь с логикой откатов, при которых сбрасываются значения всех переменных, номера которых больше некоторого, а также с экономией объема глобального стэка интерпретатора ЛОСа. Однако, иногда возникает необходимость при доработке программы вставлять внутри нее операторы, вычисляющие те или иные дополнительные значения. Чтобы ввести для этих значений программные переменные, не нарушая принципа последовательной нумерации, приходится выполнять сдвиг (в данной ситуации - увеличение) номеров всех программных переменных, определяемых после точки вставки оператора, на заданную величину. Это осуществляется следующей последовательностью действий:

а) Войти в режим просмотра подтермов операторов и выделить оператор, после которого (включая его самого) требуется выполнить сдвиг нумерации переменных.

б) Нажать клавишу "п". Тогда в левом верхнем углу экрана возникает курсор текстового редактора. Если нужно увеличить номера переменных, большие или равные n, на величину m, то набирается терм "плюс(хn m)"(буква "х кир.). Если нужно уменьшить номера переменных, большие или равные n, на величину m, то набирается терм "минус(хn m)". В обоих случаях после нажатия "Enter"выполняется указанный сдвиг (он затрагивает не только данный фрагмент, но и все подфрагменты, достижимые из выбранного оператора).

7.2.5 Операции с ветвями программы

Чтобы скопировать всю ветвь некоторого фрагмента программы, из обычного просмотра этого фрагмента нажимается клавиша "Insert это приводит к регистрации ссылки на фрагмент в специальном буфере редактора программ. Затем предпринимается переход к тому фрагменту программы, всю ветвь которого следует заменить на всю ветвь исходного (учтенного в буфере) фрагмента. Эта ветвь НЕ ДОЛЖНА находиться внутри заменяющей ветви либо содержать ее. Как правило, такую ветвь сначала приходится специально создавать - состоящей из единственного фрагмента с единственным оператором "продолжение" - ответвляя в нужном месте переход к ней по "ветвь" либо "иначе". Если заменяемая ветвь относится к программе другого логического символа, то выход в главное меню для перехода к программе этого символа должен происходить только по нажатии клавиши "PageUp" - иначе ссылка в буфере на заменяющую ветвь будет утеряна. После того, как на экране возник корневой фрагмент заменяемой ветви, последовательно нажимаются клавиши "о" (кир.) - для перехода в режим просмотра подтермов - и "к" (кир.) - собственно для копирования ветви.

Для удаления ветви программы, переход к которой из текущего фрагмента программы определяется выделенным желтым цветом оператором перехода, достаточно нажать клавишу "Ctr-Del". Эту операцию можно применять лишь однократно; для повторного удаления с ее помощью следует сначала выйти из просмотра программы текущего логического символа.

Для удаления всей программы логического символа следует перейти в корневой фрагмент этой программы и нажать "Ctr-F7".

Для удаления сразу нескольких ветвей программы, к которым из текущего фрагмента имеются переходы, можно войти в режим редактирования этого фрагмента и исключить операторы перехода, ссылающиеся на указанные ветви. По завершении редактирования (клавиша "Enter") эти ветви будут полностью удалены.

Если ветвь программы была удалена с помощью "Ctr-Del"либо "Ctr-F7", то ее можно восстановить в качестве корневой ветви всей программы данного логического символа - достаточно в корневом фрагменте программы нажать "Ctr-F6". Такое нажатие должно быть выполнено до выхода из программы текущего логического символа. Впрочем, для удаления части программы, находящейся выше некоторого фрагмента, более удобным оказалось использовать операцию слияния двух последовательных фрагментов (см. ниже).

Если фрагмент программы оказался чрезмерно большим (например, после вставки в него дополнительных операторов), то его можно разрезать на две части. Для этого следует войти в режим просмотра подтермов, выделить тот оператор, который должен стать первым оператором второй части, и нажать клавишу "р"(кир.). К концу первой части фрагмента будут добавлены оператор "ветвь N", ссылающийся на вторую часть, а также оператор "продолжение" (переход ко второй части будет

выполнен после "отражения"от этого завершающего оператора).

Если два фрагмента программы выполняются последовательно - в том смысле, что первый из них завершается оператором "ветвь N", ссылающимся на второй, и оператором "продолжение", - то можно выполнить слияние их в один общий фрагмент. Эта операция в точности обратна описанной выше операции разрезания фрагмента. Разумеется, применять ее следует так, чтобы результат слияния умещался целиком на экране. В принципе, появление фрагментов, не помещающихся на экране целиком, допустимо (они иногда возникают при работе компилятора ГЕНОЛОГа). Однако, для удобства работы их лучше разрезать на части. Во всяком случае, это необходимо делать перед редактированием такого фрагмента.

7.2.6 Обнаружение ошибок в программе

Предусмотрена возможность автоматического контроля правильности программы логического символа, включающего обнаружение простейших ошибок - несоответствия числа операндов операторов и операторных выражений их арности, использование значения переменной, которое еще не было определено, попытка повторно определить уже определенное значение, и т.п. Для выполнения такого контроля следует войти в просмотр корневого фрагмента программы и нажать клавишу "п". При наличии ошибок будут выданы соответствующие сообщения (при этом произойдет переход в режим просмотра подтермов и будет выделена точка ошибки - для последующего ее исправления нужно будет сначала нажатием клавиши "о"(кир.) перейти в обычный просмотр фрагмента). При отсутствии ошибок, по окончании анализа программы, произойдет перерисовка корневого фрагмента.

Если контролируется программа нового оператора либо операторного выражения, причем после первого оператора ее корневого фрагмента, уточняющего тип обращения, не идет "иначе", то процедура контроля автоматически вводит программу справочника "арность", указывающую на число операндов этого оператора либо операторного выражения. Для перечисляющих операторов вводится также программа справочника "комментпосылок". При определении арности здесь используется оператор "метка(икс(N))", размещаемый в начале контролируемой программы.

Возможна серийная проверка программ логических символов, начиная с текущего логического символа. Для запуска этой проверки нажимается клавиша "П". Остановка серийной проверки осуществляется нажатием любой клавиши. Для просмотра всех программ системы следует начинать с логического символа "метка", имеющего номер 1. При просмотре перерисовки фрагмента программы на экране не происходит; меняются только заголовки просматриваемых программ в верхней части экрана.

При контроле программы одного логического символа (клавиша "п") либо при серийном контроле (клавиша "П") возможно выполнение дополнительных действий, определяемых программой оператора "смпрог(a b c d e)". К этому оператору происходят обращения для каждого текущего просматриваемого вхождения в операторы фрагментов программ. При этом значением переменной a становится логический символ, к которому относится фрагмент программы; значением переменной b - набор $(F_1, \ldots, F_n)$, определяющий путь от корня программы символа a к текущему просматриваемому фрагменту программы. Каждое F_i есть тройка (F_{i1}, F_{i2}, F_{i3}), у которой (F_{i1}, F_{i2}) - ссылка на фрагмент программы;F_{i3} - символьный номер перехода от этого фрагмента к подфрагменту, соответствующему тройке F_{i-1}. Последний элемент набора b определяет переход от корня программы символа a; первый элемент - переход к текущему фрагменту. Значением переменной c служит набор троек $(G_1,$

$\ldots, G_{n+1})$; $G_i = (G_{i1}, G_{i2}, G_{i3})$. Эти тройки описывают фрагменты программы , расположенные на пути, определяемом набором b; G_{n+1} соответствует корню программы символа a, а G_1 - текущему фрагменту. G_{i1} есть сам фрагмент программы, представленный набором термов в зоне задач; G_{i2} - набор дополнительных логических условий на программные переменные, встречающиеся в операторах фрагмента G_{i1}. G_{i3} - набор информационных элементов, характеризующих фрагмент G_{i1}. Для указания корневого фрагмента здесь используется элемент "корень"; для указания вхождения v в фрагмент оператора перехода, ссылающегося на подфрагмент текущего пути - элемент (позиция v). Входной переменной d оператора "смпрог(...)"присваивается вхождение текущего оператора в список операторов текущего фрагмента; переменной e - вхождение текущего логического символа в текущем операторе.

Если оператор "смпрог"оказывается истинным в некоторой точке просмотра, то просмотр обрывается, и многоцветной указкой выделяется текущее вхождение в текущий оператор текущего фрагмента. Для продолжения просмотра после этого (кроме случаев сообщения о найденной ошибке) следует нажать клавишу "ш". Следует учитывать, что при выходе из режима многоцветной указки цикл просмотра сбрасывается, и тогда нужно запускать его с текущего логического символа заново.

Оператор "смпрог"можно использовать для поиска точек в программе, удовлетворяющих заданным условиям. Эти условия, с помощью редактора программ, записываются в начале программы данного оператора (предварительно следует удалить тождественно ложный оператор, обычно вставляемый в начало программы "смпрог"для предотвращения ее срабатываний в режиме общей проверки; обычно таким оператором является "равно(0 1)"). В найденной точке можно изменить программу вручную и возобновить поиск, начиная с текущего логического символа. Возможна реализация цикла автоматического изменения всей программы системы - для этого в операторе "смпрог"нужно реализовать процедуру модификации фрагмента или целой ветви программы, и заменить старый фрагмент (ветвь) на новый, используя специальные операторы для работы с фрагментами программ, описанные в разделе, посвященном программной реализации редактора программ.

7.2.7 Сопровождение справочной информацией избранных точек в программе

Чтобы сопровождать избранные точки программ ЛОСа необходимыми пояснениями, а также чтобы быстро можно было находить нужную процедуру и находить в этой процедуре конкретную точку, создано специальное оглавление, в котором перечислены основные процедуры системы, реализованные на ЛОСе. Об этом оглавлении уже говорилось вкратце выше (при описании способов входа в просмотр программ); вход в него - через пункт "Оглавление программ" главного меню. Концевые пункты данного оглавления суть комментарии к избранным точкам программы. Для входа в просмотр такой точки достаточно нажать в концевом пункте клавишу "курсор вправо". После этого возникает текст фрагмента, в котором синим цветом выделен оператор "прием(i)" - он отмечает нужно место в программе. Здесь имеет место режим просмотра подтермов операторов. Далее можно произвольным образом перемещаться по фрагментам программы (не выходя в главное меню) и менять режим просмотра; если в произвольной точке восстановить режим просмотра подтермов и нажать "End", то будет восстановлен исходный концевой пункт оглавления программ.

Если при просмотре фрагментов программы после перехода к ним из оглавления

программ (в обычном режиме просмотра) нужно вернуться к тому фрагменту, с которого был начат этот просмотр, то нажимается F8. Произойдет возвращение к просмотру исходного оператора "прием(i)", причем восстановится режим просмотра подтермов.

Если нужно связать комментарий с какой-либо точкой программы, то следует войти в режим просмотра подтермов, выделить тот оператор A, к которому относится этот комментарий, и нажать клавишу "PageDown". После этого на экране возникнет некоторая страница оглавления программ ЛОСа. Перемещаясь по этому оглавлению, а при необходимости - создавая новые его подразделы, следует перейти в то меню, которое связано с описанием рассматриваемой части программы. Далее нужно ввести новый концевой пункт этого меню (клавиша "к"либо "К"), и текстом этого пункта сделать требуемый комментарий. По окончании ввода текста (нажатие клавиши "Enter") перед оператором A будет вставлен фиктивный оператор "прием(N)", ссылающийся на введенный концевой пункт оглавления программ.

Если вход в программу происходил не из оглавления программ, то для получения информации оглавления программ, относящейся к выбранной ее точке, следует выделить в этой точке многоцветной указкой какой-либо оператор и нажать "End". Тогда будет найден первый предшествующий выделенной позиции оператор "прием(N)", и осуществится переход к его концевому пункту оглавления программ.

Чтобы удалить концевой пункт оглавления программ и одновременно удалить ссылку "прием(i)"в программе, соответствующую этому пункту, достаточно нажать "Ctr-Del"при выделенном концевом пункте оглавления программ.

7.2.8 Дополнительные возможности интерфейса редактора программ

Находясь в режиме просмотра фрагмента программы, можно выполнить ряд вспомогательных операций. Прежде всего, именно из этого режима осуществляется вход в создание и редактирование шаблонов диалоговых блоков. При нажатии клавиши "д" возникает оглавление диалоговых блоков, в котором можно либо выбрать для просмотра и редактирования ранее созданный шаблон блока диалога ("курсор вправо" на выбранном пункте; для возвращения в оглавление из редактирования шаблона - "Esc"), либо создать новый шаблон. Последнее осуществляется созданием нового концевого пункта и входом в него - так же, как и для ранее созданного пункта. Интерфейс редактирования шаблона блока диалога был описан выше. Для удаления шаблона достаточно удалить (Ctr-Del) соответствующий концевой пункт оглавления.

Кроме оглавления шаблонов блоков диалога, из режима просмотра фрагмента программы можно перейти в ряд других справочных оглавлений, которые бывает нужно использовать при редактировании программ: оглавление информационных элементов компилятора ГЕНОЛОГа (Ctr-п); оглавление типов комментариев к элементу фразы, используемых анализатором текстов русского языка (Ctr-ф); оглавление типов примечаний к теореме при ее целевой характеризации (Ctr-т). Подробнее об этих оглавлениях будет сказано в последующих разделах.

Глава 8

Отладчик ЛОСа

8.1 Семантическая трассировка решения задачи

Семнатическая трассировка представляет собой пошаговый просмотр процесса решения задачи с отображением на экране сообщений о применении приемов, сопровождаемых пояснениями. Такая трассировка позволяет понять способ решения задачи "в целом"и локализовать моменты, требующие специальной доработки.

Способы запуска решения задачи уже были описаны выше в разделе "Общая схема функционирования решателя". Для запуска решения без трассировки нажимается клавиша "о"; для запуска решения с семантической трассировкой нажимается "р", причем предварительно уточняется режим трассировки. Это делается с помощью диалогового блока, активизируемого нажатием клавиши "т". Выбор левой кнопкой мыши пунктов этого блока включает либо выключает соответствующие установки на трассировку. Для обычного просмотра рекомендуется режим с выключенным пунктом "ручной выбор входа в подпроцесс"; включение этого пункта происходит в тех случаях, когда при решении задачи встречается обращение к чрезвычайно трудоемкой вспомогательной задаче, не выводимое на экран. Такое включение не обязательно выполнять с самого начала трассировки: обращение к диалоговому блоку выбора режима трассировки возможно на каждом шаге. Другие пункты этого диалогового блока таковы:

1) "Пропуск обращений к проверочным операторам" - обычно выключен. Его включение приведет к тому, что перестанут появляться сообщения об обращениях к процедурам проверки различных вспомогательных утверждений.

2) "Пропуск шагов удаления посылок и условий" - обычно включен. Его выключение приведет к тому, что будут явно указываться шаги удаления ненужных условий и посылок задачи.

3) "Пропуск изменений сопровождающих условий" - обычно включен. Его выключение приведет к тому, что будут отображаться изменения условий на область допустимых значений, сопровождающие основные шаги решения.

4) "Пропуск входа в любые подпроцессы" - обычно выключен. Сообщения об обращениях к наиболее существенным для решения задачи вспомогательным задачам и процедурам при его выключении автоматически выводятся на экран (хотя такие обращения означают не срабатывание приема, а только начало попытки его применения). Если этот пункт включен, то данные сообщения пропадут, а останутся лишь сообщения о фактически сработавших приемах.

5) "Просмотр шагов изменения комментариев" - обычно выключен. Изменения технических пометок - так называемых комментариев образуют достаточно интен-

сивный поток, вывод которого на экран существенно замедлит трассировку.

6) "Пропуск регистрации условий на о.д.з." - обычно включен. При его выключении более подробно будет отображен процесс первоначального ввода условий на область допустимых значений, сопровождающих условия и посылки задачи.

Чтобы перейти из режима семантической трассировки в просмотр текущей ситуации с помощью отладчика ЛОСа, достаточно нажать клавишу "ф". Тогда на экране появится текст текущего исполняемого фрагмента программы, вплоть до текущего оператора (пока не реализованного), который будет выделен малиновым цветом. Дальнейшие действия - согласно инструкции по отладчику ЛОСа (см. ниже). Для возвращения в просмотр текущего кадра семантической трассировки и продолжения ее достаточно нажать клавишу "р"либо клавишу "з".

8.2 Установка режимов технической трассировки перед запуском решения

Если нужно отладить прием, реализованный на ГЕНОЛО́Ге, то сначала следует создать либо найти в задачнике такую задачу, где он должен сработать. После этого можно создать установку на прерывание при попытке применения данного приема и запустить решение задачи до обнаружения такой попытки. Для этого (из просмотра задачи в задачнике) нажимается клавиша "г"или выбирается пункт "Генолог" меню выбора режима трассировки, расположенного в верхней части экрана. Тогда возникает некоторый пункт оглавления базы приемов ГЕНОЛОГа. В этом оглавлении находится нужный пункт, и внутри группы приемов этого пункта находится нужный прием (для этого служат клавиши "курсор вверх" - "курсор вниз"). Далее нажимается клавиша "Ctr-Enter", которая одновременно создает установку на прерывание и запускает решение задачи.

Если в процессе решения произойдет выход на начальный отрезок программы выбранного приема (именно, на используемый для указания этого начала фиктивный оператор "контрольприема(A_1 A_2 A_3)"; здесь A_1, A_2, A_3 - логические символы и термы, образующие стандартную техническую ссылку на прием), то включится отладчик ЛОСа, и на экране будет отображен фрагмент программы с выделенным малиновым цветом оператором, который должен будет выполняться непосредственно на следующем шаге (этот оператор следует за оператором "контрольприема($A_1A_2A_3$)"). При этом устанавливается пошаговый режим трассировки (см.ниже).

Если выбранный прием применяется при сканировании задачи либо при работе пакетного нормализатора (см. описание ГЕНОЛОГа), то можно установить прерывание при фактическом применении этого приема. Для этого вместо клавиши "Ctr-Enter"нажимается клавиша "Enter".

Если нужно проверить, применяется ли при решении задачи некоторый оператор либо операторное выражение, реализованные с помощью программы на ЛОСе, и проследить по шагам ход выполнения этой программы, то перед запуском решения задачи нажимается клавиша "Л". Тогда в левой нижней части экрана возникает надпись "Логический символ:" и появляется курсор текстового редактора. Выполняется набор необходимого логического символа и нажимается "Enter". Если было набрано слово, не являющееся названием логического символа, то оно пропадает с экрана и курсор возвращается в исходное положение - для повторного набора. Иначе - одновременно создается установка на прерывание при обращении к программе указанного логического символа и запускается процесс решения задачи. По достижении

первого оператора корневого фрагмента программы выбранного символа включается отладчик ЛОСа; сам этот оператор прорисован малиновым цветом и еще не выполнен. Для дальнейших действий автоматически устанавливается пошаговый режим трассировки.

Если нужно проследить моменты выхода к точкам программы, для которых созданы соответствующие концевые пункты оглавления программ, то перед запуском решения задачи нажимается клавиша "л". Тогда возникает некоторый пункт оглавления программ. Если выбрать нужный концевой пункт этого оглавления и нажать на нем клавишу "курсор вправо", то одновременно инициируется прерывание при выходе на соответствующую контрольную точку программы (напомним, что эта точка представляет собой фиктивный оператор "прием(N)"; N - номер контрольной точки внутри программы данного логического символа). Как и в предыдущих случаях, по достижении контрольной точки включается отладчик ЛОСа и устанавливается пошаговый режим трассировки.

При отладке бывает полезно применение счетчика шагов работы интерпретатора ЛОСа. Оно дает возможность при повторных запусках решения задачи идентифицировать (например, методом деления отрезка пополам) момент ошибочного действия. Это средство применяется, впрочем, достаточно редко, так как подавляющее большинство ошибок обнаруживаются по недопустимым типам входных данных операторов ЛОСа и немедленно выводятся на экран. Однако, в некоторых случаях появление заведомо ошибочных элементов структуры данных не выявляется средствами общего контроля, и анализ текущей информации не позволяет объяснить их происхождение. В этих случаях и используется счетчик шагов. При трассировке на уровне отладчика ЛОСа номер текущего шага работы интерпретатора выводится на экран в правом верхнем углу. Этот номер позволяет примерно определить диапазон, в котором следует искать момент появления ошибки. Если при повторном запуске решения задачи нажать клавишу "Ш"и ввести текстовым редактором в появившемся окне диалога (после надписи "Число операторов:") номер шага, соответствующего началу диапазона поиска ошибки, то по нажатии "Enter"будет одновременно установлено прерывание по достижении заданного шага и начато решение задачи. По достижении нужного шага включается отладчик ЛОСа и устанавливается пошаговый режим.

Кроме установок на автоматическое прерывание процесса решения, можно в любой момент прервать этот процесс вручную. Для этого достаточно нажать клавишу "Break". В результате появится текст текущего исполняемого фрагмента программы, в котором текущий (еще не выполненный) оператор выделен малиновым цветом. Для работы в отладчике ЛОСа устанавливается пошаговый режим. Заметим, что нажатие клавиши "Break"не позволяет входить в просмотр функционирования программ отладчика.

Наконец, самый простой (но сравнительно трудоемкий) способ вызвать прерывание работы программы и обращение к отладчику ЛОСа в нужной точке программы - вставить в этой точке оператор "трассировка(стоп 0)". Этот оператор вызывает немедленный вход в отладчик ЛОСа с пошаговым режимом, не выполняя никаких других действий (он всегда истинен). По завершении анализа текущего шага отладчиком и продолжении трассировки будет выполняться следующий за ним оператор. Единственное необходимое условие его корректной работы - наличие после него в программе хоть какого-либо другого оператора. Заметим, что помещение данного оператора внутри процедур интерфейса и даже процедур самого отладчика приводит к тому же эффекту, так что он делает возможной отладку с помощью отладчика

ЛОСа тех участков программы отладчика, в которых этот оператор не будет немедленно повторно выполняться при попытке обращения к отладчику.

8.3 Просмотр информации о моменте прерывания

Собственно отладчик ЛОСа представляет собой программу логического символа "прерывание", которая активизируется на тех шагах работы системы, для которых либо оказывается истинным заранее установленное условие прерывания, либо нажимается клавиша "Break", либо обнаруживается попытка реализации оператора ЛОСа для недопустимых входных данных. Эта программа позволяет просмотреть информацию о текущем состоянии системы и изменить режим трассировки, в том числе ввести новые установки на прерывание. При выходе из нее продолжается прерванное решение задачи. На момент входа в программу "прерывание"счетчик шагов интерпретатора ЛОСа отключается, так что работа с отладчиком никак не сказывается на соответствии номеров шагов действиям решателя. В процессе работы с отладчиком ЛОСа можно восстановить исходное изображение, возникшее на момент обращения к нему при прерывании, нажав клавишу "р"(кир.).

8.3.1 Просмотр программы

При входе в отладчик ЛОСа на экране прорисовывается некоторый фрагмент программы, в котором малиновым цветом выделен текущий (пока не выполненный) оператор. Этот фрагмент, вообще говоря, не является корневым; можно просмотреть все его надфрагменты в дереве программы текущего логического символа, используя клавиши "Home" (шаг по направлению к корню программы) и "End" (шаг от корня программы к фрагменту с текущим оператором). С указанными фрагментами связан общий набор значений программных переменных текущей ситуации, возникшей при выполнении текущей программы; эти значения сохраняются в некотором кадре глобального стэка интерпретатора ЛОСа. Заметим, что из режима трассировки возможен просмотр только линейной цепочки фрагментов, от корня программы до фрагмента с текущим оператором; выход на боковые ветви программы не предусмотрен.

При просмотре фрагментов программы выдается только начальный отрезок фрагмента до оператора, выделенного малиновым цветом (это либо текущий оператор, либо оператор перехода к подфрагменту). Чтобы просмотреть все операторы фрагмента, нажимается клавиша "ф". Чтобы убрать с экрана лишние (идущие после выделенного малиновым цветом) операторы, нажимается клавиша "о". В левой верхней части экрана размещен текст "Программа A", где A - заголовок программы (выделяется синим цветом). В правой верхней части экрана размещен текст "шаг № N", где N - номер текущего шага работы интерпретатора ЛОСа. Этот шаг не является абсолютным; он отсчитывается только с момента запуска решения извлеченной из задачника задачи, что делает его неизменным при повторных запусках и позволяет использовать для локализации событий, происходящих при решении. Под текстом фрагмента программы проведена горизонтальная черта, отделяющая область экрана, в которой можно получать дополнительную информацию о текущем шаге.

Кадр глобального стэка интерпретатора ЛОСа сохраняет информацию о текущих значениях программных переменных, возникающих при реализации программы некоторого оператора либо операторного выражения, либо при сканировании некото-

рой задачи. При обращении от программы к подпрограмме создается новый стэковый кадр, для сохранения значений программных переменных этой подпрограммы. Таким образом, возникает цепочка стэковых кадров, в которых сохраняются значения программных переменных как для текущей программы, так и для всех ее надпрограмм. Самая "верхняя"программа в этой цепочке - сканирование исходной фиктивной задачи на исследование; от нее имело место обращение к сканированию задачи, извлеченной из задачника; далее - к некоторой вспомогательной задаче либо оператору, и т.д. Возможен просмотр всей цепочки "надпрограмм" текущей реализуемой программы с помощью отладчкиа ЛОСа. Для этого служат клавиши "PageUp" и "PageDown". Первая из них переводит от просмотра программы к ее надпрограмме; вторая - от программы к подпрограмме (все это - вдоль линейной цепочки обращений, сложившейся на текущий момент). При переходах прорисовываются отрезки соответствующих фрагментов программы вплоть до того оператора, от которого имело место обращение к подпрограмме. На каждом уровне цепочки обращений возможен просмотр значений программных переменных данного уровня. Так как значения этих переменных при выполнении программы на ЛОСе часто сохраняются неизменными (значение однажды определенной переменной обычно меняется только при откатах), то становится возможен анализ значительной части "предыстории" текущей ситуации.

Использование пар клавиш "Page Up - Page Down"и "Home - End"делает просмотр текущей ситуации двумерным: можно варьировать фрагмент программы по цепочке переходов внутри программы одного логического символа, а также варьировать саму рассматриваемую программу по цепочке обращений от одной программы к другой.

При просмотре программы отладчиком, как и при просмотре ее редактором программ, возможен вход в режим просмотра подтермов операторов (многоцветная указка). Этот режим упрощает чтение и понимание сложных логических операторов, а также используется при установке прерываний (см. ниже). Для входа в режим просмотра подтермов операторов следует нажать клавишу "курсор вниз" - первый из операторов текущего фрагмента выделяется синим цветом. Далее клавиши "курсор вправо" - "курсор влево" позволяют выбрать нужный оператор фрагмента; клавиша "курсор вниз" - войти в просмотр операндов выделенной операции; клавиши "курсор вправо" - "курсор влево" - выбрать нужный операнд. Для возвращения от операнда к внешней операции и выхода из режима просмотра подтермов нажимается клавиша "курсор вверх".

Для получения информации о логическом символе, выделенном при просмотре подтермов оператора, нажимается клавиша F3.

В режиме просмотра фрагмента программы (без выделения подтермов операторов) возможен обычный доступ к справочной информации о логических символах: нажатие клавиши F3 приводит к выдаче информации о логическом символе - заголовке текущей программы; нажатие клавиши F2 - к появлению курсора текстового редактора, с помощью которого набирается название логического символа, о котором требуется получить информацию.

8.3.2 Просмотр цепи задач

Для входа в просмотр цепи задач (текущей задачи и всех ее надзадач) нажимается клавиша "з". При просмотре применяются клавиши "курсор вверх" - "курсор вниз"; "PageUp" - "PageDown"; "Ctr-PageUp" (начало цепи задач); "Ctr - PageDown" (конец цепи задач, т.е. текущая задача); "Ctr - курсор вверх" (к началу предыдущей зада-

чи); "Ctr - курсор вниз" (к началу следующей задачи). Заметим, что хотя внешне просмотр цепи задач из отладчика ЛОСа ничем не отличается от просмотра ее при трассировке по шагам решения задачи, нажатие клавиши "Enter" в первом случае не приводит к продолжению процесса решения задачи.

Для возвращения в основной интерфейс отладчика ЛОСа из интерфейса просмотра цепи задач нажимается клавиша "ф".

8.3.3 Просмотр значений программных переменных

Для просмотра различной информации о текущей ситуации используется область экрана, расположенная под горизонтальной чертой, отделяющей текст фрагмента программы. Если эта область недостаточна, то можно вообще убрать с экрана текст фрагмента программы, нажав клавишу "Delete" (этот текст восстанавливается нажатием клавиш "о" - до текущего оператора, либо "ф" - полный текст фрагмента). Если нужно убрать текст программы, сохранив информацию, размещенную в нижней части экрана, то вместо "Delete" нажимается F5. В нижней части экрана данные выдаются в режиме автоматической прокрутки вверх (обратная прокрутка невозможна, и для повторного просмотра удаленных с экрана объектов их следует востребовать заново). Возможна ручная прокрутка вверх содержимого нижней части экрана (при сохраняющемся в верхней части тексте фрагмента программы) с помощью клавиши "курсор вниз".

Для просмотра значения программной переменной "хi"следует нажать клавишу "х" (кириллица) - появятся буква "х" и курсор текстового редактора, и далее набрать номер i в обычной десятичной записи.

Если значением программной переменной "хi"является терм, то он будет выдан в стандартной либо в скобочной записи в зависимости от ранее установленного режима просмотра термов. Клавиша "с" переводит в скобочный режим, клавиша "м" (кир.) - в режим стандартной математической записи.

Если значением переменной "хi" служит набор, не являющийся термом, то будут последовательно выданы все его разряды. Здесь логические символы и символы переменных указываются явно (последние - в виде, согласованном либо со скобочной записью, т.е как буква "х"с номером, либо в виде, согласованном со стандартной математической записью - как латинская буква, быть может, с индексом). Элемент набора, представляющий собой терм, обозначается посредством записи "тj", где j - номер, закрепляемый за данным термом на период просмотра элементов текущей структуры данных (эти номера сбрасываются при смене текущего просматриваемого фрагмента клавишами PageUp,PageDown, Home,End). Элемент набора, представляющий собой набор, отличный от терма, обозначается посредством записи "нj", где j - номер, закрепляемый за указанным набором на период просмотра структуры данных. Элемент набора, представляющий собой вхождение в некоторый набор либо в терм, обозначается посредством "вj", где j - номер, закрепляемый за данным вхождением на период просмотра. Нумерация объектов всех указанных типов - общая.

Если значением переменной "хi"служит вхождение в терм либо в набор, то будет прорисован, соответственно, этот терм либо набор, в котором подтерм либо разряд, находящийся по рассматриваемому вхождению, выделен синим цветом.

Объекты "нj", "тj", "вj", введенные при просмотре набора, сами могут быть отображены на экране в одном из указанных выше видов. Для этого следует лишь нажать, соответственно, клавишу "н", "т" либо "в" и после этого набрать номер j объекта (набор номера завершается нажатием клавиши "Enter").

Для большей наглядности можно изображать набор в виде последовательности строк, каждая из которых в указанном выше формате представляет один разряд набора. При этом предусмотрена выдача лишь тех элементов набора, которые не являются логическими символами либо переменными (символьные элементы доступны при обычном просмотре набора, и здесь они пропускаются). Если рассматриваемый набор является значением переменной "хi", то нажимается клавиша "Х" (кир.) и вводится номер i; если же набор обозначен как "нi", то нажимается клавиша "Н" (кир.) и вводится номер i. Возможно отображение в указанном виде лишь тех элементов набора, которые начинаются с заданного логического символа S. Для этого после набора номера i в окне текстового редактора помещается название символа S (отделенное от номера пробелом).

Если требуется одновременно выдать на экран в скобочной записи все элементы набора термов "нi", то нажимается клавиша "а"(кир.), после чего вводится текстовым редактором номер i.

По-видимому, для просмотра сложных структур данных наиболее удобным является использование "сквозного режима". В этом режиме используется все пространство экрана (текст фрагмента программы временно удаляется); каждый разряд набора прорисовывается, после указания его номера, на отдельной строке либо группе строк, причем возможна обычная прокрутка (клавиши "курсор вверх-вниз" и "PageUp - PageDown"). Синим цветом выделяется номер текущего разряда набора (смена этого номера - клавишами "курсор вверх - вниз"). Если текущий разряд сам является набором, то для его просмотра нажимается клавиша "курсор вправо", после чего он прорисовывается на экране в указанном поразрядном виде. Таким образом можно дойти до атомарных объектов, не являющихся наборами. Возвращение от элемента к внешнему набору, а также выход из указанного режима обеспечиваются нажатиями клавиши "курсор влево". Номера атомарных (не являющихся наборами) разрядов завершаются круглой скобкой; номера разрядов, являющихся наборами - точкой. При отображении разряда, являющегося набором, используются обычные сокращения: "т" означает терм, "н" - набор, "в" - вхождение. Эти сокращения, однако, не сопровождаются номерами, излишними при возможности выбора объекта для просмотра с помощью курсора. Чтобы просмотреть в сквозном режиме набор, являющийся значением переменной "хi", нажимается клавиша "К"(кир.) и вводится номер i. Если просматривается набор "нi", то нажимается "к"(кир.) и вводится i.

Если задача сопровождается чертежом, то возможна его прорисовка в отладчике ЛОСа. Нажатие клавиши "ч"приводит к его прорисовке в верхней части экрана; клавиши "Ч к прорисовке чертежа под ранее выведенной в нижней части экрана информацией. После вывода чертежа возможен обычный просмотр элементов текущей структуры данных, размещаемых под чертежом.

Если используются как стандартная математическая, так и скобочная записи термов, то для установления связи между обозначениями переменных в этих способах записи используется переходная таблица (она перечисляет пары "обозначение переменной задачи в стандартной записи - обозначение этой переменной в скобочной записи"). Для вызова на экран переходной таблицы нажимается клавиша "п".

8.3.4 Сообщение об ошибке в программе

Если интерпретатор ЛОСа обнаруживает попытку реализации некоторого оператора с недопустимыми для него типами входных данных, то он инициирует вход в отладчик ЛОСа и выдачу сообщения об ошибке: в верхней левой части пустого экрана

возникает сообщение "Некорректное обращение к оператору". При входе в просмотр текущего фрагмента программы (нажатие клавиши "ф") будет прорисована часть этого фрагмента, предшествующая текущему оператору (выделенному малиновым цветом). Анализ окрестности этого оператора обычно позволяет выявить описку в программе. Возможны также обращения к отладчику ЛОСа при обнаружении интерпретатором переполнения различных ресурсов. В этих случаях выдаются следующие сообщения об ошибке:

1) "Переполнение глобального стэка" - означает, что имеется чрезмерно длинная цепочка обращений от программы к подпрограмме, так что суммарный объем кадров глобального стэка, описывающих значения программных переменных всех этих программ, превысил предельно допустимую величину. Часто это происходит при "зацикливании" в рекурсивных обращениях.

2) "Переполнение буфера текстов" - означает, что предпринята попытка создать настолько большой терм или набор, что он превысил предельно допустимую величину.

3) "Переполнение зоны задач" - означает, что вся зона задач заполнена объектами, введенными решателем, и места для дальнейшей его работы не хватает. Так как величина зоны задач достаточно велика, и обычный (нормальный) режим работы системы - при практически пустой зоне задач, появление этого сообщения обычно связано с каким-либо зацикливанием программы. Появляется это сообщение чрезвычайно редко.

4) "Исчерпание переменных либо переполнение при арифметическом действии" - обычно означает, что ведено слишком много символов переменных и запрос на ввод нового такого символа не может быть удовлетворен.

5) "Превышение допустимого сброса" - означает попытку применения оператора сброса, выводящую за рамки действия текущей программы (число указанных для сброса перечисляющих операторов в рамках программы больше фактически имеющегося).

Кроме перечисленных сообщений, может появиться сообщение "Нет программы логического символа S" - если предпринята попытка обратиться к выполнению отсутствующей программы этого символа.

8.3.5 Выход в базу приемов, реализованных на ГЕНОЛОГе

Если выход в отладчик ЛОСа произошел при семантической трассировке после срабатывания некоторого приема, то нажатие клавиши "б"приводит к просмотру описания примененного приема на ГЕНОЛОГе - если был применен прием, запрограммированный на ГЕНОЛОГе, либо к просмотру концевого пункта оглавления программ ЛОСа, - если прием запрограммирован непосредственно на ЛОСе, и данному концевому пункту соответствует последняя пройденная перед срабатыванием контрольная точка "прием(N)". При технической трассировке, если просматривается некоторый фрагмент программы (текущий либо некоторый его надфрагмент, достижимый из текущего с помощью клавиши "PageUp"), то нажатие клавиши "б" также приводит к просмотру описания соответствующего этому фрагменту приема ГЕНОЛОГа либо концевого пункта оглавления программ ЛОСа - однако, лишь при условии, что текущий оператор рассматриваемого фрагмента расположен после оператора "контрольприема(...)", указывающего на прием ГЕНОЛОГа, либо оператора "прием(N)", указывающего на концевой пункт оглавления ЛОСа.

Выйти в оглавление ГЕНОЛОГа из отладчика ЛОСа можно и безотносительно

к срабатыванию текущего приема. Для это следует нажать клавишу "г", после чего появляется некоторый пункт оглавления базы приемов. По этом оглавлению можно перемещаться без каких-либо ограничений, входя в просмотр любых приемов ГЕНОЛОГа - так же, как и при просмотре этих приемов из главного меню. Заблокированы лишь те операции просмотра приемов, которые приводят к изменению приемов.

Если в оглавлении базы приемов был выбран некоторый концевой пункт и произошел вход в просмотр приема, закрепленного за данным пунктом (смена таких приемов, если их несколько, обеспечивается клавишами "курсор вверх - курсор вниз"), то возможна установка прерывания при попытке применить данный прием. Для ее ввода достаточно нажать клавишу "Ctr-Enter". Автоматически будет возобновлен процесс решения задачи, и выход в отладчик произойдет при обращении к оператору "контрольприема(...)", с которого начинается программа данного приема. Здесь устанавливается режим пошаговой трассировки и прорисовывается текущий фрагмент программы с выделенным малиновым цветом оператором, непосредственно следующим за оператором "контрольприема(...)". Для выявления собственно момента применения данного приема далее можно установить прерывание при выполнении завершающей части его программы (выбрав один из заключительных ее операторов, например, оператор "пересмотр").

8.3.6 Техническая трассировка

При отладке программы обычно используются такие режимы ее выполнения (называемые технической трассировкой), при которых процесс дробится на отдельные небольшие фрагменты, после каждого из которых осуществляется выход в отладчик ЛОСа. Принцип дробления можно либо задать один раз на весь период трассировки, переходя после этого к очередному шагу нажатием клавиши "Enter", либо каждый раз устанавливать очередное прерывание процесса специально.

Перечислим используемые в отладчике ЛОСа режимы технической трассировки и способы установки прерываний процесса. Сразу заметим, что для отмены любых установок на прерывания служит клавиша "0". После нажатия на нее, если не создать новых установок на прерывания и нажать "Enter", то решение задачи будет продолжено без выдачи каких-либо пояснений на экране вплоть до завершения (ответа либо отказа).

1) Трассировка с самым мелким дроблением процесса - так называемая пошаговая трассировка. Для установки этого режима из отладчика ЛОСа достаточно нажать клавишу "1"(кроме того, пошаговый режим устанавливается автоматически после целого ряда прерываний, выводящих в отладчик). После этого каждое нажатие клавиши "Enter" будет вызывать выполнение единственного шага - реализацию оператора (корневого либо расположенного внутри некоторого составного оператора) либо вычисление невырожденного операторного выражения. При пошаговом режиме трассировки можно входить в просмотр шагов выполнения составных операторов (дизъюнкций, конъюнкций, кванторов и т.п.).

2) Следующий режим - пооператорная трассировка. В этом режиме за один шаг выполняется один корневой оператор программы - как простой, так и составной. Для установки режима следует выбрать, используя клавиши "PageUp - PageDown", ту программу, в которой предполагается выполнять трассировку, и нажать клавишу "2". Будут отслеживаться только корневые операторы, которые выполняются в "кадре" этой программы, без выхода в выполнение подоператоров и вспомогательных задач. По окончании выполнения программы трассировка должна быть переуста-

новлена - иначе следующее прерывание произойдет в каком-то достаточно случайном месте. Эта трассировка при отладке является наиболее употребительной.

3) Иногда бывает нужно отслеживать моменты обращения к программе заданного логического символа A. Чтобы установить прерывание при обращении к этой программе, следует сначала нажать клавишу "л" - в результате появится курсор текстового редактора, - а затем набрать текстовым редактором название символа A. После этого, при нажатии "Enter", установка оказывается введенной, хотя продолжение процесса реализации программы еще не включается. При запуске программы (снова нажатием "Enter") отладчик обеспечит выход в просмотр первого оператора начального фрагмента программы символа A, как только произойдет обращение к этой программе. Установка на данное прерывание сохраняется до установки другого прерывания либо отмены прерываний, так что последовательные нажатия "Enter" позволят проследить цепочку указанных обращений.

4) При просмотре отладчиком некоторого фрагмента программы (текущего фрагмента либо его надфрагмента, достигнутого при помощи "Home" либо "PageUp") возможен ввод установки на прерывание при переходе к его подфрагменту. Для этого следует войти в режим просмотра операторов фрагмента (нажатием клавиши "курсор вниз"; выбрать требуемый оператор перехода к подфрагменту (клавиши "курсор вправо - курсор влево") и нажать клавишу "Ctr-Enter". Нажатие этой клавиши автоматически запускает процесс продолжения выполнения программы; при выходе на указанный подфрагмент произойдет прерывание. Заметим, что прерывание будет иметь место, даже если выход на подфрагмент произошел не из того "кадра" реализации программы, где была введена установка на прерывание, а из какого-либо совсем другого "кадра" реализации той же программы. Если требуется, чтобы прерывание произошло лишь при условии, что обращение к подфрагменту имело место в текущем "кадре"реализации программы, то после выбора оператора перехода нажимается "Enter". В этом случае завершение выполнения программы без выхода на выбранный ее подфрагмент приведет к прерыванию по ее завершении.

5) При просмотре отладчиком некоторого фрагмента программы (текущего фрагмента либо какого-либо его надфрагмента, достигнутого при помощи клавиш "Home" либо "PageUp") возможен ввод установки на прерывание перед выполнением какого-либо оператора этого фрагмента (не обязательно корневого; возможна установка прерывания перед выполнением подоператора составного оператора). Для этого следует войти в режим просмотра операторов фрагмента ("курсор вниз"); выбрать требуемый оператор (сначала клавишами "курсор вправо - курсор влево" выйти на нужный терм фрагмента, а при необходимости использовать клавишу "курсор вниз" и указанные выше клавиши для выделения подтерма) и нажать "Ctr-Enter". При этом автоматически запускается продолжение действий по программе, и при выходе на указанный оператор (непосредственно перед его реализацией) происходит прерывание. Заметим, что прерывание произойдет, даже если "кадр" реализации программы, в котором было установлено прерывание, отличается от "кадра" реализации этой же программы, в котором имело место обращение к оператору. Чтобы ввести установку на прерывания только в том же самом "кадре" реализации программы, следует нажать "Enter".

Использование установок на прерывания из пунктов 3,4,5 позволяет при отладке свободно перемещаться по программам различных логических символов и анализировать ситуации, возникающие на моменты выхода в различные их точки.

6) Чтобы использовать при отладке счетчик шагов работы интерпретатора ЛОСа, можно установить прерывание по достижении заданного числа шагов. Для этого

нажимается клавиша "ш" - появляется курсор текстового редактора, - и набирается номер нужного шага. При наборе номера следует учитывать расположенный в правом верхнем углу экрана (если просматривается фрагмент программы) номер текущего шага. Как уже говорилось выше, при обращениях к отладчику счетчик шагов отключается, так что процесс отладки не изменяет выраженных в числе шагов интерпретатора "координат" событий, происходящих при решении задачи. После набора номера требуемого шага (как обычно, завершаемого нажатием "Enter") установка на прерывание введена; для продолжения цикла выполнения программы еще раз нажимается "Enter".

Если решается задача, извлеченная из задачника, то можо сохранить номер текущего шага и при повторных запусках ее решения выходить в отладчик по достижении шага с данным номером. Для сохранения номера следует нажать клавишу "щ". Чтобы повторно запустить задачу с прерыванием по достижении данного шага, следует нажать клавишу "а" при просмотре условия задачи в задачнике - это нажатие одновременно запускает решение и вводит установку на прерывание.

7) Если произошло прерывание с выходом в отладчик ЛОСа, отличное от обычного шага семантической трассировки, то для возвращения в режим семантической трассировки следует нажать клавишу "пробел", после чего нажать "Enter". Прерывание произойдет при первом же применении приема - в режиме сканирования задачи либо при реализации пакетного оператора ГЕНОЛОГа (только для особо крупных операторов, выделяемых специальной опцией в своем описании). Если вместо клавиши "пробел" нажать клавишу "й", то прерывание произойдет при первом же применении приема на уровне сканирования текущей задачи - выполнение пакетных операторов при этом игнорируется.

8) Если нужно ввести прерывание при выходе из текущего "кадра"реализации программы (выбираемого с помощью "PageUp, PageDown"), то нажимаются клавиши "4" и "Enter". При ликвидации этого кадра и возвращении во внешнюю программу произойдет выход в отладчик и установится пошаговый режим трассировки.

9) Для ввода прерывания при изменении оператором "замена"в текущем кадре реализации программы значения переменной "хi", если этим значением служит терм, нажимается клавиша "7" - появляются буква "з" и курсор текстового редактора, - и далее набирается символьный (!) номер i. При каждом изменении указанной переменной будет возникать текст "значение переменной хi: A заменяется на B".

10) Если требуется установить прерывание при изменении разрядов набора, обозначенного при просмотре элементов текущей структуры данных посредством "нi", то нажимается клавиша "и" - появляется курсор текстового редактора, - и вводится номер i в обычном формате десятичного числа. Прерывание произойдет непосредственно перед выполнением оператора, изменяющего указанный набор. Если представляющий интерес набор является значением программной переменной "хi", для установки прерывания при изменении его разрядов нажимается клавиша "И" и вводится номер i.

11) Из отладчика можно выйти в оглавление программ и установить прерывание по достижении заданной в этом оглавлении контрольной точки. Для этого сначала нажимается "Ctr-г" - появляется некоторое меню оглавления программ. После выбора в оглавлении нужного концевого пункта, следует нажать на нем клавишу "курсор вправо". Это приведет к установке прерывания по выходу на соответствующую контрольную точку "прием(N)" и возобновлению процесса решения задачи.

12) Прерывание перед выполнением заданного оператора фрагмента программы (см. пункт 5) можно сопровождать списком дополнительных условий, а также

обеспечивать автоматическую выдачу на экран при таком прерывании значений заданных программных переменных. Для входа в просмотр и редактирование списка дополнительных условий следует нажать при просмотре фрагмента программы клавишу "у" (кир.). При этом включается редактор списков, отображающий ранее введенные условия. Окно его располагается непосредственно под горизонтальной чертой, завершающей текст фрагмента программы. Список условий разбивается на группы; первым элементом каждой группы служит некоторый ключевой терм (см.ниже), после которого размещаются сопровождающие его информационные элементы. Используются ключевые термы следующих типов:

а) "терм(A)". A есть программное выражение, определяющее объект для сравнения. Следующий элемент списка есть терм, который должен совпадать с этим объектом.

б) "заголовок(хi A)". Значением программной переменной "хi" является логический символ A либо набор или терм, начинающийся с логического символа A.

в) "входит(A хi)". Значением переменной "хi"является терм либо набор, содержащий логический символ либо символ переменной A.

г) "корень". Текущий терм задачи должен совпадать с следующим элементом данного списка установок.

д) "См(A)". A - программная переменная "хi"либо выражение "буква(хi)". Если значением A служит терм либо логический символ, то при прерывании предпринимается выдача этого значения на экран.

В редакторе списков используется следующий интерфейс. Для набор очередного элемента списка формульным редактором служит клавиша "Enter"; для набора элемента текстовым редактором - клавиша "т"(кир.). Для изменения текущего элемента формульным редактором нажимается "Ctr-ф"; для изменения его текстовым редактором - "Ctr-т". Завершается просмотр и редактирование списка нажатием клавиши "курсор влево". После этого, как обычно вводится установка на прерывание при обращении к некоторому оператору текущего фрагмента программы. Данное прерывание будет происходить всякий раз перед обращением к выбранному оператору, как только окажутся выполнены дополнительные ограничения согласно списку. Для сброса списка дополнительных установок на прерывание нажимается клавиша "Ctr-у"(кир.).

13) В заключение отметим еще одну исключительно важную возможность, предоставляемую отладчиком ЛОСа. При автоматическом решении серии задач из задачника используется так называемый режим "внутреннего перезапуска" - после каждой задачи интерпретатор повторно инициирует выполнение программы логической системы, обнулив все ее регистры и массивы в оперативной памяти. Решение серии задач невозможно отменить, даже используя предоставляемые операционной системой возможности по экстренному выходу из программы - при повторном запуске оно возобновится с той точки, где было прервано. Единственным способом отмены этого режима служит выход в отладчик ЛОСа с помощью клавиши "Break" и нажатие в отладчике клавиши "Ctr-з", которое, собственно, и отменяет серийное решение задач. Далее можно вернуться в главное меню, например, нажатием клавиши "Esc" (возвращение в главное меню без нажатия "Ctr-з" просто вызовет переход к следующей задаче серии).

8.4 Тестирование программы оператора, операторного выражения либо справочника

Для тестирования программы нового оператора, операторного выражения либо справочника лучше всего воспользоваться реальным контекстом, в котором она должна применяться - приемом решения задачи, либо приемом пакетного оператора, либо блоком интерфейса системы, и т.п. Однако, можно обратиться к этому оператору и непосредственно. Для этого служит программа логического символа "пуск", которую можно запускать из просмотра программы в редакторе программ ЛОСа нажатием клавиши "Ctr-Enter". Эта программа имеет единственную входную переменную x1, значением которой служит исходная задача. Войдя в редактирование программы символа "пуск", нужно набрать обращение к тестируемому оператору (программному выражению, справочнику). Сначала программируется построение входных объектов для тестируемой программе, затем размещается само обращение, после которого нужно поместить хотя бы один оператор - чтобы до выхода из программы "пуск" иметь возможность просмотра значений выходных переменных, полученных после обращения. Для удобства создания входных объектов и просмотра результатов обращения, если необходимо, в программу "пуск" можно включить упоминавшиеся выше вспомогательные операторы интерфейса. По завершении редактирования программы "пуск" следует нажать "Ctr-Enter". Тогда инициализируется работа только что набранной программы "пуск", причем сразу же включается отладчик ЛОСа с режимом пошаговой трассировки. Далее с помощью отладчика ЛОСа можно предпринять анализ выполнения тестируемой программы. По завершении программы "пуск" происходит возвращение в главное меню.

Глава 9

Примеры и упражения по программированию на ЛОСе

9.1 Примеры программ приемов, применяемых при сканировании задачи

ЛОС создавался для программирования приемов решения задач, и естественно продемонстрировать прежде всего несколько примеров таких программ. Хотя после появления ГЕНОЛОГа практически все приемы записываются на нем, эти приемы затем компилируются в программы на ЛОСе, которые и работают при решении задач. Поэтому представление о том, как устроена типичная программа приема на ЛОСе, весьма полезно при отладке.

Начнем с приема, выполняющего упрощение выражений по формуле $a(\sin x)^2 + a(\cos x)^2 = a$. Первый шаг, который следует сделать при написании программы приема - выбрать логический символ, с рассмотрения которого при сканировании задачи начинается выполнение этой программы. При выборе такого символа следует учитывать, что не каждый символ, встречающийся в записи теоремы, на которой основан прием, обязательно будет встречаться в преобразуемом терме задачи. Прием должен иногда усматривать и "замаскированные" возможности срабатывания, используя те или иные простые преобразования для искусственного представления встретившегося в задаче терма в том виде, какой указан в теореме.

В нашем случае выбор, например, символа "умножение" нежелателен, так как это умножение (на a) при $a = 1$ либо $a = -1$ будет отсутствовать в задаче. С другой стороны, выбор символа "синус" является, по-видимому, вполне приемлемым. При выборе логического символа (будем называть его символом привязки) следует отдавать предпочтение из нескольких вариантов тому символу, который реже встречается в задачах - при этом меньшее время будет затрачиваться на попытки применения приема в ситуациях, когда его срабатывание заведомо невозможно.

Следующий важный шаг - выбор значения либо нескольких значений текущего уровня сканирования, при которых будет предприниматься попытка применения приема. Здесь следует исходить из соображений о приоритетности попыток применения приемов и учитывать ранее введенные уровни срабатывания конкурирующих приемов (находя эти приемы, например, по оглавлению базы приемов ГЕНОЛОГа либо анализируя процессы решения задач, в которых желательно либо нежелательно срабатывание нового приема).

Несколько различных уровней срабатывания приема используются в следующих

случаях:

а) Выделяются несколько качественно различных ситуаций, с различными приоритетами срабатывания; для каждой из этих ситуаций (распознаваемых дополнительно внутри программы приема) предусматривается свой уровень срабатывания;

б) Вводится дополнительный контроль возможных изменений в задаче, при которых ранее невозможное либо нежелательное применение приема (заблокированное при выходе на программу с малым значением текущего уровня) впоследствии (при выходе на программу с бóльшим значением текущего уровня) становится возможным и целесообразным. Например, если для срабатывания приема необходимо наличие какой-либо дополнительной посылки задачи, отсутствовавшей изначально (при малом значении текущего уровня), но выведенной впоследствии, то дополнительная попытка применения приема на большем текущем уровне приведет к необходимому срабатыванию. Альтернативой такому средству контроля является уменьшение весов термов, содержащих символ привязки данного приема, как только текущие преобразования задачи (например, вывод следствий) делают вероятной целесообразность его применения.

В нашем примере выберем уровень срабатывания приема равным 1 - на этом уровне обычно применяются преобразования типа "общей стандартизации" термов задачи.

Следующий шаг при программировании на ЛОСе приема сканирования задачи - поиск в программе логического символа привязки той точки, в которой следует вставить оператор перехода ("ветвь N" либо "иначе N") для ветви, содержащей собственно новую программу. Здесь следует попытаться использовать как можно большее число операторов, уже введенных для программ других приемов, и выбрать точку ответвления как можно "глубже". Впрочем, выбор этой точки должен происходить таким образом, чтобы одна из программ не "портила" значений программных переменных, используемых другой программой - как правило, весьма необременительное ограничение, так как приемы на ЛОСе крайне редко прибегают к "переопределению" значений ранее определенных переменных. При вставке оператора "ветвь N"следует учитывать, что этот оператор является перечисляющим, и если после него в программе находились операторы сброса заданного числа внешних перечислений ("обрыв", "сброс(n)"), в "интервал перехода"которых попадает точка вставки, то необходимо увеличить на 1 константу сброса этих операторов. В приемах сканирования задач операторы сброса практически не используются, и здесь указанная ситуация не возникает.

Мы опустим здесь шаг поиска в уже существующей программе символа "синус" точки ответвления для нашего приема и приведем (в учебных целях) программу приема начиная с самого первого ее оператора.

Стандартное начало программы приема сканирования задачи имеет вид "решить стандарт(x2)". Оператор "решить"является истинным , если обращение к программе символа произошло из сканирования задачи; оператор "стандарт(x2)"проверяет, что вхождение x2 этого символа сопровождается открывающей скобкой (этот оператор используется только в тех случаях, когда рассматриваемый логический символ является символом операции либо предиката; вхождение такого символа без открывающей скобки может использоваться в построениях структурного уровня, описывающих вид термов).

Далее размещается фиксирующий уровень обращения к приему оператор "уровень(1)".

После выполнения указанных выше трех предварительных операторов начинается основная часть приема - идентификация ситуации, в которой возможно его применение. Мы имеем на текущий момент следующие определенные значения программных переменных: x1 - текущая задача; x2 - вхождение в некоторый ее терм логического символа "синус"; x3 - вхождение этого терма (называем его текущим) в соответствующий список задачи (в список условий либо список посылок либо в саму задачу); x4 - равно 0, если текущий терм является посылкой, и 1, если он является условием; x5 - значение текущего уровня сканирования (в нашем случае равно 1). Значение переменной x6 пока не определено. Начнем идентификацию указанной в теореме ситуации с того, что перейдем от x2 к вхождению x6 внешней для синуса операции: "операнд(x6 x2)". Эта операция должна быть степенью: "символ(x6 степень)". Показатель данной степени равен 2: "второйсимвол(x6 2)".

Далее возникает необходимость идентифицировать коэффициент a перед квадратом синуса. Снова рассматриваем внешнюю для x6 операцию x7: "операнд(x7 x6)". Эта операция, однако, не обязательно должна быть умножением. Приходится использовать выражение "вариант", определяющее значение x8 коэффициента a в зависимости от символа по вхождению x7: "равно(x8 вариант(символ(x7 умножение) исключениеоперанда(x7 x6)набор(1)))". Заметим, что если $a = 1$, то x8 имеет своим значением не сам символ 1, а одноэлементный набор из 1 (т.е. однобуквенный терм). Если же по вхождению x7 расположен символ "умножение", то a получается из данного произведения вычеркиванием сомножителя - квадрата синуса.

В действительности только что определенное значение x8 - это еще не коэффициент a, так как необходимо учесть знак слагаемого с квадратом синуса. Поэтому рассматриваем вхождение x9 внешней операции при умножении. Здесь, однако, удобно применить оператор "альтернатива": "альтернатива(символ(x7 умножение)операнд(x9 x7)равно(x9 x7))". Если по вхождению x7 не была расположена операция "умножение", то x7 уже является тем вхождением, на котором может быть расположен "минус". Наконец, определяем значение x10 коэффициента a с учетом знака: "равно(x10 вариант(символ(x9 минус)запись(минус x8)x8))".

Теперь переходим к внешней операции, которая должна оказаться суммой: "операнд(x11 x9) символ(x11 плюс)". Далее просматриваем слагаемые найденной суммы, отличные от слагаемого с квадратом синуса: "операнд(x11 x12) не(равно(x12 x9))". x12 - вхождение, которое следует идентифицировать с $a(\cos x)^2$. Прежде всего убеждаемся в том, что знак этого слагаемого совпадает со знаком исходного слагаемого: "альтернатива(символ(x9 минус)символ(x12 минус)не(символ(x12 минус)))". Здесь можно было бы использовать более экономное "символ(x12 буква(x9))", так как не просто знаки, но и заголовки двух слагаемых должны совпадать.

Далее определяем вхождение x13 второго слагаемого с отброшенным знаком "минус"(если он есть): "равно(x13 вариант(символ(x12 минус)первыйоперанд(x12)x12))". Определяем вхождение x14 во второе слагаемое, которое будет идентифицироваться с квадратом косинуса: "альтернатива(символ(x13 умножение)операнд(x13 x14)равно(x14 x13))". Проводим идентификацию для квадрата косинуса: "символ(x14 степень) первыйсимвол(x14 косинус) второйсимвол(x14 2) равныетермы(первыйоперанд (первыйоперанд(x14)) первыйоперанд(x2))". Последний оператор здесь проверяет совпадение аргументов синуса и косинуса. Проверяем совпадение коэффициентов (с отброшенными знаками) при квадратах синуса и косинуса: "равно(x8 вариант(символ(x13 умножение)исключениеоперанда (x13 x14)набор(1)))".

Теперь переходим к программированию завершающей части приема, выполняющей необходимую замену. Прежде всего, заметим, что необходимо найти все слага-

емые, отличные от двух выделенных, и учесть их в заменяющем терме, на который будет заменена сумма, расположенная по вхождению x11. Список этих слагаемых равен "выписка(x15 и(операнд(x11 x15)не(равно(x15 x9))не(равно(x15 x12)))подтерм (x15))". Соответственно, определяем заменяющий терм x15: "равно(x15 унисборка(плюс префикс(x10 выписка(x16 и(операнд(x11 x16)не(равно(x16 x9))не(равно(x16 x12)))подтерм(x16)))))". Операторное выражение "унисборка"применяется здесь из-за того, что число слагаемых в результирующем терме может оказаться равным 1, и тогда внешняя операция "плюс"будет опущена. Возможно альтернативное определение заменяющего терма при помощи операторного выражения "склейкаоперандов": "равно(x15 склейкаоперандов(x11 x9 x12 x10))".

Наконец, собственно оператор, выполняющий замену суммы по вхождению x11 на выражение x15: "заменавхождения(x11 x3 x4 x1 x15 пустоеслово)". Далее располагаем завершающий программу оператор "пересмотр", инициирующий повторное рассмотрение задачи с текущим уровнем, равным 0.

Основная часть приведенной программы выполняла усмотрение ситуации для применения заданного тождества - идентификацию его переменных с термами задачи. Программы такого типа, а их подавляющее большинство, создаются в решателе с помощью компилятора ГЕНОЛОГа на основе теорем предметной области, снабженных определенной технической разметкой. Непосредственно на ЛОСе обычно реализуются лишь приемы общелогического характера, не укладывающиеся в рамки стандартных алгоритмических построений ГЕНОЛОГа. В оставшейся части раздела мы рассмотрим несколько простых примеров таких программ, оставляя рассмотрение программ приемов, основанных на теоремах, до разделов, посвященных ГЕНОЛО́Гу.

Первый пример - программа приема, исключающего переменную связывающей приставки квантора общности, для которой в антецедентах этого квантора указано ее явное выражение. Заменяемое утверждение здесь имеет вид

$$\forall_{x_1 \ldots x_n}(x_i = t \ \& \ A_1(x_i) \& \ldots \& \ A_m(x_i) \Rightarrow A_0(x_i)),$$

где выражение t не содержит переменной x_i; заменяющее - вид

$$\forall_{x_1 \ldots x_{i-1} x_{i+1} \ldots x_n}(A_1(t) \& \ldots \& A_m(t) \Rightarrow A_0(t))$$

.

В качестве символа привязки выбираем "длялюбого" (равенство также могло бы претендовать на эту роль, но квантор общности встречается в задачах реже).

Так как прием выполняет вполне естественную общелогическую стандартизацию, его можно применять уже на нулевом уровне. Поэтому первые операторы программы имеют вид "решить стандарт(x2) уровень(0)". Так как возможность применения приема никак не связана с измененем его внешнего контекста, то далее размещается оператор "новый", который блокирует попытки повторного применения к терму задачи данного приема, если текущий уровень сканирования, после некоторой паузы, вновь поднялся до значения веса этого терма, возвратив его из теневой области в поле зрения.

Далее определяем список x6 переменных, относящихся к кванторной приставке квантора, и проверяем, что все они различны: "равно(x6 связприставка(x2)) различны(x6)". Находим список x7 антецедентов квантора (утверждения слева от стрелки $\Rightarrow$) и консеквент x8 этого квантора (утверждение справа от стрелки): "равно(x7 выписка(x8 антецедент(x2 x8)подтерм(x8))) равно(x8 последнийтерм(x2))".

Чтобы проверить, имеется ли среди антецедентов равенство, определяющее значение переменной кванторной приставки, просматриваем элементы x9 списка антецедентов: "входит(x9 x7)".

Начинаем идентификацию антецедента x9 с равенством требуемого вида: "заголовок(x9 равно) операнд(левыйкрай(x9)x10) символ(x10 x11) входит(x11 x6)". Здесь x10 - вхождение того операнда равенства, который оказался переменной x11, входящей в список x6. Эта переменная идентифицирована с x_i. Чтобы определить другой операнд равенства, используем операторы: "операнд(левыйкрай(x9) x12) не(равно (x10 x12)) равно(x13 подтерм(x12))". Здесь мы применяем перечисляющий оператор "операнд", выдающий последовательно вхождения каждого из двух операндов, и далее отбираем то вхождение x12, которое отлично от вхождения x10 уже рассмотренного операнда. x13 - переписанное в виде отдельного терма выражение, идентифицированное с t. Используя оператор "Операнды", можно было бы выполнить идентификацию равенства несколько короче: "заголовок(x9 равно) Операнды(левыйкрай(x9)x10 x11) символ(x10 x12) входит(x12 x6) равно(x13 подтерм(x11))"(здесь x_i идентифицируется с x12, t - с x13). Далее мы продолжаем программу для первого варианта идентификации.

После того, как найдены x_i и t, определяем набор утверждений $A_1(t), \ldots, A_m(t)$. Так как внутри выражения t могли встречаться связанные переменные, то после подстановки его вместо x_i в какое-либо $A_j(x_i)$, также имевшее связанные переменные, может оказаться, что некоторое вхождение переменной в $A(t)$ имеет два вложенных друг в друга внешних квантора либо описателя по этой переменной. Такое двойное связывание одной и той же переменной может повлечь за собой логические ошибки, так как для ускорения ряда процедур решателя делается допущение о невозможности двойного связывания либо одновременного использования в общем контексте переменной как свободной и как связанной. Это допущение обеспечивается своевременной нормализацией обозначений, и в данном случае лучше сразу же переобозначить связанные переменные в $A_j(x_i)$, сделав их отличными от переменных выражения t. Например, для этого можно использовать операторное выражение "новыесвязки(T S)", значением которого служит результат переобозначения в терме T связанных переменных на переменные, не встречающиеся в термах списка S. Таким образом, для нахождения списка x14 утверждений $A_1(t), \ldots, A_m(t)$ имеем оператор "равно(x14 выписка(x15 x16 и(входит(x15 x7)не(равно(x15 x9))результподст (набор(x11)набор(x13)новыесвязки(x15 набор(x13)) x16)) x16))". Он просматривает антецеденты x15, отличные от уже выделенного равенства x9, и применяет к результату указанного выше переобозначения связанных переменных в x15 подстановку x13 (т.е t) вместо x11 (т.е. x_i).

Аналогичную подстановку применяем к консеквенту, получая в качестве значения переменной x15 утверждение $A_0(t)$: "результподст(набор(x11)набор(x13)новыесвязки(x8 набор(x13))x15)".

Далее нужно найти остаток x16 связывающей приставки квантора общности после исключения из него переменной x_i: "равно(x16 вычеркивание(x6 набор(x11)))".

Теперь остается построить заменяющее утверждение из найденных списка его антецедентов x14, консеквента x15 и новой связывающей приставки x16. Однако, здесь следует учитывать вырожденный случай: если исходный квантор имел лишь одну переменную в связывающей приставке, то заменяющее утверждение нужно строить без использования квантора общности, как дизъюнкцию консеквента и отрицаний всех антецедентов. Учитывая это, получаем следующий оператор для формирования заменяющего утверждения x17: "равно(x17 вариант(равно(x16 пустоеслово)

унисборка(или суффикс(выписка(x18 входит(x18 x14) отрицание(x18))x15))сборка (длялюбого конкатенация(x16 вариант(равно(x14 пустоеслово)набор(x15)конкатенация(если x14 то набор(x15)))))))". Если связывающая приставка x16 пустая, то для получения x17 соединяем связкой "или"отрицания утверждений набора x14, к которым в конце добавляем x15. Если она непустая, то в случае пустого списка новых антецедентов x14 сразу после связывающей приставки помещаем консеквент x15, иначе - сначала вставляется логический символ "если", затем идут антецеденты, затем логический символ "то", и в конце - консеквент x15. В конце программы приема размещаются операторы "заменавхождения(x2 x3 x4 x1 x17 пустоеслово)" (фактическая замена утверждений) и "пересмотр".

Второй пример простого общелогического приема, реализованного на ЛОСе, связан с решением задачи на доказательство Z, условие которой имеет вид кванторной импликации:

$$\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_m \Rightarrow A_0).$$

Для решения такой задачи создается вспомогательная задача Z' на доказательство, посылки которой суть все посылки задачи Z, а также антецеденты $A_1, \ldots, A_m$. Условием новой задачи на доказательство служит консеквент A_0. Так как задачи Z и Z' эквивалентны, то после решения второй сразу выдается ответ (либо отказ) на первую. Попутно выполняется коррекция тех структур данных задачи Z, в которых регистрируется, какие из ее посылок в действительности были использованы при доказательстве.

В данном приеме символом привязки естественно выбрать "длялюбого"; уровень срабатывания будет равен 3 (чтобы успели сработать приемы общей стандартизации, которые могли бы существенно упростить кванторную импликацию или вовсе исключить ее квантор).

Первые операторы программы приема нам уже знакомы: "решить стандарт(x2) уровень(3) новый". Чтобы указать на то, что кванторная импликация размещается в условиях задачи и имеет корневое вхождение, далее добавляем операторы "равно(x4 1) корень(x2)". Следующий оператор усматривает задачу на доказательство: "тип(x1 доказать)".

Определяем список x6 антецедентов кванторной импликации и находим результат x7 добавления к нему всех посылок текущей задачи: "равно(x6 выписка(x7 антецедент(x2 x7)подтерм(x7))) равно(x7 конкатенация(x6 списокпосылок(x1)))".

Теперь можно формировать новую задачу Z'. Ее можно было бы создать без какого-либо использования комментариев задачи Z - с пустыми списками комментариев. Однако, комментарии задачи Z, возможно хранят какую-то ценную информацию, для повторного получения которой потребовались бы дополнительные затраты вычислительной трудоемкости. Перенесение всех ее комментариев в задачу Z' нецелесообразно, так как она все же отличается от Z, и какие-то комментарии могут в ней утратить свою ценность либо оказаться ложными (хотя такая ложность обычно не сказывается на корректности действий с посылками и условиями задачи, она может существенно исказить логику принятия решений при управлении логическим процессом). Поэтому при создании задачи Z' следует проанализировать старые комментарии с помощью какой-то процедуры и отобрать либо модифицировать их для использования в Z'. Обычно здесь используются справочники, которые распадаются на множество мелких подпроцедур, каждая из которых анализирует и при необходимости модифицирует комментарии с заданным заголовком. В данном приеме введены два таких справочника, названия которых суть "новаяпосылка"и длялюбого". Пер-

вый из них обрабатывает комментарии к посылкам задачи, второй - комментарии к самой задаче. Справочник "новаяпосылка"имеет входные переменные x1 - задачу Z; x2 - либо 0 (если анализируется общий комментарий к посылкам), либо вхождение посылки, к которой относится комментарий, в список посылок; x3 - анализируемый комментарий. Если перенесение комментария в новую задачу нецелесообразно, то справочник выдает значение 1; если комментарий переносится без изменения, то выдается 0, в остальных случаях выдается модифицированный комментарий x3. Аналогично устроен справочник "длялюбого", но у него только две входных переменных: x1 - задача; x2 - комментарий. Фактическое заполнение двух указанных справочников процедурами обработки комментариев выходит за рамки нашего примера. Оно может быть продолжено по мере обучения решателя и появления комментариев новых типов. Так как случаи обязательного отбрасывания или коррекции комментариев довольно редки, число реально написанных процедур этих двух и аналогичных им справочников обработки комментариев, используемых в других приемах, невелико.

Оператор, формирующий новую задачу x8 (т.е. Z'), достаточно громоздок; в отсутствии многоцветной указки решателя, будем здесь и далее в таких случаях использовать разбиение записи оператора на блоки. Прежде всего, нам понадобится операторное подвыражение A_1, определяющее список списков комментариев к посылкам новой задачи. Так как новый список посылок получился присоединением к началу старого набора антецедентов x6, которые пока не сопровождаются комментариями, то A_1 представляем в виде конкатенации набора пустых слов, имеющего ту же длину, что и x6, и набора, составленного из модифицированных списков комментариев задачи Z: "конкатенация(бланк(x6 пустоеслово 0)выписка(x9 позиция(x9 левпозиция (x1 3))A_2))". Подвыражение A_2 здесь определяет модификацию текущего (расположенного по вхождению x9) списка комментариев к посылке (или посылкам) старой задачи. Эта модификация получается при просмотре элементов x10 данного списка: "выписка(x10 x11 и(входит(x10 буква(x9))равно(x11 A_3)не(равно(x11 1))вариант(равно(x11 0)x10 x11)))". x11 - результат обращения к справочнику "новаяпосылка"для обработки комментария x10; он определяется подвыражением A_3: "справка(новаяпосылка начало(x10)x1 вариант(равно(правсосед(x9)x9)0 соотвпозиция(x9 левпозиция (x1 3)списокпосылок(x1))) x10)". Условие "равно(правсосед(x9)x9)"здесь означает, что вхождение x9 списка комментариев - последнее, то есть он является списком общих комментариев к посылкам задачи Z. Выражение "соотвпозиция(x9 левпозиция(x1 3)списокпосылок(x1))" определяет вхождение той посылки, которой соответствует вхождение x9 списка комментариев.

Далее строим подвыражение A_4 для списка комментариев к задаче Z': "выписка (x10 x11 и(входит(x10 комментарии(x1)) равно(x11 A_5)не(равно(x11 1)))вариант(равно(x11 0)x10 x11))". Здесь A_5 - обращение к справочнику "длялюбого", анализирующему старый комментарий x10: "справка(длялюбого начало(x10)x1 x10)".

Используя блоки A_1, A_4 получаем оператор для построения задачи Z': "равно(x8 набор(доказать x7 бланк(x7 0 0)A_1 последнийтерм(x2) 0 A_4))".

Далее организуем обращение к решению задачи Z'. Прежде всего, применяем оператор "подуровень(x1)", который устанавливает максимальный уровень новой задачи равным максимальному уровню задачи Z (так как она целиком сводится к Z', последнюю следует решать с привлечением средств того же уровня, что и для Z). Далее размещается оператор "равно(ответзадачи(x8)истина)", который одновременно решает задачу Z' и проверяет, что ее ответом служит логический символ "истина". После него идет оператор перехода "иначе 1" (в каждом фрагменте переходы из него нумеруем начиная с 1), определяющий переход к фрагменту, состоящему из

единственного оператора "ответ(отказ)". Этот отказ на задачу Z будет выдан, если на задачу Z' не был получен ответ "истина".

После оператора перехода "иначе 1" остается выдать ответ "истина" на задачу Z, предварительно скорректировав ее комментарий "выводимо A". В списке A перечисляются все посылки задачи Z, использованные при ее решении, причем они берутся в том виде, какой они имели на начальный момент решения задачи Z. Оператор, выполняющий коррекцию, имеет вид: "длялюбого(x9 если ключ(комментарии(x8)выводимо x9)то учетпосылок(x1 конец(x9)0 1))". Он извлекает из задачи Z' список "конец(x9)"ее исходных посылок, использованных при решении, и определяет, из каких именно исходных посылок задачи Z они получены. При этом используется несложный вспомогательный оператор "учетпосылок(t_1 t_2 t_3 t_4)". Программу приема завершает оператор "ответ(истина)".

В заключение рассмотрим прием решения задач на описание, в которых нужно подобрать какой-нибудь пример значений неизвестных, удовлетворяющих условиям. Прием предпринимает попытку усмотреть эти значения непосредственно из посылок задачи. Если удается найти такие посылку A и условие B, что A представимо (с точностью до изменения порядка операндов коммутативных операций и симметричных отношений) как результат подстановки в условие B вместо всех неизвестных $x_1, \ldots, x_n$, имеющихся в B, выражений $t_1, \ldots, t_n$, то эти выражения фиксируются в качестве значений неизвестных, причем найденные значения подставляются в оставшиеся условия. Если после подстановки в каком-либо условии не остается неизвестных, то его истинность проверяется с помощью решения вспомогательной задачи на доказательство; процесс подбора значений и проверки продолжается до полной реализации всех условий.

Данный прием, в отличие от трех предыдущих, не может быть закреплен за каким-либо определенным логическим символом, встречающимся в условиях и посылках задач. Поэтому он закрепляется за типом задачи - логическим символом "описать". Уровень его применения не должен быть ни чрезмерно большим - так как тогда слишком поздно будет происходить усмотрение очевидных наборов значений, ни чрезмерно маленьким - так как во многих случаях имеются более простые средства получения примера значений неизвестных, а действия приема сравнительно трудоемки. В решателе этот прием имеет уровень срабатывания 4.

На примере данного приема продемонстрируем, как происходит поиск точки ответвления к приему от уже имеющихся фрагментов программы логического символа. Если войти в корневой фрагмент программы символа "описать", то будут видны операторы "решить иначе 1 равно(x2 x3) ветвь 2 ...". Оператор "решить" означает, что обращение к программе символа "описать" происходит из сканирования задачи - то есть программа нашего приема должна быть размещена после этого оператора. Переход "иначе 1"ведет к рассмотрению других возможных типов обращения к данной программе.

Обычно программа логического символа имеет главный ствол, на котором размещаются операторы, фиксирующие тип обращения; от каждого такого оператора к следующему имеется переход через "иначе 1".

После оператора "решить"расположен оператор "равно(x2 x3) его роль заключается в отсечении ситуаций, где обращение к программе символа "описать" произошло из-за появления этого символа в условии либо посылке задачи (например, если решается задача относительно структур данных самого решателя). В указанных ситуациях значением x2 было бы вхождение символа "описать" в терм задачи, а значением x3 - вхождение терма задачи, то есть эти значения различались бы. Сов-

падают они только при обращении к программе символа "описать" из рассмотрения типа текущей задачи - тогда x2 и x3 равны 0. При поиске точки ответвления к приему мы проходим данный оператор и перемещаемся вдоль программы дальше.

Затем идет оператор "ветвь 2", после которого располагаются операторы, никак не связанные с нашим приемом. Поэтому переходим к следующему фрагменту через "ветвь 2".

Новый фрагмент начинается с двух операторов перехода "ветвь 1", "ветвь 2", после которых идет оператор "уровень(0)". Он отсекает в нашем случае (т.е. при уровне 4) дальнейшее продвижение, так что нужно воспользоваться одним из указанных переходов. Сразу заметим, что первый из них группирует приемы, для которых уровень срабатывания либо очень велик, например, равен максимальному уровню задачи, либо почти не фиксирован (например, попытка применить прием выполняется для каждого уровня, начиная с некоторого). Во втором размещаются обычные приемы, для срабатывания которых выделены один или два-три уровня. Поэтому для дальнейшего продвижения выбираем переход "ветвь 2".

Очередной фрагмент начинается с операторов "ветвь 1 уровень(2 3)", так что однозначно переходим по оператору "ветвь 1". Далее идет фрагмент с началом вида "уровень(1 3)иначе 1", и снова однозначно переходим через "иначе 1".

Наконец, возникает фрагмент с началом "уровень(4)иначе 1 ветвь 2". Собственно говоря, этот фрагмент является как раз началом той программы, которую мы хотим получить в данном примере. Если бы это было не так, то можно было бы продолжить перемещение вдоль его операторов, пытаясь максимально использовать уже имевшиеся ранее операторы (они "бесплатные", так как все равно будут выполняться при сканировании). Встретив где-либо операторы, использование которых невозможно или нежелательно, можно было бы создать ветвь программы нового приема двумя способами - либо вставить оператор перехода, ведущий к этой ветви, либо сделать так, чтобы старая часть программы сама оказалась достижима через оператор "ветвь (...)"из новой программы. Последнее достигается стандартной операцией - переходя в режим просмотра подтермов операторов, выбираем тот оператор, который должен быть началом отдельной ветви "старой" программы, и нажимаем "p" (кир.). После этого конец фрагмента отрезается и попадает в отдельную ветвь. Начало фрагмента завершается операторами "ветвь(...)" и "продолжение". Войдя в режим редактирования, далее вместо оператора "продолжение" набираем текст новой программы.

Возвращаемся к программе нашего приема - приводим ее далее в точности такой, как она указана в фрагменте, до которого мы дошли указанным выше образом. Следующий ее оператор - "цель(x1 пример)" - означает, что текущая задача на описание должна иметь цель "пример". Расположенный за ним оператор перехода "иначе 3" ведет к программе некоторого другого приема, тоже относящегося к задачам на описание с целью "пример". Далее располагаются операторы "не(цель(x1 развертка)) не(цель(x1 редакция)) не(цель(x1 перечисление)) не(Входит(независит цели(x1)))". Все эти операторы отсекают ситуации, в которых применение данного приема не нужно (например, при редактировании уже найденного ответа) - они появились не при первоначальном создании программы приема, а вводились по мере надобности при проработке обучающего материала. Они являются эвристическими фильтрами, и при развитии системы могут оказаться измененными либо вообще удаленными. Для понимания того, зачем нужен тот или иной фильтр, необходимо предъявление конкретных контекстов, их создавших, так что сейчас мы не будем комментировать перечисленные оперторы.

Следующий оператор - "лимит(набор(3 0 0 0 0 0)) выполняет роль ограничителя

времени, которое будет затрачено на попытку непосредственного подбора значений неизвестных. Эта попытка будет прервана ровно через 300000 шагов работы интерпретатора после своего начала, и программа вернется к данному оператору "лимит", рассматривая его уже как ложный. Константа 300000 тоже является эвристической (выбрана из рассмотрения тех задач, в которых данный прием должен был сработать), и при развитии решателя может быть изменена. Она не очень велика - трудоемкость решения простых и средних задач обычно составляет один-два десятка миллионов шагов.

Чтобы определить список x6 всех неизвестных задачи, встречающихся в ее условиях, используем оператор "равно(x6 пересечениесписков(окончание(неизвестные(x1))параметры (списокусловий(x1))))". Напомним, что выражение "неизвестные(x1)" дает набор (неизвестные $x_1 \dots x_n$), и для выделения собственно списка неизвестных $x_1 \dots x_n$, у этого набора нужно отбросить первый элемент, что и делается операторным выражением "окончание(...)". Подбор значений имеет смысл предпринимать лишь в случае непустого списка неизвестных, так что далее размещается оператор "не(равно(x6 пустоеслово))".

Прежде, чем переходить к подбору значений переменных списка x6, проверяем, что все не содержащие неизвестных условия задачи являются следствиями ее посылок, и одновременно создаем список x7 всех посылок, использованных при проверке истинности таких условий: "равно(x7 пустоеслово) длялюбого(x8 если входит(x8 списокусловий(x1)) известно(x8 x1)то существует(x9 и(извлекается(x8 списокпосылок(x1)x9) замена(x7 объединениесписков(x7 x9)))))".

Далее определяем список x8 всех условий задачи, содержащих неизвестные, и проверяем, что все они суть элементарные (бесквантворные, не содержащие связок "и","или") утверждения: "равно(x8 выписка(x9 и(входит(x9 списокусловий(x1)) не(известно(x9 x1)))x9)) длялюбого(x9 если входит(x9 x8)то элементарно(x9))".

Наконец, обращаемся к оператору "подборнеизвестных", уже упоминавшемуся в главе "Библиотека вспомогательных операторов ЛОСа" : "подборнеизвестных(списокпосылок(x1)x8 x6 x9 x10)". Затем регистрируем в комментарии "выводимо A" к задаче использованные на данном шаге исходные посылки (для этого имеем списки использованных утверждений x7 и x10): "учетпосылок(x1 объединениесписков(x7 x10)0 1)".

Выходная переменная x9 оператора "подборнеизвестных"имеет своим значением набор $t_1 \dots t_n$ выражений, определяющих значения переменных $x_1 \dots x_n$ списка x6 (расположенных в том же порядке). На основе списков x6 и x9 создаем ответ задачи - утверждение $x_1 = t_1 \,\&\dots\, \& \, x_n = t_n$: "равно(x11 унисборка(и выписка(x12 x13 серия(x12 x6 x13 x9)запись(равно буква(x12)буква(x13)))))"; в конце программы помещаем оператор выдачи ответа - "ответ(x11)".

9.2 Примеры программ вспомогательных операторов и операторных выражений

Рассмотрим пример вспомогательного оператора, осуществляющего поиск в некотором списке термов всех его элементов A, имеющих хотя бы одно вхождение заданного терма T, и выдающего в качестве результата список пар: (A - список всех вхождений в A терма T, упорядоченных слева направо). Входные данные этого оператора суть: x1 - набор термов; x2 - заданный терм T. Оператор имеет единственную выходную переменную x3 - ей присваивается указанный выше набор пар.

Первый шаг при программировании вспомогательного оператора - выбор названия для этого оператора. Через главное меню осуществляется ввод нового логического символа - названия оператора (в отдельных случаях можно использовать ранее введенный логический символ, если для него не была создана программа вспомогательного оператора; наличие такой программы распознается по оператору "программа"в начале фрагмента). В нашем примере введем для названия программы новый логический символ "вхождениятерма".

После выбора названия A оператора следует войти в просмотр корня программы логического симвла A и выбрать точку ответвления для записи программы оператора A. Если программы логического символа A вообще не было, то программирование начинается прямо с корня; если, например, корневой фрагмент имел вид "решить ...", то он изменяется на "решить иначе N...", где N - номер перехода к новой программе (N выбирается произвольно - лишь бы отличалось от номеров других, ранее введенных переходов из корневого фрагмента).

Первые операторы в нашем примере имеют вид: "программа метка(икс(4))". Здесь оператор "метка(икс(4))"указывает, что номер первой не определенной при обращении программной переменной равен 4. Это оператор является фиктивным и интерпретатор ЛОСа его игнорирует. Однако, он важен для процедур контроля ошибок в программах ЛОСа, так как позволяет распознать уже определенные при обращении переменные. Кроме того, при наличии оператора "метка(икс(...))", проверка новой программы оператора или операторного выражения приводит к автоматическому созданию приема справочника "арность", указывающему, сколько операндов должен иметь оператор. Этот справочник тоже нужен для контроля ошибок в программах. Наконец, система контроля ошибок в программах устроена так, что само отсутствие в начале программы оператора или операторного выражения фиктивного оператора "метка(икс(...))"воспринимается как ошибка.

Для получения требуемого списка пар - значения выходной переменной нашего оператора - организуем просмотр списка x1. Для текущего терма из x1 составим список всех вхождений в него терма T, и если этот список непуст, то зарегистрируем его в накопителе результата. Все эти действия выполняются единственным оператором выдачи результата: "результат(x3 выписка(x4 x5 и(входит(x4 x1)равно(x5 выписка(x6 вхождениетерма(x4 x2 x6)x6))не(равно(x5 пустоеслово)))набор(x4 x5)))". Просмотр списка x1 здесь обеспечивается перечисляющим оператором "входит(x4 x1)"; список x5 всех вхождений в x4 терма x2 (т.е. T) составляется при помощи перечисляющего оператора "вхождениетерма".

После указанного выше оператора "результат(...)"необходимо поставить завершающий программу оператор "выход", так как иначе данный оператор "вхождениятерма"выполнялся бы в режиме перечисления (в данном случае по определению оператора может быть выдано лишь одно значение переменной x3, и режим перечисления не нужен).

Если программа вводилась для нового логического символа, то после ее набора можно выйти в корневой фрагмент и нажать клавишу "п". Тогда (как говорилось выше) будет автоматически введена программа справочника "арность", определяющая арность (число операндов) у нового оператора - эта программа будет использоваться при при контроле правильности программ. Одновременно будет выполнена проверка правильности программы (анализ арностей операторов; учет определенности либо неопределенности входных и выходных переменных операторов программы, и т.п.).

Следующий пример - программа оператора, работающего в режиме перечисления

значений своих выходных переменных. Такие операторы полезны при программировании процедур, осуществляющих поиск в сложных структурах данных: собственно просмотр необходимых групп объектов осуществляется единственным обращением к оператору. Перечисляющие операторы избавляют от необходимости создавать специальные конструкции для организации циклов; особенно упрощает программирование их использование при перечислениях по рекурсии. Чтобы программа оператора заработала в режиме перечисления, достаточно после оператора "результат(...)", определяющего значения выходных переменных, не ставить оператор "выход". Тогда при откате произойдет возвращение в данную программу - в последний из ее перечисляющих операторов, который был выполнен перед выдачей результата. Хотя формально можно часть выходов из программы делать перечисляющими, а часть - неперечисляющими, и интерпретатор ЛОСа будет правильным образом обрабатывать такие выходы, этот режим программирования на ЛОСе оказался фактически невостребованным. Операторы со смешанным перечислением оказались неудобны из-за трудностей, возникающих при подсчете числа сбрасываемых внешних перечислений в явно указываемых откатах. Поэтому в операторах, программируемых на ЛОСе, режим смешанного перечисления вообще не используется. Процедура контроля программ выдает сообщение об ошибке, если часть выходов из программы оператора - перечисляющие, а часть - не перечисляющие. Среди базисных операторов ЛОСа наиболее часто порождает смешанный режим перечисления оператор "альтернатива".

Рассмотрим в качестве примера программирование следующего простого перечисляющего оператора, связанного с одночленами. Входными данными нашего оператора будут некоторый набор выражений x1, а также набор x2 натуральных чисел, имеющий ту же длину, что и x1. Выходная переменная x3 перечисляет произведения целых неотрицательных степеней выражений списка x1, показатели которых не превосходят соответствующих этим выражениям разрядов набора x2. Одновременно выходная переменная x4 приобретает в качестве значения произведение степеней выражений списка x1, показатели которых дополняют показатели произведений x3 до значений набора x2. Этот пример уже реализован в системе как программа оператора "Делитель".

Первые операторы программы суть "программа метка(икс(5))". При вычислении произведений x3,x4 будем использовать рекурсию по длине наборов x1,x2. Поэтому следующие операторы программы суть "равно(x5 окончание(x1))"; "равно(x6 окончание(x2))" - заблаговременно определяем результаты x5,x6 отбрасывания первых элементов наборов x1,x2 для рекурсивного обращения к обработке укороченных наборов. Далее будем последовательно рассматривать возможные значения показателя степени при первом элементе A набора x1 - от 0 до первого элемента n набора x2. Вводим операторы: "равно(x7 0) равно(x8 начало(x2))". Здесь x7 - текущее значение показателя степени; x8 - значение, дополняющее его до n.

Далее будет организован цикл по x7 с применением оператора "повторение"(этот цикл можно было бы не создавать, а использовать вместо него перечисляющий оператор "Номера(0 начало(x2)x7)", который перечислял бы все целочисленные значения x7 от нулевого до максимального; однако, здесь полезно показать, как на ЛОСе происходит организация "явных" циклов). После оператора "повторение" разместим оператор перехода "ветвь 1"; нумерацию переходов в этом примере берем сквозную. По этому оператору будем переходить в ветвь программы, обеспечивающую очередное изменение x7,x8 и выход из цикла. После оператора "ветвь 1" размещаем оператор "равно(x5 пустоеслово) иначе 2 ", который обеспечивает раздельную обра-

ботку в случае завершения рекурсии (если список x5 пуст) и в случае рекурсивного обращения к укороченным наборам.

Если рекурсия завершена, то непосредственно выдается результат - здесь используется оператор "результат(x3 вариант(равно(x7 0)набор(1)вариант(равно(x7 1)начало(x1)запись(степень начало(x1)десзапись(x7))))x4 вариант(равно(x8 0)набор(1)вариант(равно(x8 1)начало(x1)запись(степень начало(x1)десзапись(x8)))))". Как видно, сначала формируется вырожденное произведение x3, с разбором случаев: $x7 = 0$ (выдается константа 1), либо $x7 = 1$ (выдается выражение A - начало набора x1), либо $x7 > 1$ (выдается степенное выражение). Затем аналогичным образом формируется произведение x4.

При рекурсивном обращении к укороченным наборам (переход к подфрагменту 2) программа имеет вид: "Делитель(x5 x6 x9 x10) результат(x3 вариант(равно(x7 0)x9 соединение(умножение вариант(равно(x7 1)начало(x1)запись(степень начало(x1)десзапись(x7)))x9)) x4 вариант(равно(x8 0)x10 соединение(умножение вариант(равно(x8 1)начало(x1)запись(степень начало(x1)десзапись(x8)))x10)))". Здесь x9,x10 - результат обработки пары укороченных наборов. Далее к x9,x10 добавляются необходимые множители, формируемые так же, как и выше. Добавление множителей происходит при помощи операторного выражения "соединение(...)", которое устраняет вложенные умножения.

Подфрагмент 1 имеет вид "равно(x8 0) иначе 3 стоп". Это - проверка окончания цикла при $x7 = n$. Наконец, подфрагмент 3 имеет вид "замена(x7 плюс(x7 1)) замена(x8 вычитание(x8 1)) продолжение" - он обеспечивает увеличение на единицу значения переменной x7 и уменьшение на единицу значения переменной x8. Напомним, что фактически в программе расположены операторы "замена(7 плюс(x7 1))", "замена(8 вычитание(x8 1))", а в указанным выше виде они выдаются на экран только для работы в редакторе программ. Отладчик ЛОСа, выдавая при трассировке на экран записи операторов программы, не выполняет такого преобразования для "замен".

В качестве примера программы операторного выражения рассмотрим процедуру, осуществляющую переход от предствления матрицы "по строкам"к ее представлению "по столбцам". Пусть x1 - набор $(A_1, \ldots, A_n)$ наборов $A_i = (a_{i1}, \ldots, a_{im})$, представляющих строки некоторой матрицы A. В качестве значения операторного выражения требуется получить набор $(B_1, \ldots, B_m)$ столбцов $B_j = (a_{1j}, \ldots, a_{nj})$.

Выберем прежде всего название для операторного выражения (например, "столбцыматрицы") и введем его в качестве нового логического символа через главное меню.

Первые два оператора новой программы операторного выражения имеют вид "обращение(0) метка(икс(2))". Первый из них указывает, что происходит вычисление программного выражения; второй указывает на номер 2 первой не определенной при обращении программной переменной.

Собственно вычисление набора столбцов и выдача результата выполняются единственным оператором:

"ответ(выписка(x2 позиция(x2 начало(x1))выписка(x3 входит(x3 x1)буква(соотвпозиция(x2 начало(x1)x3)))))".

Здесь происходит просмотр всех вхождений x2 разрядов в первую строку матрицы, и для фиксированного такого x2 составляется список всех разрядов различных строк матрицы, расположенных соответственно разряду x2 первой строки. Значение операторного выражения, реализованного программой на ЛОСе, выдается при помощи оператора "ответ(...)". Если бы программа операторного выражения была

завершена оператором "стоп"либо выход из нее произошел при откате, то в качестве значения был бы выдан логический символ 0.

Во многих случаях возникает необходимость в серии программ, обеспечивающих однотипные вычисления либо выдачу справочной информации для различных логических символов. Такая серия программ получает общее название - некоторый логический символ A - и называется справочником A. Обращение к справочнику A для обработки логического символа S имеет вид "справка(A S $x_1 \dots x_n$)", где $x_1, \dots, x_n$ - дополнительные входные данные (как уже говорилось, однотипные для всех программ справочника A). Это обращение представляет собой операторное выражение, так что справочник всегда возвращает в качестве результата некоторый объект (в справочной информации о логическом символе A этот объект обычно обозначается R). Если программа справочника A для логического символа S отсутствует, либо выход из нее осуществляется через оператор "стоп", либо при откате, то в качестве зачения возвращается логический символ 0. Одно из удобств при использовании справочников заключается в возможности независимого программирования его процедур, обслуживающих различные логические символы. Кроме того, время обращения к нужной процедуре справочника практически никак не зависит от числа ранее запрограммированных процедур этого справочника.

В качестве примера рассмотри справочник "одз", обеспечивающий получение условий на допустимые значения операндов некоторой операции либо некоторого предиката. Логическим символом S здесь служит указанная операция либо предикат; дополнительные входные данные таковы: x1 - задача, в которой встречается рассматриваемое вхождение S; (x2,x3,x4) - координата вхождения символа S в задачу x1 (x2 - вхождение S в терм задачи; x3 - вхождение этого терма в список задачи; x4 - указатель посылки (0) либо условия (1)).

Запрограммируем, например, процедуру справочника "одз", обслуживающую логический символ "тангенс". Первые два оператора программы (она вставляется в программу символа "тангенс") таковы: "обращение(одз) метка(икс(5))". Как обычно, здесь 5 - номер первой не определенной при обращении программной переменной. Первый оператор указывает название справочника.

Далее располагаются следующие операторы:

"равно(x5 первыйтерм(x2)) равно(x6 выписка(x7 или(и(не(существует(x6 и(равно (x6 справка(тип буква(первыйоперанд(x2))))не(равно(x6 0))входит(число x6))))равно (x7 запись(число x5)))равно(x7 запись(не запись(равно запись(косинус x5)0))))x7)) ответ(x6)".

Первый из них определяет терм x5 - операнд тангенса. Далее используется конструкция с оператором "выписка", в которой последовательно (с помощью оператора "или") формируются два элемента x7, регистрируемые в результирующем списке условий. Первый из этих термов x7 имеет вид "число(x5)"; второй - вид "не(равно(косинус(x5)0))". Чтобы избежать перечисления условий, очевидным образом уже выполненных в контексте рассматриваемого вхождения, регистрация условия "число (x5)"в списке предваряется проверкой того, что терм x5 не имеет своим заголовком операцию, принимающую только числовые значения. Для этой проверки используется справочник "тип".

Подробная информация о справочнике с названием A размещается в справочной информации логического символа A и при работе с редактором программ ЛОСа доступна, например, через F2. В поясняющих текстах информация о справочнике A обычно начинается со слов "A - индекс обращения.". При этом результат обращения к справочнику обозначается посредством R.

9.3 Упражнения по программированию на ЛОСе

9.3.1 Просмотр программ

1. Войти из главного меню в корневой фрагмент программы символа "синус".

2. Найти в программе символа "синус" фрагмент, содержащий оператор "уровень (1 3)". Найти фрагменты, содержащие операторы "уровень(2)" и "уровень(8)".

3. Найти все вхождения в программу символа "синус" оператора "символ(x20 косинус)". Сколько таких вхождений ?

4. Находясь в программе символа "синус", посмотреть справочную информацию для символа "синус". Находясь в той же программе, посмотреть справочную информацию для символа "степень".

5. Находясь в корневом фрагменте программы символа "синус" и используя только мышь, посмотреть справочную информацию символа "заменавхождения". Просмотреть все страницы этой информации.

6. Находясь в корневом фрагменте программы символа "синус" и не выходя в главное меню, перейти к просмотру корневого фрагмента программы символа "заменавхождения". После этого, опять же не выходя в главное меню, вернуться к корневому фрагменту программы символа "синус".

7. Находясь в корневом фрагменте программы символа "синус", перейти в оглавление операторов ЛОСа; найти в нем цепочку подразделов "Операторы для работы с логическими структурами данных" - "Списки термов" - "Лексикографическое упорядочение набора термов" и просмотреть справочную информацию для логического символа "лексупорядочение". Затем вернуться в программу символа "синус".

8. Войти в корневой фрагмент программы символа "заменавхождения"и перейти от него к подфрагменту вдоль цепочки переходов с номерами: 2, 1, 1, 2. Перейти к пункту оглавления программ, соответствующему ссылке "прием(5 9)", имеющейся в найденном фрагменте.

9. Войти в корневой фрагмент программы символа "синус" и, не выходя в главное меню, просмотреть корневые фрагменты программ нескольких следующих после синуса логических символов.

10. Войти в корневой фрагмент программы символа "биссектриса", перейти в режим просмотра подтермов операторов и выделить в предпоследнем операторе фрагмента многоцветной указкой подтерм "списокпосылок(x1)". Далее вернуться в режим обычного просмотра программы. Проделать эту операцию дважды - сначала только с помощью клавиатуры, затем - только с помощью мыши.

11. Войти в программу символа "синус"и, используя только клавиши курсора, найти фрагмент, содержащий оператор "обращение(тип)". Далее вернуться к корневому фрагменту программы "синус", используя единственное нажатие клавиши.

12. Войти из главного меню в оглавление операторов ЛОСа и найти в нем операторное выражение, определяющее номер вхождения в набор. Перейти в просмотр программы этого операторного выражения, после чего снова вернуться в оглавление операторов ЛОСа.

13. Найти в оглавлении операторов ЛОСа оператор, проверяющий отсутствие в терме повторных вхождений одной и той же переменной.

14. Найти в оглавлении операторов ЛОСа оператор, проверяющий, что один терм получается из другого вычеркиванием части операндов некоторой операции.

15. Найти в оглавлении операторов ЛОСа все операторы, обращающиеся к решению вспомогательной задачи на доказательство.

Указания

1. В главном меню нажать клавишу "п" либо нажать левую кнопку мыши в окне "Просмотр программы логического символа", затем набрать текстовым редактором слово "синус" и нажать "Enter".

2. Обычно операторы "уровень(...)"располагаются вдоль главного ствола ветви программы логического символа, содержащей приемы сканирования задачи. Эта ветвь располагается после оператора "решить". В корневом фрагменте видим операторы "уровень(0) иначе 2", так что для поиска операторов "уровень(1 3)", "уровень(2)" и "уровень(8)"переходим по указателю перехода "иначе 2". Напомним, что клавиши "курсор влево - вправо" позволяют выбрать текущий указатель перехода, клавиша "курсор вниз" - переводит по этому указателю, "курсор вверх" - возвращает обратно. После перехода видим операторы "ветвь 1 уровень(1)". Наши операторы выбора уровня не могут размещаться после оператора "уровень(1)", так что перемещаемся по указателю "ветвь 1". Это перемещение приводит к фрагменту с оператором "уровень(1 3)" - один из искомых операторов. Продолжая далее перемещаться вдоль главного ствола (переходы до операторов "уровень" либо непосредственно после них), находим операторы "уровень(2)" и "уровень(8)". Заметим, что иногда сначала идет оператор "уровень($N_1 \ldots N_m$)", объединяющий несколько уровней, а после него размещается ветвь, в которой операторы "уровень" указывают различные сужения исходного списка $N_1, \ldots, N_m$.

3. Чтобы найти в некоторой ветви программы логического символа все вхождения заданного логического символа либо терма, следует сначала перейти в корень этой ветви, затем нажать клавишу F4 и набрать текстовым редактором искомый логический символ либо терм. После нажатия "Enter", завершающего набор, начинается поиск требуемых вхождений. Если вхождение найдено, то на экране изображается содержащий его фрагмент, в котором оно выделено многоцветной указкой. Каждое последующее нажатие F4 приводит к поиску очередного вхождения. По завершении поиска возникает пустой экран, и по нажатии любой клавиши далее восстанавливается изображение на момент начала поиска.

4. Для получения справочной информации о логическом символе, программа которого просматривается, достаточно нажать F3. Если нужно получить инфор-

мацию о каком-либо другом символе, то нажимается F2 и набирается нужный символ.

5. Перевести курсор мыши на заголовок оператора "заменавхождения(...)" и нажать левую клавишу мыши, чтобы этот оператор выделить многоцветной указкой. Затем нажать правую клавишу мыши. Для перелистывания использовать либо "PageUp-PageDown", либо нажатия левой кнопкой мыши на стрелках в меню. Для возвращения в просмотр фрагмента программы нажать любую кнопку мыши внутри текста. Чтобы убрать мышью выделенный подтерм, нажать ее левую кнопку вне области операторов программы.

6. Выделить левой кнопкой мыши вхождение логического символа "заменавхождения" и нажать клавишу "с". Для возвращения в программу символа "синус" нажать "End".

7. Нажать клавишу "Ctr-л", переводящую в оглавление операторов ЛОСа. Для возвращения в просмотр фрагмента программы нажать "End".

8. Чтобы перейти в пункт оглавления программ ЛОСа, соответствующий указателю "прием(N)", найденному в некотором фрагменте программы, нужно выделить многоцветной указкой какой-либо оператор, расположенный после данного указателя, и нажать "Ctr-End". Возвращение в программу - нажатие "курсор вправо" на выбранном пункте оглавления. Данная операция работает только в случаях, когда просмотр программы был начат из главного меню.

9. Для перехода в корневой фрагмент программы следующего (по номеру - первого после текущего, для которого создана программа) логического символа достаточно нажать клавишу "ш".

10. Для входа в режим многоцветной указки (режим просмотра подтермов операторов) нажимается клавиша "о"(кир.). Далее клавишами "курсор влево - вправо" выбирается нужный оператор, клавишей "курсор вниз осуществляется вход в выделение его подоператоров, и т.д. вплоть до нужного подтерма. Этого же можно добиться, нажав левую кнопку мыши на заголовке выделяемого подтерма. При единичном выделении удобнее пользоваться мышью, при последовательном просмотре операторов и их подтермов удобнее пользоваться клавишами курсора.

11. Указатели типов обращений "решить", "программа", "обращение(A)" размещаются в виде линейной цепочки, начинающейся в корневом фрагменте программы. Начало корневого фрагмента программы символа "синус" имеет вид "решить иначе 1". Переходим по указетлю "иначе 1" - появляется фрагмент с началом "обращение(блокредактора) иначе 1". Снова переходим по указателю "иначе 1", и т.д., пока не найдем фрагмент с началом "обращение(тип)". Для возвращения в корневой фрагмент достаточно нажать "End". Последняя операция работает только в случае, если просмотр программы был начат из главного меню.

12. Для входа в оглавление операторов из главного меню нажимаем клавишу "о" (кир.). Находим в корневом меню оглавления раздел "Операторы и операторные выражения, реализованные на ЛОСе". Далее смотрим подраздел "Операторы для работы с наборами и вхождениями в наборы". Затем выбираем пункт

"Вхождения в набор", и наконец - "Номер вхождения в набор". Остальные упражнения на поиск в данном оглавлении делаются аналогично.

9.3.2 Логические символы

1. Определить номер логического символа "синус".
2. Определить название логического символа с номером 1000.
3. Просмотреть названия логических символов с номерами от 290 до 294. Просмотреть справочную информацию об этих символах.
4. Найти логический символ, для которого имеется программа, но нет справочной информации.
5. Найти логический символ, для которого имеется справочная информация, но нет программы.
6. Найти номер логического символа, для которого не введено название.
7. Ввести новый логический символ с названием "pppp".
8. Изменить название логического символа "pppp" на "тттт".
9. Удалить логический символ "тттт".

Указания

1. Войти в пункт "Ресурсы и установки" главного меню, выбрать далее пункт "определение номера символа по его названию", нажав клавишу "с" либо используя левую кнопку мыши, и ввести текстовым редактором слово "синус". Затем нажать "Enter". Для возвращения в главное меню нажать любую клавишу.
2. Войти в пункт "Ресурсы и установки" и выбрать подпункт "Определение названия символа по его номеру". Затем набрать номер 1000 и нажать "Enter". Эту же операцию (и все остальные операции подменю "Ресурсы и установки", кроме описанной в предыдущем пункте) можно выполнять непосредственно из главного меню - начиная ее сразу с нажатия "Ctr - c").
3. Для просмотра списка упорядоченных по номерам логических символов войти из главного меню в пункт "Логические символы". Затем перелистывать страницы с помощью "PageDown - PageUp" до нахождения нужного диапазона номеров символов. Выделить нужный символ голубым цветом, используя клавиши "курсор вниз-вверх" в рамках одного столбца и "курсор влево - вправо" для смены столбца. Для просмотра информации о выделенном символе нажать "и". Для возвращения в просмотр символов нажать любую клавишу, не используемую при просмотре справочной информации.
4. Войти в просмотр списка упорядоченных по номерам логических символов и перелистывать страницы до обнаружения символа, после которого располагается знак "&" и нет знака "!".

5. Аналогично предыдущему, но искать символ, после которого идет "!", но нет "&".

6. Перелистывать список символов до номеров, после которых идут три знака "!", "?", "&".

7. Войти в пункт "Ресурсы и установки" главного меню. Выбрать подпункт "Ввод названия нового символа". Ввести текстовым редактором требуемое слово (если оно уже использовано в качестве названия, то об этом появляется сообщение) и нажать "Enter".

8. Войти в пункт "Ресурсы и установки" и выбрать подпункт "Изменение названия символа". Затем ввести изменяемое название (эту операцию требуется проводить осторожно, избегая вводить отсутствующее название) и нажать "Enter". После нажатия "Enter" старое название уже уничтожено; далее водится новое название. Если новое название уже было использовано, то будет выдано соответствующее сообщение, после чего по нажатии любой клавиши произойдет выход в главное меню. Чтобы в этом случае завершить операцию изменения названия, нужно дальше действовать так же, как при вводе нового логического символа - новое название будет сопоставлено первому свободному номеру символа, т.е. (в обычной ситуации) номеру, для которого только что было уничтожено старое название.

9. Войти в пункт "Ресурсы и установки" и выбрать подпункт "Удаление названия символа". Ввести название удаляемого символа (избегая вводить несуществующие названия) и нажать "Enter".

9.3.3 Редактирование программы

1. Ввести новый логический символ с названием "включсимв". Войти в просмотр программы этого символа. Войти в редактирование справочной информации для символа "включсимв" и набрать текст: "включсимв(x1 x2). x1,x2 - термы. Оператор проверяет, что каждый логический символ, встречающийся в терме x1, встречается также в терме x2". Затем вернуться в просмотр программы, войти в редактирование программы и набрать текст программы: "программа метка(икс(3)) равно(x3 перечисление(x4 и(входит(x4 x1) логсимвол(x4))x4)) равно(x4 перечисление(x5 и(входит(x5 x2)логсимвол(x5))x5)) включается(x3 x4) выход".

2. Разрезать программу символа "включсимв", введенную в предыдущем упражнении, на два последовательно идущих фрагмента, так, чтобы второй начинался с оператора "равно(x4 ...)".

3. Склеить два фрагмента, на которые была разрезана программа символа "включсимв", в один общий фрагмент.

4. Вставить в программу символа "включсимв"программу справочника "арность", состоящую из операторов "обращение(арность) ответ(2)". Затем удалить эту введенную вручную программу и осуществить ее автоматический синтез.

5. Увеличить на 2 номера всех программных переменных, начиная с переменной x4, в операторах программы символа "включсимв", идущих после оператора "равно(x3 ...)". Затем восстановить исходные номера этих переменных.

6. Войдя в редактирование программы символа "включсимв", выполнить следующие стандартные операции текстового редактора: а) переставить местами операторы "равно(x3 ...)" и "равно(x4 ...)", после чего восстановить исходное их размещение; б) Найти для каких-либо открывающих и закрывающих скобок операторов программы двойственные им скобки; в) Перед оператором "равно(x3 ...)" вставить оператор "равно(x3 0)", а затем удалить этот оператор; г) после оператора "равно(x3 ...)" разместить его копию, а затем удалить эту копию.

7. Войдя в редактирование программы символа "включсимв", просмотреть справочную информацию для логического символа "включается", после чего вернуться в редактирование.

8. Найти в программе логического символа "делит" ветвь фрагмента, начинающегося с оператора "обращение(программа)", и скопировать эту ветвь в программу логического символа "включсимв", после чего удалить добавленную ветвь.

9. Найти в программе логического символа "делит" фрагмент, начинающийся с оператора "обращение(выводсимвола)", и вставить копию этого фрагмента в программу оператора "включсимв" между операторами "равно(x3 ...)" и "равно(x4 ...)", отбросив идущий после "обращение(выводсимвола)" оператор "иначе 1". Затем удалить вставленные операторы.

10. Войти в редактирование программы "включсимв" и ввести несколько ошибок в программу - исказить название какого-либо оператора; добавить лишнюю скобку либо удалить имевшуюся. Затем нажать "Enter" и исправить ошибку, на которую будет указано в верхней правой части экрана.

11. Войти в редактирование программы "включсимв" и внести в нее ошибку, изменив оператор "включается(x3 x4)" на "деление(x3 x4)". Затем выйти из редактирования и предпринять автоматическую проверку программы. После получения указания на ошибку исправить эту ошибку.

12. Аналогично предыдущему, но оператор "включается(x3 x4)" заменить на "включается(x3 x5)".

13. Удалить программу логического символа "включсимв", удалить справочную информацию для этого символа, после чего удалить сам символ.

Указания

1. Введя название нового символа, вернуться в главное меню и набрать название этого символа в окне "Просмотр программы логического символа" (набор завершается нажатием "Enter"). После появления пустого экрана нажать F3 и набрать требуемый текст справочной информации. После этого вернуться в просмотр программы (будет восстановлен пустой экран) и нажать клавишу "р". В правом верхнем углу экрана появится предупреждающий о режиме редактирования программы красный прямоугольник, а в левом верхнем углу - курсор

текстового редактора. Далее выполняется набор текста программы, и по завершении набора нажимается "Enter". Если при наборе были допущены ошибки, то в правом верхнем углу появится сообщение об ошибке. При этом режим текстового редактора сохраняется, так что можно сразу же исправить ошибку и снова нажать "Enter". При отсутствии ошибок нажатие "Enter" завершает создание либо изменение фрагмента программы.

2. Для разрезания фрагмента программы на две части войти в режим просмотра подтермов операторов (клавиша "о", кир.), выделить тот оператор, с которого должна начинаться вторая половина разрезанного фрагмента, и нажать клавишу "р"(кир.).

3. Если фрагмент программы заканчивается операторами "ветвь N продолжение", то нажатие клавиши F6 приводит к тому, что текст фрагмента по ссылке "ветвь N" заносится в конец текущего фрагмента вместо двух его последних операторов, а сам этот фрагмент исключается.

4. Нужно войти в редактирование корневого фрагмента программы символа "включсимв" и вставить после оператора "программа" оператор "иначе 1". Затем нажать "Enter". Тогда появится пустой экран, в левом верхнем углу которого будет прорисована цифра 1 - номер ссылки к редактируемому подфрагменту, а под ней - курсор текстового редактора для набора текста этого подфрагмента. После набор текста нажимается "Enter", завершающее ввод ветви программы. Чтобы удалить ветвь программы, имеются два способа: а) в режиме обычного просмотра фрагмента программы выделить оператор перехода к удаляемой ветви и нажать "Ctr-Del"; б) войти в редактирование фрагмента программы и исключить оператор перехода к удаляемой ветви. Способ а) может быть применен лишь однократно; для повторного его применения необходимо сначала выйти в главное меню через "Esc". Это объясняется тем, что удаленная ветвь временно сохраняется в буфере и может быть восстановлена (до указанного выхода в главное меню) в другом месте программы. Способ б) можно применять без каких-либо ограничений.

   Для автоматического контроля программы и ввода подфрагментов, указывающих арность оператора либо операторного выражения, реализованного этой программой, нужно в корневом фрагменте программы нажать клавишу "п". Синтез программы справочника "арность" (для перечисляющего оператора - также справочника "комментпосылок") выполняется только при условии, что после начинающего корневой фрагмент оператора "программа" либо "обращение(0)" не идет оператор перехода "иначе".

5. Для увеличения либо уменьшения номеров программных переменных сначала следует войти в режим просмотра подтермов операторов, затем выделить тот оператор, начиная с которого должны быть изменены номера переменных, и нажать клавишу "п". Тогда в левом верхнем углу появляется курсор текстового редактора. Для увеличения на n номеров переменных, начиная с переменной хm (включая эту переменную), набирается текст "плюс(хm n)"; для уменьшения - "минус(хm n)". После нажатия "Enter" происходит коррекция номеров переменных в данном фрагменте и всех его подфрагментах, достижимых через выделенный оператор.

6. Напомним используемые в данном упражнении операции текстового редактора: а) Для перенесения фрагмента текста на новое место курсор подводится к началу этого фрагмента и нажимается F2; затем курсор подводится к концу фрагмента и снова нажимается F2 (начало и конец фрагмента выделены зелеными маркерами). Наконец, курсор подводится к той позиции, начиная с которой должен быть размещен фрагмент, и опять нажимается F2; б) Для перехода от скобки к двойственной ей курсор устанавливается на скобку и нажимается F1; в) Для вставки фрагмента текста курсор подводится к тому месту, с которого должна начинаться вставка, и нажимается "Insert" курсор становится голубым. Каждый набираемый с клавиатуры символ вставляется на позицию курсора, а курсор перемещается на одну клетку вправо. Выход из режима вставки - повторное нажатие "Insert" либо (для исправления ошибки) "Backspace"; г) Для копирования фрагмент текста курсор подводится к его началу и нажимается "PageDown". Затем курсор подводится к концу фрагмента и снова нажимается "PageDown" - фрагмент таким образом записан в буфере. Для извлечения его из буфера (что может происходить при повторных обращениях к текстовому редактору в произвольных интерфейсах системы) курсор подводится к тому месту, начиная с которого должна быть вставлена копия, и нажимается "PageUp".

   Для удаления фрагмента текста следует подвести курсор к началу фрагмента и нажать "Delete", затем подвести его к первой позиции после конца фрагмента и повторно нажать "Delete".

7. Для просмотра справочной информации о логических символах непосредственно из текстового редактора следует нажать клавишу F4 и набрать название нужного логического символа. После просмотра будет восстановлена текущая ситуация редактирования.

8. Сначала следует войти в программу символа "делит" и, спускаясь вдоль "главного ствола", найти фрагмент, начинающийся с оператора "обращение(программа)". Затем нажимается "Insert" - ссылка на ветвь этого фрагмента сохранена в буфере. После этого нужно выйти в главное меню по нажатию клавиши "PageUp"(при другом способе выхода ссылка на ветвь будет утеряна) и набрать в окне "Просмотр программы логического символа" название "включсимв". Внутри программы "включсимв" следует найти фрагмент оператора "обращение(арность)", войти в его редактирование и вставить после первого оператора переход "иначе 1". Войдя в редактирование подфрагмента по ссылке "иначе 1", набрать единственный оператор "продолжение". Наконец, после нажатия "Enter" вернуться в созданный подфрагмент "продолжение", выделить его оператор многоцветной указкой ("о", кир.) и нажать клавишу "к" для копирования выделенной ветви вместо ветви подфрагмента "продолжение".

9. Эта операция выполняется с помощью буфера текстового редактора: сначала в программе символа "делит" находим фрагмент оператора "обращение(вывод-символа)", входим в редактирование этого фрагмента и заносим (нажатиями "PageDown") его в буфер. Затем переходим в редактирование корневого фрагмента программы символа "включсимв", устанавливаем курсор перед оператором "равно(x4 ...)" и нажимаем "PageUp".

10. При неправильном наборе логического символа в правой верхней части экрана

появляется текст "ошибка в логическом символе", причем курсор текстового редактора устанавливается на начало неправильно набранного символа. Если число закрывающих скобок больше числа открывающих, появляется текст "избыточная правая скобка" и курсор устанавливается на первую правую скобку, не имеющую двойственной левой. Если число открывающих скобок больше числа закрывающих, то появляется текст "незакрытая левая скобка" и курсор устанавливается в левый верхний угол экрана. В этом случае для поиска незакрытой скобки можно последовательно устанавливать курсор на все открывающие скобки, нажимая F1 для проверки наличия соответствующей закрывающей скобки.

11. Для автоматической проверки набранной программы нажимается клавиша "п". Если указать неправильное число операндов (у оператора "деление" оно равно 4), то в правом верхнем углу экрана появится текст "ошибка в числе операндов", причем ошибочный оператор будет выделен многоцветной указкой. Как обычно, для выхода из режима многоцветной указки нажимаем "о" и переходим в режим редактирования для исправления ошибки. Заметим, что такой контроль числа операндов распространяется только на операторы и операторные выражения, запрограммированные на ЛОСе. Если допущена ошибка в числе операндов для базисного оператора либо операторного выражения, то скорее всего она будет обнаружена немедленно, так как после нажатия "Enter" и перерисовки набранного текста программы произойдет искажение сразу нескольких операторов, соседних с тем, у которого число операндов не соответствует норме.

12. При автоматическом контроле правильности программы выявляются входные переменные, значения которых еще не определены, а также выходные переменные, значения которых уже были определены.

13. Для удаления программы логического символа следует войти в ее корневой фрагмент и нажать "Ctr-F7". Для удаления справочной информации о символе войти в ее просмотр (если она размещена на нескольких страницах, то найти последнюю станицу) и далее нажимать "Ctr-Del" столько раз, сколько понадобится для удаления всех ее страниц.

9.3.4 Запуск программы и ее отладочная трассировка

Тестирование процедур логической системы обычно предпринимается в реальном контексте их функционирования - в рамках приема либо блока интерфейса. Поэтому для проверки новой программы обращение к ней включается в соответствующие прием либо блок интерфейса, и далее либо запускается процесс решения задачи, в которой должен сработать рассматриваемый прием, либо выбирается режим интерфейса, обращающийся к новой программе.

Однако, для особых случаев предусмотрена также возможность запуска новой программы безотносительно к контексту предполагаемого ее применения. Для такого запуска нужно предварительно создать входные данные обращения к программе; это удобно делать в рамках некоторой другой программы. В качестве такой стандартной программы, обращающейся к тестируемой программе, выбрана программа логического символа "пуск". Ее первые два оператора "программа метка(икс(2))" следует оставлять без изменений, а все остальные операторы - изменять в соответствии с организацией тестирования. Запуск программы "пуск" осуществляется не-

посредственно из просмотра ее корневого фрагмента нажатием "Ctr-Enter"; после этого устанавливается режим пошаговой трассировки в рамках отладчика ЛОСа. По завершении программы "пуск" происходит возвращение в главное меню.

1. Предпринять пошаговое выполнение программы "включсимв(x1 x2)" при входных данных - термах "синус(плюс(x1 x2 x3))"и "степень(x2 плюс(x1 x3 синус (x2)))".

2. В процессе пошагового выполнения программы "включсимв(x1 x2)"при указанных выше входных данных посмотреть значения переменных x3 и x4.

3. В процессе пошагового выполнения программы "включсимв" посмотреть внешнюю обратившуюся к ней программу "пуск", а также внешнюю обратившуюся к программе "пуск" программу "вход" (она является программой общего интерфейса логической системы). Посмотреть значения программных переменных x2 и x3 для программы "пуск" в двух режимах - с обычной и скобочной прорисовкой формул. Посмотреть значение переменной x1 для программы "вход" (ее значением служит исходная задача).

4. Найти в задачнике раздел "Элементарная алгебра - Упрощение выражений - Разложение на множители-1" и войти в просмотр первой задачи этого раздела. Запустить решение этой задачи с прерыванием при обращении к оператору "заменавхождения". Определить значения программных переменных x1 (заменяемое вхождение) и x5 (заменяющий терм). Определить значения переменных x2, x3, x4 (вхождение текущего терма в список задачи; указатель условия либо посылки; текущая задача). Определить текущий уровень сканирования, при котором был применен прием, обратившийся к оператору "заменавхождения". Используя сквозной просмотр, рассмотреть текущую задачу x4.

5. В контексте предыдущего упражнения (т.е. в начале пошаговой трассировки внутри программы оператора "заменавхождения") предпринять пооператорную трассировку вплоть до обращения к оператору "сопровождение(x4 x1 x2 x3)". Войти в трассировку внутри последнего оператора. Затем выйти в главное меню; повторно войти в контекст предыдущего упраженения, установить прерывание при обращении к программе "сопровождение"и запустить программу до данного прерывания.

6. Войти в трассировку программы "заменавхождения" при решении указанной выше задачи через оглавление программ ЛОСа. После этого выбрать пункт 16 в подразделе оглавления программ, перечисляющем действия оператора "заменавхождения", и выйти в трассировку при достижении программой этого пункта. Проделать ту же операцию непосредственно из запуска задачи, без прерывания в начале выполнения программы "заменавхождения".

7. Запустить решение указанной выше задачи с прерыванием на 14-м пункте подраздела оглавления программ для оператора "заменавхождения". Далее, используя многоцветную указку, установить прерывание при обращении к оператору "равно(x14 справка(одз ...))", и запустить выполнение программы до этого прерывания.

8. Войти в прерывание при обращении к оператору "заменавхождения" в указанной выше задаче и определить результат обращения к операторному выражению "видумножение(...)", находящемуся в программе приема, обратившегося к "заменавхождения".

9. Войти в прерывание при обращении к оператору "заменавхождения" в указанной выше задаче, определить число на счетчике шагов и перезапустить решение задачи с прерыванием по достижении этого числа шагов.

10. Войти в оглавление приемов через главное меню, выбрать какой-либо концевой пункт этого оглавления (после номера концевого пункта идет не скобка, а точка), и нажать "End". Войти через главное меню в корневой фрагмент программы логического символа "оглавление" и вставить перед оператором "повторение" в этом фрагменте оператор "трассировка(стоп 0)". Затем вернуться в главное меню и войти в оглавление базы приемов. После того, как произойдет выход в отладчик ЛОСа по установленной контрольной точке (по оператору "трассировка(стоп 0)"), пошаговой трассировкой дойти до выполнения оператора "файл(терм x10 x12)" и посмотреть значение переменной x12. Затем вернуться в главное меню и убрать из программы "оглавление" контрольную точку.

11. Выбрать в задачнике произвольную задачу на разложение на множители, установить прерывание при обращении к логическому символу "заменавхождения" и запустить решение до этого прерывания. Затем посмотреть текущий кадр семантической трассировки, сопровождающий данное обращение.

12. Для той же задачи запустить ее решение с семантической трассировкой и при первом срабатывании приема посмотреть текущий фрагмент программы, реализующей этот прием. Найти в этой программе оператор "заменавхождения"; посмотреть текущую точку программы "заменавхождения" на момент прерывания.

13. Найти в задачнике раздел "Элементарная алгебра - Решение уравнений - Логарифмические уравнения - Системы уравнений" и войти в просмотр задачи номер 10. Войти в семантическую трассировку решения этой задачи и продолжать ее до появления кадра "Переходим к основанию логарифмов a". Перейти в отладчик ЛОСа и посмотреть список комментариев к текущей задаче. Найти в нем комментарий "нормлогарифм a", определивший новое основание логарифма. Найти комментарий "сопровождение A" и установить прерывание до момента изменения этого комментария, после чего продолжить решение до данного прерывания.

Указания

1. Упражнение выполняется после того, как ранее была создана программа символа "включсимв". Нужно войти в программу символа "пуск" и набрать корневой фрагмент: "программа метка(икс(2)) равно(x2 запись(синус запись(плюс икс(1) икс(2) икс(3)))) равно(x3 запись(степень икс(2) запись(плюс икс(1) икс(3) запись(синус икс(2))))) включсимв(x2 x3) выход". В этом фрагменте сначала вводятся входные данные x2, x3 для обращения к оператору "включсимв", затем реализуется такое обращение. Далее нажать "Ctr-Enter" - начнется пошаговая

трассировка только что набранной программы "пуск" отладчиком ЛОСа. Здесь удобно сразу нажать клавишу "2" для установки пооператорного режима вместо слишком подробного пошагового, и тремя нажатиями "Enter" дойти до обращения к оператору "включсимв". Чтобы войти в трассировку внутри этого оператора, следует снова установить пошаговый режим (нажатием клавиши "1") и нажать "Enter". Далее опять устанавливаем пооператорный режим и последовательно проходим операторы программы "включсимв"до оператора "выход", после которого автоматически происходит возвращение в главное меню.

2. Действия аналогичны описанным выше, но после определения значений переменных x3, x4 в программе "включсимв" предпринимаем просмотр значений этих переменных. Для просмотра сначала нажимаем клавишу "х" (кир.), и далее набираем номер переменной. После нажатия "Enter" справа от набранной переменной появляется ее значение - в данном случае это будут наборы логических символов.

3. Чтобы посмотреть фрагмент программы, из которого произошло обращение к текущему фрагменту программы, нажимаем "PageUp". Делаем это дважды - сначала возвращаемся к программе "пуск", затем - к программе "вход", обеспечивающей интерфейс логической системы. Обратные переходы - с помощью "PageDown". Находясь в рамках программы "вход", можно с помощью клавиш "Home - End" пройтись по всей текущей цепочке ее фрагментов - от текущего фрагмента до корневого. Чтобы перейти к режиму просмотра термов x2,x3 в скобочной записи, нажимается клавиша "с"; для возвращения в стандартную математическую запись - нажимается "м" (кир.). При просмотре значения x1 в программе "вход" появится строчка "исследовать н1 т2 н3 т4". Записи "н1","н3" обозначают некоторые наборы (в данном случае - списки посылок и комментариев к посылкам); записи "т2", "т4" - некоторые термы либо наборы из единственного символа, отождествляемые с термами. Чтобы посмотреть, из чего состоят наборы "нi", нажимаем клавишу "н" и набираем i; чтобы посмотреть термы "тi", нажимаем клавишу "т"и набираем i. Эту операцию можно повторять на любую глубину (кроме термов и наборов, могут встречаться ссылки на вхождения "вi", для которых принцип просмотра тот же). Если экран заполнен просмотренными значениями и нужно его расчистить, нажимаем "Delete" (если этого не делать, то при переполнении экрана начинается прокрутка выписанных значений вверх, не затрагивающая области фрагмента программы). После нажатия "Delete" можно восстановить текст фрагмента прогаммы, нажав клавишу "ф".

4. Для входа в задачник из главного меню нажать клавишу "з"; используя клавиши курсора, найти нужный раздел и в нем - пункт с номером требуемой задачи. Затем войти в просмотр этой задачи (курсор вправо). Для установки прерывания при обращении к программе логического символа "заменавхождения" нажать клавишу "Л" и набрать название данного символа. После нажатия "Enter" инициируется решение задачи с выходом в отладчик ЛОСа при первом же обращении к оператору "заменавхождения". Для просмотра значения переменной x1 нажимаем клавиши "х" (кир.) и "1", "Enter". Голубой цвет всего прорисованного после этого терма либо его части означает, что значением переменной является не сам терм, а выделенное голубым вхождение в него. Точ-

но так же просматриваются значения переменных x2,. . .,x5. Чтобы определить текущий уровень сканирования, при котором был применен прием, обратившийся к оператору "заменавхождения", нажимаем клавишу "PageUp", переводящую в просмотр программы, обратившейся к программе "заменавхождения". После этого достаточно посмотреть значение переменной x5 (напомним, что в программе приема, используемого при сканировании задачи, значения первых 5 переменных определены уже при обращении к ней, и значением x5 при этом служит текущий уровень сканирования; x1 - текущая задача; (x2,x3,x4) - координата текущего вхождения в задачу). В данном случае текст программы приема уже достаточно велик; если бы нам было нужно найти в нем заданный оператор (например, "заменавхождения"), то помогла бы многоцветная указка, для перехода к которой достаточно нажать "курсор вниз" либо нажать левую клавишу мыши в нужной точке программы. Выход из многоцветной указки - нажатие левой клавиши мыши вне области программы либо нажатие (в зависимости от глубины просматриваемого подтерма - одно или несколько) клавиши "курсор вверх". Для входа в сквозной просмотр задачи x4 нужно вернуться (PageDown) в программу "заменавхождения", нажать клавишу "K" (кир.) и номер переменной, затем - "Enter". Тогда появляется список элементов просматриваемого набора; если после номера элемента стоит круглая скобка, то этот элемент - атомарный, и просмотр его элементов невозможен. Если после номера стоит точка, то элемент является набором, и можно посмотреть его элементы, используя клавишу "курсор вправо". Для возвращения к предыдущему уровню просмотра служит "курсор влево".

5. Для перехода от пошаговой трассировки к пооператорной нажать клавишу "2". Затем нажимать "Enter" до появления выделенного малиновым цветом оператора "сопровождение(x4 x1 x2 x3)". Чтобы войти в трассировку внутри этого оператора, вернуться к пошаговой трассировке (клавиша "1") и нажать "Enter". Для обрыва трассировки и выхода в главное меню нажимается "Esc". Повторно войдя в начало выполнения оператора "заменавхождения", нажать клавишу "л" и набрать название оператора "сопровождение". Первым нажатием "Enter" завершить набор, вторым нажатием "Enter" - запустить программу до прерывания при входе в оператор "сопровождение".

6. Войти в просмотр задачи и нажать клавишу "л", переводящую в оглавление программ ЛОСа. В корневом меню этого оглавления выбрать пункт "Приемы решателя", далее - подпункт "Общие процедуры, используемые в приемах", далее - подпункт "Процедура ЗАМЕНАВХОЖДЕНИЯ". После этого нажать клавишу "курсор вправо". Автоматически запускается решение задачи, и при обращении к оператору "заменавхождение" организуется выход в отладчик ЛОСа с установкой пошагового режима трассировки. Чтобы теперь вернуться в оглавление программ ЛОСа, нажимается "Ctr-г". Далее выбирается пункт 16 и нажимается "курсор вправо" - снова организуется прерывание при достижении контрольной точки, соответствующей выбранному пункту (в нашем случае этой контрольной точкой является оператор "прием(1 0)"). При повторе прерывание по достижении пункта 16 устанавливается из первого обращения к оглавлению программ.

7. Войти в прерывание по достижении 14 - го пункта списка контрольных точек оператора "заменавхождения". Затем нажать "курсор вниз" - появляется мно-

гоцветная указка, выделяющая первый оператор программы. Используя "курсор вправо", дойти до выделения оператора "равно(x14 справка(одз ...))" и нажать "Enter". После нажатия запускается решение задачи до прерывания по достижении выделенного оператора.

8. Войти в прерывание при обращении к программе "заменавхождения" и нажать "PageUp" для просмотра внешней программы приема. Найти в ней (например, с помощью многоцветной указки) оператор вида "равно(xi видумножение(...))" - в нашем случае $i = 11$. Затем посмотреть значение переменной x11.

9. Войти в прерывание при обращении к программе "заменавхождения". В правом верхнем углу экрана имеется текст "шаг № N", где N - номер текущего шага интерпретатора ЛОСа, отсчитываемый с момента запуска задачи. Запомнить этот номер, нажать "Esc" для обрыва трассировки. Затем вернуться в просмотр задачи и нажать клавишу "Ш", после чего набрать номер N и нажать "Enter". После нажатия "Enter" устанавливается прерывание по достижении шага с нужным номером и запускается решение задачи. Заметим, что выход в отладчик здесь может произойти не в программе "заменавхождения", а непосредственно перед обращением к ней.

10. Подготовить к последующей трассировке оглавление базы приемов, как указано в упражнении (выбрать его концевой пункт и нажать "End"). Без такой подготовки выполнение всех требований упражнения будет невозможным, так как выполнение программы будет происходить без выхода в нужную точку. После входа в просмотр корневого фрагмента программы "оглавление" нажать клавишу "р" и поместить курсор перед оператором "повторение". Нажать "Insert" и набрать оператор "трассировка(стоп 0)". Затем снова нажать "Insert" (выход из режима вставки) и нажать "Enter". Эти действия следует выполнять чрезвычайно осторожно, так как ошибка в редактировании и нажатие "Enter" могут привести к порче программы "оглавление" (например, к удалению каких-либо ее ветвей), после чего нужно будет восстанавливать программу логической системы из резервной директории "SLCOPY". По завершении редактирования нужно вернуться в главное меню (например, через "Esc") и нажать клавишу "г". Это приведет к входу в программу "оглавление" и немедленной остановке по достижении введенной контрольной точки "трассировка(стоп 0)". Здесь войти в пооператорную трасировку (клавиша "2") и, нажимая "Enter", дойти до выполнения оператора "файл(терм x10 x12)". Длительность трассировки будет зависеть от того, сколь длинным является путь до концевого пункта оглавления базы приемов от корневого меню. Посмотреть значение переменной x12 лучше с помощью процедуры сквозного просмотра (К12; К - кир.). Значением этой переменной служит содержимое логического терминала текущего концевого пункта оглавления базы приемов. Выйдя из трассировки ("Esc"), вернуться в редактирование программы символа "оглавление" и удалить оператор "трассировка(стоп 0)". Это можно делать либо с помощью "Delete" (сначала нажатие "Delete" на первой букве слова "трассировка", затем - на букве, расположенной после удаляемой закрывающей скобки), либо с помощью клавиши "пробел", используемой как ластик. Данную операцию следует выполнять исключительно осторожно, чтобы не удалить других операторов (в особенности операторов перехода). Если, все же, были искажены или удалены какие - то нужные операторы, следует нажать "Esc" для отмены редактирования, затем войти в ре-

дактирование повторно. Удаление контрольной точки "трассировка(стоп 0)" из программы лучше выполнить не откладывая, так как эта контрольная точка блокирует использование каких-либо оглавлений логической системы.

11. Для выхода из технической трассировки в семантическую нажать клавишу пробела и затем "Enter". Можно также, не выходя из режима технической трассировки, посмотреть текущую цепь задач на уровне обычной математической записи, нажав клавишу "з" и используя далее обычные возможности прокрутки (курсоры вверх-вниз; PageUp; PageDown; Ctr-курсоры вверх-вниз). Для возвращения в просмотр ситуации через отладчик ЛОСа нажимается "ф".

12. Войдя в просмотр задачи, нажать клавишу "р" (кир.). После появления на экране сообщения о преобразовании задачи нажать клавишу "ф". Оператор "заменавхождения" расположен в конце программы приема. Чтоюы посмотреть текущую точку внутри программы этого оператора (не завершенной на момент прерывания), нажать "PageDown".

13. Войдя в просмотр указанной задачи, нажать "р", и далее нажимать "Enter" до появления требуемого кадра. Затем нажать "ф". Для просмотра списка комментариев к текущей задаче х1 можно либо сразу войти в сквозной просмотр этой задачи (нажатием "К","1","Enter"), либо сначала посмотреть значение переменной х1 (нажатием "х", "1","Enter") и затем войти в сквозной просмотр набора комментариев н6 (нажатием "к", "6", "Enter"). В наборе комментариев сначала идут комментарии к отдельным условиям, а в конце расположен список общих комментариев к задаче. Найти этот список (выделить его при помощи "курсор вниз"), и нажать "курсор вправо". Далее найти комментарий "нормлогарифм т"; выделить его и снова нажать "курсор вправо" для просмотра терма "$_{\text{т}}$".

    Чтобы устанавливать прерывание на момент изменения разряда набора (только с помощью оператора "изменение(...)"; изменения с помощью специальных операторов списков задачи - посылок, условий, комментариев, весов, - этой установкой не контролируются), нужно сначала вызвать этот набор на экран, чтобы он был обозначен посредством "нi". В нашем случае для этого сначала вызываем для просмотра х1; затем - н6 (список комментариев); затем - н12 (последний элемент списка комментариев - список общих комментариев к задаче); затем - н13 (для проверки того, что это и есть искомый комментарий "сопровождение A"). Далее нужно набрать "и", "13", "Enter". После этого установка прерывания осуществлена, и для запуска решения нажимается "Enter". Заметим, что в случае, когда изменяемый набор является значением программной переменной хi, можно установить аналогичное прерывание нажатием "И", "i", "Enter".

Глава 10

Язык для записи приемов ГЕНОЛО́Г

В первых версиях решателя задач все приемы были реализованы на языке ЛОС. Анализ этих приемов позволил прийти к выводу, что в подавляющем большинстве случаев программа приема основана на применении какой-либо одной теоремы предметной области. Чтобы ускорить программирование приемов и сделать программы их достаточно наглядными (последнее особенно важно для возможностей быстрой коррекции логики принятия решений в процессе обучения), в этой ситуации естественно было разработать язык, в котором прием представляется как теорема предметной области, снабженная некоторой алгоритмизирующей разметкой. Таким образом и возник язык, получивший название ГЕНОЛО́Га (ГЕНетический язык ЛОГического программирования). Алгоритмизирующая разметка теоремы здесь играет роль "генотипа" приема, из которого компилятор создает собственно программу на ЛОСе, реализующую прием.

Развитие ГЕНОЛОГа означает, фактически, описание и систематизацию различных алгоритмических "трюков", применение которых позволяет преобразовать теорему в эффективную программу. Это предполагает рассмотрение все новых и новых предметных областей, которые, как показывает опыт развития решателей, постоянно вносят новые элементы в копилку "алгоритмизаторов" и логических режимов, используемых при решении задач. По этой причине, освоение новой предметной области, как правило, требует (в основном небольших) добавлений к реализованной на ЛОСе программе компилятора для ГЕНОЛОГа. Впрочем, эта работа по своей трудоемкости оказывается несопоставима с последующей экономией, достигаемой при использовании ГЕНОЛОГа - собственно программирование приемов и анализ их устройства при коррекциях ускоряется по сравнению с ЛОСом на порядок. Следует также добавить, что поток новых элементов языка, после проработки значительного обучающего материала, явно идет на убыль.

Помимо соображений экономии трудоемкости программирования и придания описаниям приемов возможно большей наглядности, ГЕНОЛОГ открывает возможность автоматизации синтеза новых приемов, так как играет роль "связующего звена" между логическим языком, на котором происходит накопление новых знаний в предметной области, и алгоритмическим языком, заставляющим работать эти знания при решении задач.

В настоящее время подавляющее большинство приемов решателя реализованы на ГЕНОЛОГе. Он обеспечивает быстрое преобразование логических описаний предметной области (теорем) в описания для логических структур данных, каковыми являются ЛОС - программы. Однако, остается небольшая часть приемов, основанных на утверждениях, которые уже относятся к структурам данных (в основном это

приемы "общелогического" типа), и для них надобность в специальном переходнике от предметного к структурному уровню отсутствует. Эти приемы реализованы непосредственно на ЛОСе. Впрочем, часть реализованных пока на ЛОСе приемов имеют в себе невырожденную компоненту знаний в конкретных предметных областях и, при некотором развитии ГЕНОЛОГа, могли бы также быть переведены на него.

10.1 Основные компоненты описания приема на языке ГЕНОЛО́Г

Описание приема на ГЕНОЛОГе состоит из следующих компонент:

1. Теорема предметной области, на которой основан прием. Эта теорема вводится при помощи формульного либо текстового редактора, в стандартной математической либо скобочной записи. Для большей наглядности предусмотрено сопровождение теорем с геометрическими объектами чертежом; компилятор этот чертеж никак не использует.

   Заметим, что хотя ГЕНОЛОГ и приближен к логическому языку предметной области, все же он представляет собой язык программирования. Настоящие теоремы предметной области содержатся в базе теорем. В теореме приема же допустимы (как правило, небольшие) отклонения от логических стандартов предметного уровня. Эта теорема обычно преобразуется к виду, более удобному для практического использования - например, в посылках могут вводиться вспомогательные обозначения для объектов, упоминаемых в утверждении теоремы; часть посылок, которые в обычных "разумных" контекстах автоматически бывают выполнены, отбрасывается, и т.п. В особых случаях теорема приема вообще может представлять собой техническую заглушку - "псевдотеорему".

2. Заголовок приема. Он указывает на общую схему использования теоремы в приеме.

3. Список фильтров приема. Элементы этого списка суть сформулированные с помощью несколько модифицированного логического языка условия на целесообразность применения приема в текущем контексте. Это - второй логический уровень описания приема; первым уровнем является сама теорема, также задаваемая на логическом языке. Однако, в случае теоремы логический язык используется для описания ситуаций на предметном уровне, а в случае фильтров - для описания ситуаций на структурном уровне. В этом он близок ЛОСу и может рассматриваться (с рядом существенных оговорок) как "ЛОС в теоремных переменных". В действительности многие операторы ЛОСа имеют своих одноименных двойников в языке фильтров приемов, хотя эти двойники используются уже по-другому - у них часто отбрасываются однозначно восстанавливаемые из контекста описания приема операнды. Кроме того, в языке фильтров приема возникают конструкции, совершенно не имеющие аналогов в ЛОСе.

4. Список указателей компилятору (далее называем их просто указателями приема). Элементы этого списка уточняют способ формирования ЛОС-программы приема компилятором ГЕНОЛОГа. По существу, это основная часть "генотипа" приема. Здесь уточняются: средства, используемые для усмотрения в задаче

"замаскированных" возможностей применения теоремы; способы обработки отдельных посылок теоремы; технические комментарии, вводимые в задачу либо удаляемые при применении приема; логика переключения внимания после применения приема, и др.

5. Список нормализаторов. При формировании результирующих (включаемых в структуру данных задачи) либо вспомогательных (рассматриваемых в процессе работы приема) термов могут применяться процедуры для их упрощения либо каких-либо иных специальных преобразований. Такие процедуры задаются последовательным применением нормализаторов - либо специальных пакетов приемов для тождественных и эквивалентных преобразований, либо вспомогательных задач. Использование небольшого пакета приемов вместо общей базы приемов, активизируемой при сканировании задачи, во многих случаях на порядок ускоряет вычисления. Эти "локальные" приемы отличаются от приемов сканирования задачи - они применяются не в контексте задачи, а в некотором другом специальном контексте, состоящем из преобразуемого терма, списка посылок, в предположении истинности которых ведутся преобразования, и списка технических комментариев. Задаются они, как и приемы сканирования задач, на ГЕНОЛОГе.

   Специальный интерфейс позволяет достаточно быстро вводить обращения к нормализаторам путем указания в теореме (либо в сопровождающих теорему термах описания приема) подлежащих нормализации точек. В результате использования нормализаторов и вспомогательных процедур обработки посылок теоремы, срабатывание приема оказывается иногда чрезвычайно сложным вычислительным процессом, основанным не на единственной теореме предметной области, а на десятках и даже сотнях (с учетом прменяемой рекурсии) различных теорем.

6. Шаблоны сопровождения. Здесь хранятся текст-формульные шаблоны, используемые при создании текстов, объясняющих срабатывания приемов. Способ их применения регламентируется специальными указателями приема.

Для примера описания приема по данной схеме, рассмотрим простейший прием решения уравнения $ax = b$.

Теорема приема имеет в этом случае вид:

$\forall abx(\text{число}(a)\ \&\ \text{число}(b)\ \&\ \text{число}(x) \Rightarrow (ax = b \Leftrightarrow (\neg(a = 0)\ \&\ x = b/a) \vee (a = 0\ \&\ b = 0)))$

Заголовком приема служит логический символ "второйтерм", указывающий, что теорема будет применяться для эквивалентной замены слева направо.

Фильтры приема указывают следующий контекст, в котором выполняется замена:

а) Тип решаемой задачи - на описание;

б) Преобразуемое равенство входит в условие задачи, причем это вхождение - корневое;

в) Текущий уровень сканирования, при котором происходит срабатывание приема, равен 1;

г) Выражение b не содержит неизвестных задачи, а x - содержит.

Указания компилятору уточняют следующие подробности:

а) Переменная x идентифицируется как совокупность всех множителей рассматриваемого произведения, содержащих неизвестные задачи (это автоматически предопределяет, что переменная a будет идентифицирована с выражением без неизвестных);

б) При идентификации выражения ax возможен учет стоящего перед ним знака "минус", который относится программой приема к коэффициенту a;

в) Посылки теоремы проверяются при помощи специального пакета приемов, обеспечивающего быстрое усмотрение условий вида "число(...)".

В приеме используются следующие обращения к нормализаторам:

а) Предпринимаются обращения к вспомогательному пакету приемов для разложения на множители выражений a, b. Если такие разложения не находятся сразу же, до преобразования рассматриваемого уравнения, то впоследствии может возникнуть разбор случаев, приводящий к дублирующим попыткам разложения на множители в различных подслучаях;

б) Выполняется обращение к пакету стандартизации дробных выражений для выражения b/a. Этот пакет состоит из нескольких десятков простых продукций, выполняющих, в частности, сокращение дробей, деление и умножение дробных выражений, учет констант 0 и 1, и т.п.;

в) Для уравнений $a = 0, b = 0$ выполняются обращения к пакетам приемов, стандартизирующим уравнения данного вида (сокращение левой части на множители, не обращающиеся в 0; усмотрение тождественной ложности; преобразование равенства нулю произведения в дизъюнкцию равенств нулю сомножителей, и т.п.);

г) К заменяющей дизъюнкции применяется пакет общелогических приемов (устранение логических констант "истина" и "ложь"; стандартизация конструкций с логическими связками и кванторами).

В результате выполнения указанных преобразований заменяющий терм в данном приеме приобретает достаточно устойчивый вид, не вызывающий немедленного срабатывания цепочки простейших нормализующих приемов. Заметим, что трудоемкость одного цикла сканирования задачи обычно возрастает пропорционально суммарной длине находящихся в активной области ее термов, а всего времени решения задачи - пропорционально квадрату средней такой длины. Поэтому вынесение во вспомогательные задачи преобразований подтермов сложного выражения приводит, если эти преобразования достаточно интенсивны, к значительному ускорению решения задачи. Соответственно, стандартом программирования на ГЕНОЛОГе является как можно большее использование нормализаторов.

10.2 Типы заголовков приемов

Перечислим в этом разделе типы заголовков приемов, используемые в ГЕНОЛОГе.

10.2.1 Приемы, осуществляющие тождественную либо эквивалентную замену

1. Замена подтерма с помощью тождества либо эквивалентности. Если теорема приема имеет вид $\forall x(A \to B = C)$ либо вид $\forall x(A \to (B \leftrightarrow C))$, то заголовок приема, использующего эту теорему для замены слева направо (B на C) есть "второйтерм"; заголовок для замены справа налево - "первыйтерм". Название заголовка приема здесь указывает на номер заменяющего операнда в равенстве

либо эквивалентности. Заметим, что приемы с указанными двумя заголовками (а также приемы вывода в посылках) - наиболее часто встречающиеся в базе приемов. Пример приема эквивалентной замены был приведен в предыдущем подразделе.

2. Замена группы условий задачи на описание при помощи эквивалентности. Если теорема приема имеет вид $\forall x(A \to ((B_1 \ \& \ \ldots \ \& \ B_n) \leftrightarrow C))$, где $n \geq 1$, то на ней может быть основан прием, идентифицирующий группу условий задачи на описание с утверждениями $B_1, \ldots, B_n$, и заменяющий их на утверждение C (если оно само является конъюнкцией, то прием не разбивает ее на атомарные подутверждения - это выполняет после его применения другой прием). Заголовок приема имеет вид "заменаусловия(второйтерм)", если замена выполняется слева направо; если же части приведенной выше эквивалентности поменять местами и замена будет выполняться справа налево, то заголовок приема будет иметь вид "заменаусловия(первыйтерм)". Заметим, что рассматриваемая эквивалентность может одновременно порождать и другой прием ГЕНОЛОГа - для эквивалентной замены конъюнкции $B_1 \ \& \ \ldots \ \& \ B_n$, встречающейся внутри некоторого одного терма задачи. Здесь уже будет использован заголовок "второйтерм" (соответственно, "первыйтерм"). Пример приема замены группы условий - использование теоремы $\forall xAB(x \in A \ \& \ x \in B \leftrightarrow x \in (A \cap B))$ для группировки условий на неизвестную x при известных множествах A, B.

3. Замена группы посылок задачи при помощи эквивалентности. На теореме указанного выше вида $\forall x(A \to ((B_1 \ \& \ \ldots \ \& \ B_n) \leftrightarrow C))$, где $n \geq 1$, может быть основан также прием, идентифицирующий группу посылок задачи с утверждениями $B_1, \ldots, B_n$, и заменяющий их на утверждение C. Заголовок этого приема - "заменатермов(второйтерм)" в случае замены слева направо либо "заменатермов(первыйтерм)" в случае замены справа налево (в последнем случае части указанной выше эквивалентности переставлены). Пример приема замены группы посылок - использование теоремы $\forall x(\text{число}(x) \to 0 \leq x \ \& \ \neg(x = 0) \leftrightarrow 0 < x)$ для объединения нестрогого неравенства и отрицания равенства нулю в строгое неравенство.

4. Исключение несущественных неизвестных задачи на описание. Если теорема приема имеет вид $\forall x(A \to (\exists y(B) \leftrightarrow C))$, то на ней может быть основан прием, идентифицирующий конъюнктивные члены утверждения B со всеми условиями $U_1, \ldots, U_m$ задачи на описание, содержащими неизвестные списка y. Если все эти неизвестные - несущественные (т.е. достаточно установить существование их значений, не включая информацию об этих значениях в ответ задачи; такие неизвестные $z_1, \ldots, z_k$ определяются целью "параметры $z_1 \ldots z_k$"), то теорема позволяет заменить $U_1, \ldots, U_m$ на C и устранить неизвестные y из списка неизвестных. Прием, выполняющий такое преобразование, имеет заголовок "связка". Пример приема исключения несущественных неизвестных - использование теоремы $\forall abc(\exists c(a < c \ \& \ c < b \ \& \ \text{число}(c)) \leftrightarrow a < b)$ для исключения несущественной неизвестной c, ограничения на которую сводятся к указанию числового типа значений и интервала значений.

5. Лексикографическое упорядочение операндов коммутативных операций и предикатов. Если некоторая операция (предикат) A не изменяет своего значения при любой перестановке операндов, то с ней связан прием, обеспечивающий лексикографическое упорядочение операндов. При выдаче терма на экран обычно

происходит некоторое стандартное переупорядочение операндов для прорисовки формул в их привычном виде, однако во внутреннем представлении для коммутативных логических символов постоянно применяется именно лексикографическое упорядочение. Заголовок приема, выполняющего такое упорядочение, - логический символ "лексупорядочение". Теорема приема в этом случае имеет вид "коммутативно(A)" (например, "коммутативно(умножение)").

6. Циклическая перестановка операндов для размещения на первом месте первого в лексикографическом порядке операнда (для теоремы "циклупорядочение(A)"). Если операция A не изменяет своего значения при произвольных циклических перестановках операндов, то используется прием, переносящий на первое место первый в лексикографическом порядке операнд. Заголовок приема - "циклупорядочение"; теорема его - "циклупорядочение(A)".

7. Устранение вложенных одинаковых ассоциативно - коммутативных операций. Для устранения вложенных одинаковых ассоциативно - коммутативных операций A: $A(\ldots A(\ldots)\ldots) = A(\ldots\ldots\ldots)$ - служит прием, имеющий заголовок "спускоперандов". Теорема этого приема имеет вид "коммутативно(A)".

8. Лексикографическое упорядочение элементов набора, расположенного под одноместной операцией. Иногда многоместная операция с переменным числом операндов представляется в виде одноместной операции над набором, представленным термом вида "набор($A_1 \ldots A_n$)". В тех случаях, когда значение этой операции не зависит от порядка элементов набора, используется прием, выполняющий лексикографическое упорядочение операндов $A_1, \ldots, A_n$. Заголовок этого приема - "стандупорядочение(A)"; теорема его - "список(A)".

10.2.2 Приемы для вывода следствий

1. Вывод следствий в посылках задачи. Теорема вида $\forall x_1 \ldots x_n(A_1\ \&\ \ldots\ \&\ A_m \to A_0)$ может быть использована для вывода следствий в посылках задачи. Часть утверждений $A_1, \ldots, A_m$ при этом идентифицируется с посылками; остальные утверждения этого списка - проверяются с помощью вспомогательных процедур либо служат для ввода вспомогательных обозначений. В список посылок заносится утверждение A_0. Заметим, что иногда список непосредственно идентифицируемых с посылками утверждений A_i может быть и пустым: попытка применения приема может начинаться с усмотрения встречающегося где-либо в задаче (в том числе и в условиях) терма специального вида, и далее выполняются лишь проверки истинности необходимых A_i с помощью вспомогательных процедур. Такие режимы применения приема уточняются специальными его указателями. Заголовок приема, обеспечивающего вывод в посылках задачи - "вывод". Пример приема вывода в посылках - использование теоремы $\forall ABC(B \in \text{прямая}(AC)\ \&\ l(AB) = l(BC)\ \&\ \text{разныеточки}(A, C) \to B \in \text{отрезок}(AC))$ для усмотрения принадлежности отрезку точки, равноудаленной от его концов и лежащей на той же прямой, что и эти концы.

2. Вывод следствий в условиях задачи на описание. Теорема указанного выше вида $\forall x_1 \ldots x_n(A_1\ \&\ \ldots\ \&\ A_m \to A_0)$ может быть использована не только для вывода следствий в посылках, но для вывода следствий в условиях задачи на описание. Вывод такого рода используется достаточно редко, так как загромождение списка условий задачи следствиями исходных условий нежелательно

- усложняется последующая проверка найденных значений. Как правило, совместные следствия условий и посылок извлекаются в блоке анализа задачи на описание - специально предназначенной для этого вспомогательной задаче на исследование. Часть утверждений $A_1, \ldots, A_m$ (возможно, пустая) идентифицируется с посылками либо условиями; остальные утверждения этого списка - проверяются с помощью вспомогательных процедур либо служат для ввода вспомогательных обозначений. В список условий заносится утверждение A_0. Заголовок приема, обеспечивающего вывод в условиях задачи на описание - "выводусловия". Большей частью, вывод в условиях задачи на описание используется либо для регистрации дизъюнкции, по которой будет выполняться разбор случаев, либо для регистрации утверждений, подготавливающих такой разбор случаев (например, утверждений о принадлежности значения целочисленной неизвестной некоторому конечному промежутку). Пример приема вывода в условиях задачи на описание - использование теоремы $\forall xn(x < n$ & натуральное$(x) \rightarrow \exists m(m \in \{1, \ldots, n-1\}$ & $x = m))$ для занесения в условия задачи утверждения $(m = 1 \vee \ldots \vee m = n-1)$ & $x = m$, позволяющего найти неизвестное значение x путем разбора случаев.

10.2.3 Обратный вывод для условий задачи

1. Использование кванторной импликации для обратного вывода условия задачи на описание. Теорема вида $\forall x(A_1$ & ... & $A_n \rightarrow A_0)$ - утверждения такого вида называем кванторными импликациями - может быть использована для попытки замены условия задачи на описание, идентифицированного с A_0, на группу антецедентов A_i, выделенных в описании приема указателями "подборзначений(i)". Перед попыткой проверяется истинность остальных антецедентов. Прием, реализующий эти действия, имеет заголовок "подборзначений". Применение его допустимо лишь в задачах, не требующих получения полного описания множества допустимых значений неизвестных. Чаще всего прием применяется для выбора конкретного значения неизвестной, входящей в A_0, если прочие ограничения на эту неизвестную сводятся только к указанию типа значения. Пример приема - использование теоремы $\forall ax(x = \emptyset \rightarrow x \subseteq a)$ для выбора пустого множества в качестве неизвестного подмножества x некоторого множества a.

2. Попытка использования параметрического описания для эквивалентного преобразования условия задачи на описание. Теорема вида $\forall x(A \rightarrow (B(x) \leftrightarrow \exists y(F(x, y))))$ может быть использована для замены условия задачи на описание, идентифицированного с утверждением $B(x)$, на параметрическое описание $\exists y(F(x, y))$ (в большинстве случаев $F(x, y)$ дает явное выражение x через y, но не обязательно). Такая замена реализована как переход к вспомогательной задаче, условия которой получены из условий текущей задачи заменой $B(x)$ на группу конъюнктивных членов утверждения $F(x, y)$ и присоединением новых переменных y к списку неизвестных. Если вспомогательная задача решена, то на ее ответ навешивается квантор существования по y и предпринимается редактирование полученного таким образом параметрического описания; если же ее решить не удалось, то возобновляется процесс решения текущей задачи без применения данной замены. Заголовок приема, выполняющего указанные действия - "параметризация". Заметим, что хотя теорема приема и имеет вид эквивалентности,

допустимо применение приема и в случаях, когда эквивалентность не имеет места - если в задаче не требуется получить полный ответ (например, есть цель "пример"), то достаточно лишь, чтобы имела место импликация в направлении справа налево. Формально, для простоты компиляции, и в этом случае теорема приема записывается в виде эквивалентности. Пример приема - использование теоремы $\forall ab(a \subseteq b \leftrightarrow \exists c(\text{set}(c)\ \&\ b = a \cup c))$ для представления неизвестного надмножества b множества a виде объединения a с некоторым вспомогательным множеством c, которое далее будет рассматриваться как новая неизвестная; прием применяется, если нужно лишь подобрать какой-либо пример для b.

3. Попытка решить задачу на доказательство путем идентификации ее условия с консеквентом кванторной импликации, а посылок - с антецедентами этой импликации. Если теорема имеет вид $\forall x(A \rightarrow \forall y(B_1\ \&\ \ldots\ \&\ B_n \rightarrow B_0))$, то ее можно использовать для решения задачи на доказательство, условие которой идентифицируется с B_0, а утверждения $B_1, \ldots, B_n$ - со всеми посылками, содержащими переменные списка y. При установлении истинности конъюнктивных членов утверждения A (без использования посылок, содержащих переменные списка y) с помощью средств, уточняемых в указателях приема (решение вспомогательных задач на доказательство, проверочных пакетных операторов и т.п.) выдается ответ "истина". Прием имеет заголовок "свертка". В решателе приемы такого типа используются для доказательств по индукции. Пример - использование теоремы $\forall mg(\text{натуральное}(m)\ \&\ g(m)\ \&\ \forall n(\text{натуральное}(n)\ \&\ g(n)\ \&\ 0 \leq n - m \rightarrow g(n+1)) \rightarrow \forall n(\text{натуральное}(n)\ \&\ 0 \leq n - m \rightarrow g(n)))$ для доказательства утверждения $g(n)$ при всех натуральных n, не меньших m, индукцией по n.

10.2.4 Приемы для усмотрения ответа задачи

1. Усмотрение явного описания для значения неизвестной. Для задания вида явного описания значений неизвестной x используется псевдотеорема "явное(x набор($f_1 \ldots f_n$) набор($g_1 \ldots g_m$) набор($h_1 \ldots h_k$))". Здесь $f_1, \ldots, f_n$ - все не более чем однократно встречающиеся в явном описании утверждения; $g_1, \ldots, g_m$ - все утверждения, которые могут встречаться в явном описании несколько раз. Каждая отличная от x переменная может встречаться в наборе $f_1, \ldots, f_m, g_1, \ldots, g_m$ лишь однократно; при неоднократном использовании некоторого g_i такие переменные переобозначаются в g_i на новые переменные. $h_1, \ldots, h_k$ - условия на отличные от x переменные явного описания, при которых оно непротиворечиво (набор этих условий, по возможности более полный, тем не менее не обязан гарантировать непустоту множества удовлетворяющих явному описанию значений x). Возможны случаи $n = 0; m = 0; k = 0$ - тогда вместо терма "набор(...)" используется логический символ "пустоеслово".

   Псевдотеорема набирается при помощи текстового редактора. На ней основана группа приемов, соответствующих идентификации, начинающейся с усмотрения различных утверждений списка $f_1, \ldots, f_n, g_1, \ldots, g_m$. Заголовки этих приемов имеют, соответственно, вид "ответ(f_1)", ..., "ответ(f_n)", "ответ(g_1)", ..., "ответ(g_m)". Для удобства ввода таких приемов рекомендуется, набрав текст псевдотеоремы и введя логический символ, за которым закрепляется прием, далее нажать Esc и "а" (кир.). Тогда включается цикл автоматической гене-

рации указанной серии приемов - для регистрации очередного приема следует нажимать F3, и далее - снова "a", пока новые приемы не иссякнут.

Указанное выше дублирование точек активизации приема позволяет выявлять ответ задачи сразу же, как только он появляется - в противном случае точка привязки могла бы находиться в одном из условий задачи, имеющем большой вес и таким образом находящемся вне поля зрения решателя, в то время как только что измененное условие (веса 0), после преобразования которого возникла требуемая форма ответа - не анализировалось бы с точки зрения усмотрения ответа.

Прием срабатывает при текущем уровне сканирования, равном 1; сопровождение его фильтрами не предусмотрено (компилятор игнорирует указание фильтров).

Пример приема - использование псевдотеоремы "явное(x1 набор(число(x1)меньшеилиравно(x1 x2)меньшеилиравно(x3 x1))набор(не(равно(x1 x4)))пустослово)" для усмотрения явного описания значений числовой неизвестной x1 в виде одного, двух (либо нуля) нестрогих неравенств, ограничивающих ее промежуток изменения, быть может, сопровождаемых указанием конечного числа отбрасываемых точек.

2. Выдача ответа задачи на описание с учетом контекста. Хотя приемы выдачи ответа задачи на описание при усмотрении явного описания значений ее неизвестной являются наиболее употребительными, они обладают одним недостатком - применение их не может быть ограничено с учетом целевой установки задачи. Поэтому введена другая форма усмотрения и выдачи ответа задачи на описание. В этом случае теорема приема имеет вид A_1 & ... & A_n - эта конъюнкция определяет вид списка условий задачи, при котором следует выдавать ответ. Специальные указатели приема ("смответ", "серия") уточняют, в котором из утверждений A_i выбирается точка привязки - с рассмотрения ее будет начата попытка применения приема, а также выделяют те A_i, которые могут идентифицироваться с несколькими условиями задачи. Прием имеет заголовок "ответзадачи". В отличе от приема "ответ", он допускает обычное использование фильтров для уточнения контекста срабатывания. Пример приема - использование описания число(x) & $a \leq x$ & $x \leq b$ & $f(x) \leq y$ & $y \leq g(x)$ & число(y) вида ответа при решении системы неравенств с двумя неизвестными x, y. Фильтры приема уточняют, что выражения a, b не содержат неизвестных, а выражения $f(x), g(x)$ - не содержат неизвестной y.

10.2.5 Перенесение во внешнюю задачу на описание утверждений из блока анализа

В процессе вывода совместных следствий из условий и посылок задачи на описание (такой вывод реализуется во вспомогательной задаче на исследование, называемой блоком анализа данной задачи на описание) может сложиться необходимость передачи некоторых ценных следствий во внешнюю задачу на описание. Это выполняется приемом с заголовком "замещениеусловий"; псевдотеорема приема имеет вид "замещениеусловий(A)", где A - утверждение, идентифицируемое с посылкой блока анализа, переносимой во внешнюю задачу на описание. При перенесении указанного

утверждения предпринимается попытка исключить возможно большее число условий задачи на описание, если они становятся следствиями A и некоторых (дополнительно переносимых) простых утверждений, имеющих по отношению к A сопровождающий характер. Специальный указатель приема ("обрывзадачи") может сразу же по перенесении A прервать сканирование блока анализа и возобновить сканирование внешней задачи на описание.

Пример приема - использование псевдотеоремы "замещениеусловий(эллипс(x1))" для передачи в список условий внешней задачи на описание, решаемой с целью характеризации типа кривой второго порядка x1, найденного типа "эллипс(x1)" этой кривой. Прием не имеет указателя "обрывзадачи", и после передачи указанного утверждения анализ кривой продолжается.

10.2.6 Ввод и удаление комментариев

Предусмотрены приемы "управляющего" характера, которые не вносят каких-либо изменений в логические структуры данных задачи, а выполняют ввод, удаление либо изменение ее комментариев. Псевдотеорема такого приема имеет вид "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то контекст(A_0))", если идентификация начинается с усмотрения вхождения терма A_0, либо "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то $\emptyset$)", если идентификация начинается с усмотрения некоторой посылки либо условия (в зависимости от указания в фильтрах приема) A_i. Собственно действия с комментариями обеспечиваются указателями приема так же, как это происходит при применении невырожденных приемов, изменяющих логические структуры данных. Приемы данного типа имеют заголовок "замечание". Примеры использования указателей для определения преобразований комментариев задачи будут приведены в разделе, посвященном таким указателям.

10.2.7 Пакетные операторы и их приемы

Процедура сканирования задачи оказалась чрезвычайно эффективным "диспетчером", организующим совместные действия по решению задачи всего многообразия приемов, накопленных при обучении. По существу, это - механизм "внутреннего логического зрения" интеллектуальной системы, позволяющий ей планировать свои действия на основе прямого доступа к огромному массиву знаний и умений.

Однако, этот механизм имеет существенный недостаток - полный просмотр и осмысление всего "поля задачи" оказывается весьма трудоемкой процедурой. Как показывает трассировка, основную часть времени решатель, сканирующий задачу, затрачивает на обработку начальных отрезков фрагментов программ встречаемых им в задаче логических символов - то есть в каком-то смысле "впустую". На этом фоне какая-либо оптимизация либо замедление вычислений, выполняемых фактически реализуемыми приемами, оказывается практически незаметной - она не влияет на итоговое время решения задачи.

Наоборот, весьма ощутимо сказывается на времени работы решателя уменьшение числа циклов сканирования за счет укрупнения процедур, выполняемых отдельными приемами. Оптимизация решателя оказалась связанной с максимальным увеличением объема вычислений и преобразований, выполняемых приемом вне цикла сканирования. Поведение его как-бы состоит из циклов "планирования" (поиск наилучшего приема при сканировании) и реализации найденного плана. Чем больше необходимых

действий вовлекается в этот план, тем меньше будет число сканирований, и тем больше - быстродействие решателя.

Одним из наиболее эффективных средств для "укрупнения" приемов сканирования задачи оказалось использование в них пакетов "локальных" приемов, применяемых уже не в контексте сканирования всей задачи, а для обработки каких-либо выделенных подтермов. Такие объединения локальных приемов получили название пакетных операторов. Эффективность их применения объясняется малым числом используемых приемов и обычно малым по сравнению со всей задачей размером обрабатываемых термов. Приемы пакетного оператора упорядочены по уровням срабатывания, и на каждом уровне организованы в древовидную процедуру, позволяющую отсекать заведомо не реализуемые ситуации. Посимвольного просмотра обрабатываемых термов здесь не происходит - поиск приема выполняется однократным прохождением программы пакетного оператора.

Приемы пакетных операторов (как и приемы сканирования задачи) задаются на ГЕНОЛОГе и затем компилируются в программы ЛОСа. В них допускаются обращения к другим пакетным операторам и рекурсивные обращения. Возникло 4 основных типа пакетных операторов, аналогичных 4 основным типам задач (на доказательство, описание, преобразование и исследование):

1. Аналогом задачи на доказательство является проверочный оператор, получающий в качестве входных данных проверяемое утверждение (точнее, набор выражений, подставляемых вместо переменных в "шаблон", задающий вид такого утверждения); список посылок, в предположении истинности которых проводится проверка, а также список комментариев управляющего характера. Помимо проверки истинности, проверочный оператор также выдает список посылок, фактически использованных при проверке. Пример проверочного оператора - оператор "усмменьше", получающий в качестве входных данных: выражения a, b, для которых проверяется условие $a < b$; список посылок, в предположении истинности которых происходит проверка; набор комментариев (в частности, в нем могут содержаться ограничения на применяемые при проверке средства).

2. Аналогом задачи на описание является пакетный оператор, названный синтезатором. Он предназначен для решения задач на описание с единственным условием, имеющим строго ограниченный вид, определяемый некоторым "шаблоном". Часть переменных шаблона выделена как "известные", а часть - как "неизвестные". Синтезатор определяет значения неизвестных, получая в качестве входных данных: набор выражений, подставляемых вместо "известных" переменных в шаблон; список посылок, в предположении истинности которых предпринимается подбор значений неизвестных, и набор комментариев, имеющих управляющий характер (целевая установка и т.п.). Результатом работы синтезатора служат набор искомых значений неизвестных и список фактически использованных посылок. В зависимости от своего типа, синтезатор либо находит единственный пример допустимых значений неизвестных, либо перечисляет такие примеры. Пример синтезатора - оператор "верхняяоценка", реализующий утверждения вида $a \leq x$ для известной переменной a и неизвестной x, который перечисляет представляющие интерес (с учетом комментариев) константные верхние оценки x выражения a.

3. Аналогом задачи на преобразование является пакетный оператор, названный нормализатором. Он получает в качестве входных данных преобразуемый терм

(выражение либо утверждение), список посылок, в предположении истинности которых выполняются преобразования, и набор комментариев, имеющих управляющий характер. Прием нормализатора выполняет некоторое преобразование его "условия". Это преобразование (в зависимости от предназначения нормализатора) не обязательно должно быть тождественным (например, при получении асимптотической оценки какого-либо выражения). В отличие от проверочного оператора и синтезатора, представляющих собой операторы ЛОСа, нормализатор является операторным выражением - его значением служит возникший после цепочки преобразований терм. Чтобы определить, какие из посылок фактически были использованы при его получении, в комментариях нормализатора используется информационный элемент "выводимо A", накапливающий такие посылки в наборе A. Пример нормализатора - операторное выражение "нормдробь", осуществляющее общую стандартизацию дробных выражений (отбрасывание знаменателя 1, деление дроби, деление на дробь, вынесение минуса из числителя либо знаменателя, и т.п.).

4. Аналогом задачи на исследование является пакетный оператор, названный анализатором. Он обрабатывает копию списка посылок текущей задачи Z, оформленную в виде вспомогательной задачи на исследование Z'. Анализатор не обращается для сканирования Z' к основной базе приемов, а использует свои собственные приемы. Так как их существенно меньше, то удается за сравнительно небольшое время получить значительное число следствий. Иэ этих следствий отбираются лишь ценные для внешней задачи Z, которые по окончании работы анализатора переносятся в ее список посылок. Отбор выполняется приемами анализатора, помечающими нужные посылки комментарием "внешвывод". Обращение к анализатору обеспечивается специальными приемами, указывающими лимит времени его работы. Передача отобранных утверждений происходит по исчерпании этого лимита. При получении особо важных следствий возможен немедленный обрыв работы. Входными данными обращения к анализатору служат: задача Z, максимальный уровень используемых приемов и целевая установка, передаваемая задаче Z'. Вспомогательная задача Z' создается самим анализатором. Он же реализует передачу отобранных утверждений в задачу Z.

Кроме указанных выше основных четырех типов пакетных операторов, используется еще один, вспомогательный. В фильтрах нескольких различных приемов иногда возникает необходимость применения одного и того же условия, представляющего собой конъюнкцию либо дизъюнкцию большого (и возрастающего по мере обучения) числа описаний частных ситуаций. Здесь становится целесообразным создание реализованного на ГЕНОЛОГе пакетного оператора, выполняющего проверку этого условия либо определение значений его "выходных" переменных. Этот оператор, в отличие от перечисленных выше, относится к "структурному" уровню - он проверяет условие, относящееся не к объектам предметной области, а к логическим структурам данных, имеющимся в задаче. Хотя, в принципе, для задания операторов "структурного" уровня мог бы с равным успехом использоваться ЛОС (ведь он представляет собой логический язык для формулировки условий на текущую структуру данных), в решающих правилах приемов часто возникает необходимость задавать вид термов "предметного" уровня - и здесь существенное упрощение записи создает ГЕНОЛОГ. Пакетный оператор данного, пятого, типа, называется оператором фильтра.

Прием проверочного оператора

Прием проверочного оператора обычно имеет теорему вида $\forall x(A_1 \,\&\, \ldots \,\&\, A_n \to A_0)$. Здесь утверждение A_0 идентифицируется с проверяемым условием (оно само должно иметь вид, согласованный с шаблоном проверочного оператора). Применение приема заключается в проверке (с учетом указателей приема) утверждений $A_1, \ldots, A_n$ и выходе из проверочного оператора (как из программы на ЛОСе) по значению "истина"; выходной переменной оператора при этом присваивается список фактически использованных при проверке $A_1, \ldots, A_n$ посылок. Заголовок приема имеет вид "спуск(B)", где B - название проверочного оператора. Пример приема проверочного оператора - использование теоремы $\forall x(\text{число}(x) \,\&\, \neg(x = 0) \to \neg(-x = 0))$ для проверки отличия от нуля значения выражения, начинающегося со знака "минус". Заголовок данного проверочного оператора - "усмне0"; шаблон проверяемых им утверждений - $\neg(a = 0)$.

Прием синтезатора

Прием синтезатора имеет теорему вида: $\forall x(A_1 \,\&\, \ldots \,\&\, A_n \to A_0)$. Здесь утверждение A_0 идентифицируется с реализуемым условием; применение приема заключается в проверке либо реализации (с учетом указателей приема) утверждений $A_1, \ldots, A_n$ и выдаче найденных значений переменных; дополнительной выходной переменной оператора при этом присваивается список фактически использованных при реализации посылок. Заголовок приема имеет вид "значение(B)", где B - название синтезатора. Пример приема синтезатора - использование теоремы $\forall abc(0 < c \,\&\, a \le b \,\&\, 0 \le a \to a^c \le b^c)$ для получения верхней оценки степенного выражения a^c, у которого показателем степени служит положительная константа, а основание степени неотрицательно. Название синтезатора - "верхняяоценка"; значение b определяется из антецедента $a \le b$, представляющего собой рекурсивное обращение к тому же самому синтезатору. Шаблон реализуемых синтезатором утверждений - $x \le y$, где переменная x - входное (известное) выражение; переменная y - выходная (не известная).

Прием нормализатора

Приемы нормализатора бывают следующих типов:

1. Если теорема представляет собой тождество $\forall x(A \to (B = C))$ либо эквивалентность $\forall x(A \to (B \leftrightarrow C))$, то она может быть использована для создания приема нормализатора, выполняющего замену слева направо либо справа налево. В первом случае заголовок приема имеет вид "замена(второйтерм N)"; во втором случае - "замена(первыйтерм N)". Здесь N - название нормализатора. Пример приема такого типа - использование теоремы $\forall_{abc}(b/(c/a) = ab/c)$ в нормализаторе "нормдробь" для исключения делений на дробное выражение.

2. Возможно создание нормализаторов, в которых предусмотрен разбор случаев при выполнении преобразований. Обычно потребность в таком разборе выявляется не сразу, а после некоторой цепочки преобразований. Разбор случаев инициируется специальным приемом, организующим откат к циклу преобразований исходного терма в различных подслучаях. В рамках рассмотрения отдельного подслучая возможна инициализация вложенного разбора случаев - без ограничений на глубину вложенности. По завершении рассмотрения всего

дерева подслучаев формируется объединяющий их терм - с помощью логической конструкции "вариант($A\ t_1\ t_2$)"; этот терм и выдается как результат применения нормализатора. Примером нормализатора, допускающего указанный разбор случаев, является нормализатор вычисления пределов "нормпредел".

Заметим, что при откатах, вызванных разбором случаев, теряются ранее проведенные вычисления. Однако, для нормализаторов, выполняющих сколь-нибудь трудоемкие вычисления, предусмотрено автоматическое сохранение результатов обращений к ним в специальном буфере. При каждом последующем обращении к нормализатору прежде всего проверяется наличие в буфере готового результата, и если есть, то он сразу и выдается. Это позволяет экономить при откатах, фактически извлекая из буфера готовые результаты там, где они не зависели от определяющего подслучай утверждения. Буферы, сохраняющие результаты нескольких последних обращений, используются и для некоторых других типов пакетных операторов; их применение позволило при программировании на ГЕНОЛОГе практически не заботиться о предотвращении дублирований, возникающих при обращении из различных приемов к одним и тем же вспомогательным вычислениям.

Возвращаясь к разбору случаев в нормализаторах, укажем вид псевдотеорем, применяемых для инициализации этого разбора: $\forall_x(A \to (B =$ разборслучаев$(C)))$. Здесь дизъюнкция C определяет необходимые подслучаи; терм B идентифицируется с текущим преобразуемым термом либо (в зависимости от типа нормализатора) с его подтермом. Заголовок приема здесь по-прежнему имеет вид "замена(второйтерм N)", где N - название нормализатора, хотя в действительности прием не выполняет никакой замены. Пример приема данного типа - использование псевдотеоремы

$\forall_{xfabc}(lim_{x\to b\backslash a}f(x) = 0 \to lim_{x\to b\backslash a}f(x)^c =$ разборслучаев $(0 < c\ \vee\ c < 0\ \vee\ c = 0))$

при инициализации разбора случаев для знака показателя степени, если предел основания степени равен нулю.

3. Для устранения вложенных одинаковых ассоциативно - коммутативных операций f: $f(\ldots f(\ldots)\ldots) = f(\ldots\ldots\ldots)$ - служит прием нормализатора, имеющий заголовок "замена(спускоперандов B)", где B - название нормализатора. Теорема этого приема имеет вид "коммутативно(f)".

4. Для лексикографического упорядочения операндов коммутативной операции f в нормализаторе используется прием теоремы "коммутативно(f)", имеющий заголовок "замена(лексупорядочение B)", где B - заголовок нормализатора.

5. В тех случаях, когда требуется ввести, удалить либо изменить комментарий нормализатора, не предпринимая изменений его преобразуемого терма, используется прием с заголовком "замена(замечание N)", где N - заголовок нормализатора. Теорема такого приема имеет вид "длялюбого($x_1 \ldots x_n$ если $A_1 \ldots A_m$ то контекст(A_0))", причем идентификация начинается с усмотрения вхождения в преобразуемый терм терма A_0. Необходимые действия с комментариями определяются предназначенными для этого указателями приема.

Прием анализатора

Как и приемы вывода, приемы анализаторов имеют теорему вида $\forall x(A_1 \,\&\, \ldots \,\&\, A_n \rightarrow A_0)$, при помощи которой осуществляется вывод следствия A_0; способ обработки утверждений $A_i (i = 1, \ldots, n)$ при этом уточняется указателями приема (возможны непосредственная идентификация такого утверждения с одной из посылок; проверка его истинности при помощи проверочного оператора либо задачи на доказательство; использование для ввода вспомогательных обозначений, и т.п.). Заголовок приема - "внутрвывод(B)", где B - название анализатора.

Прием оператора фильтра

Прием оператора фильтра имеет псевдотеорему "блок" (на экран при просмотре описания приема вместо этой псевдотеоремы выводится текст концевого пункта оглавления, в котором расположены задающие формат оператора фильтра приемы справочников "блок" и "блокпроверок"; этот текст обычно напоминает смысл входных и выходных переменных оператора). Заголовок приема - "контекст(A)", где A - название оператора фильтра. При отсутствии выходных переменных оператор получает значение "истина", если выполнены условия на текущий контекст, определяемые фильтрами хотя бы одного из его приемов; при наличии выходных переменных - выдает их значения при срабатывании какого-либо из его приемов. В зависимости от типа оператора, при его работе, кроме входных переменных, могут быть доступны другие структуры данных (текущая задача и координата вхождения в нее - если обращение к оператору происходит из приема сканирования задачи, либо список комментариев внешнего пакетного оператора - если обращение имеет место из такого оператора, и т.п.).

10.2.8 Вычислительные пакеты

Рассмотренные выше пакетные операторы предназначены для работы с логическими структурами данных. Однако, ГЕНОЛОГ можно использовать и для создания программ, работающих с "обычными" техническими структурами данных - числами, массивами, таблицами и т.п. Эти программы, разумеется, тоже имеют своими источниками теоремы или группы теорем, так что естественным образом возникает задача компиляции их непосредственно с теоремного уровня.

Для задания обычных, нелогических вычислений на ГЕНОЛОГе используются так называемые вычислительные пакеты. Они компилируются в программу оператора ЛОСа и состоят обычно из небольшого числа вычислительных "приемов" или "продукций".

В приемах логического характера четко разделены структурный и предметный уровни. Обрабатываемая информация представлена на структурном уровне, в виде формул, косвенным образом ссылающихся на объекты и отношения предметного уровня. Поэтому возникает необходимость в двухуровневой конструкции приемов: на предметном уровне записывается теорема, обеспечивающая возможность выполняемых приемом преобразований, а на структурном уровне записываются условия, при которых это преобразование представляется целесообразным. Но для вычислительного приема предметный и структурный уровни сливаются. Здесь обрабатываемая информация состоит из технических структур данных, которые однозначно определяют объекты и отношения между ними - являются, своего рода, именами этих

объектов и отношений. Поэтому вся управляющая часть приема может быть полностью погружена в предметный уровень - в виде дополнительных посылок теоремы, определяющих целесообразность ее применения. Соответственно, приемы вычислительных пакетов имеют (за редкими исключениями, связанными с использованием комментариев) пустой список фильтров.

Выделим здесь два типа вычислительных пакетов - пакеты продукций и технические анализаторы.

Пакет продукций можно рассматривать как вычислительный аналог синтезатора. Прием его имеет заголовок "продукция(A)", где A - заголовок пакета. Если список термов и логических символов, задающих формат пакета продукций, имеет элемент "эквивалентно", то приемы этого пакета работают в режиме эквивалентных преобразований группы входных данных, приводя ее к такому виду, для которого значения выходных переменных вычисляется непосредственно. Такой режим аналогичен режиму работы пакетных нормализаторов. При отсутствии элемента "эквивалентно" каждый прием пакета продукций определяет значения выходных переменных за один шаг, как и у пакетного синтезатора. Разумеется, этот шаг может включать в себя рекурсивные обращения к вспомогательным вычислениям.

Технический анализатор выполняет разметку некоторой сложной структуры данных, снабжая ее элементы наборами значений параметров, характеризующих эти элементы либо связи между ними. Разметка осуществляется в режиме сканирования технической структуры данных. Список параметров определен форматом анализатора; хранятся они во вспомогательной структуре данных, организованной подобно анализируемой. Режим работы технического анализатора аналогичен режиму работы пакетного анализатора, так как представляет собой, по существу, процесс вывода следствий вплоть до полного их исчерпания. Прием технического анализатора имеет заголовок "знач(A)", где A - название анализатора.

Чтобы перейти от рекурсивного характера вычислений, свойственного логическим пакетным операторам, к вычислениям итерационным, в описаниях формата вычислительных пакетов используются указатели, регулирующие объединение программ различных приемов в общую программу пакета.

10.2.9 Простейшие приемы непосредственного усмотрения истинности либо ложности

Для усмотрения истинности утверждений вида $f(A)$, где f определяет один из базисных типов объектов (число; множество; функция; точка и т.п.), вводятся приемы, усматривающие этот тип для значений выражения A из анализа заголовка A. Псевдотеорема такого приема имеет вид "родобъекта(f)"; заголовок - "родобъекта".

Для усмотрения противоречивых указаний на тип объекта вводится также прием с заголовком "различны" (его псевдотеорема - та же, т.е. "родобъекта(f)").

10.2.10 Приемы справочников

Количество различных справочников, используемых в системе, приближается к сотне. Задание их приемов на ГЕНОЛОГе обычно весьма простое - фильтры и указатели приема здесь, как правило, отсутствуют (в таких ситуациях, для обеспечения работы компилятора ГЕНОЛОГа, приходится вводить какой-либо незначащий логический символ в качестве указателя - как правило, используется логический символ "пусто"

либо "справка"). Для каждого нового справочника в компилятор ГЕНОЛОГа вводится свой (обычно коротенькая) программа трансляции - копируется с аналогичной программы другого справочника, с минимальными изменениями. Функционирование справочников достаточно подробно описано в информации о логических символах - названиях этих справочников (доступной, например, по F2 из редактора программ ЛОСа); названия справочников перечислены, например, в информации по заголовкам приемов ГЕНОЛОГа, доступной по Ctr-з (кир.) из просмотра описания какого-либо приема ГЕНОЛОГа. Поэтому мы ограничимся здесь лишь рассмотрением справочников, используемых для задания формата пакетных операторов. Приемы этих справочников создаются перед созданием самого пакетного оператора; после того, как они введены, можно переходить ко вводу и компиляции его приемов. В интерфейсе редактора приемов предусмотрены средства автоматической инициализации пакетного оператора.

Задание формата проверочных операторов

Формат проверочного оператора определяется справочником "легковидеть". Теорема приема этого справочника имеет вид "легковидеть($f(x_1 \dots x_n \ x_{n+1} \ x_{n+2} \ x_{n+3})$ уровень$(N)T$)". Здесь $f(x_1 \dots x_n \ x_{n+1} \ x_{n+2} \ x_{n+3})$ - вид обращения к оператору; T - шаблон проверяемого утверждения. $x_1, \dots, x_n$ - все свободные переменные утверждения T; Входной переменной x_{n+1} передается список утверждений, истинность которых может быть использована при проверке (посылок); переменной x_{n+2} - список комментариев. Выходной переменной x_{n+3} присваивается набор посылок, фактически использованных при усмотрении. N - число уровней срабатывания оператора f (при $N = 1$ указатель "уровень(N)" может отсутствовать).

Кроме обычных проверочных операторов, можно создавать операторы усиленной проверки. Их формат определяется справочником "проверка", аналогичным справочнику "легковидеть". Теорема приема этого справочника имеет вид "проверка($f(x_1 \dots x_n \ x_{n+1} \ x_{n+2} \ x_{n+3})$ уровень$(N)T$)", и далее - так же, как у обычного проверочного оператора.

Задание формата синтезаторов

Формат синтезатора определяется справочниками "синтезатор" и "значения". Оба этих справочника основаны на одной и той же псевдотеореме вида "синтезатор($f \ T$ вход$(x_1 \dots x_n)$ выход$(y_1 \dots y_k) \ A_1 \dots A_m$)". Здесь f - заголовок синтезатора (логический символ); T - конъюнкция утверждений, связывающих значения входных и выходных переменных (T не имеет двух различных вхождений одной и той же переменной); $x_1, \dots, x_n$ - список входных переменных; $y_1, \dots, y_k$ - список выходных переменных. $A_1, \dots, A_m$ - информационные термы, уточняющие тип синтезатора (см. ниже). Обращение к синтезатору имеет вид $f(x_1 \dots x_n \ x_{n+1} \ x_{n+2} \ y_1 \dots \ y_k \ y_{k+1})$, где x_{n+1} - список посылок; x_{n+2} - список комментариев; y_{k+1} - список утверждений, использованных в качестве обоснования результата.

Используются следующие типы информационных термов A_i:

1. уровень(N). N есть число уровней срабатывания приемов синтезатора;

2. перечисление. Синтезатор определяет перечисление значений выходных переменных;

3. коррекцияпосылок. Перед обращением к синтезатору из приема сканирования задачи в комментариях синтезатора создается накопитель (коррекцияпосылок C). Набор C (вначале пустой) заполняется синтезатором тройками (B_1 B_2 B_3), где B_1 - набор дополнительных посылок, введенных для того, чтобы обеспечить непосредственное усмотрение принадлежности области допустимых значений входящих в результат обращения к синтезатору выражений; B_3 - список исходных посылок, из которых были выведены посылки списка B_1; B_2 - либо 0, либо подмножество списка B_3, эквивалентное списку B_1 в предположении истинности остальных утверждений списка B_3.

Задание формата нормализаторов

Формат нормализатора определяется справочником "быстрпреобр". Псевдотеорема приема этого справочника имеет вид "быстрпреобр(f $U_1 \ldots$ U_n)", где f - заголовок нормализатора; $U_1, \ldots, U_n$ - указатели типа нормализатора (см. ниже). Кроме справочника "быстрпреобр", для нормализаторов общей стандартизации термов с заголовком B создается также справочник "нормализатор". Псевдотеорема приема такого справочника имеет вид "нормализатор(B f)". Данный справочник нужен для определения названия f нормализатора по заголовку B преобразуемого терма. Заметим, что такое название всегда имеет вид "нормB"("нормплюс", "нормминус", нормстепень" и т.д.). Кроме приема справочника "нормализатор", обычно вводится также прием справочника "ключоператора" с псевдотеоремой "ключоператора(f B)" - он решает обратную задачу определения B по f. Заголовки приемов всех трех справочников совпадают с названием этих справочников.

Используются следующие указатели типа нормализатора:

1. "уровень(A)" - A есть логический символ, равный числу различных уровней срабатывания приемов нормализатора. Отсутствие такого терма означает, что все приемы срабатывают на одном и том же уровне (рекомендуется вводить хотя бы два различных уровня).

2. "списокпосылок" - обращение к нормализатору имеет вид $f(A_1\ A_2\ A_3)$, где A_1 - преобразуемый терм; A_2 - набор посылок, в предположении истинности которых выполняются преобразования; A_3 - набор комментариев, включающий, в частности, набор (выводимо B); B - накопитель утверждений списка A_2, фактически использованных в процессе преобразований. Если элемент "списокпосылок" отсутствует, то обращение к нормализатору имеет вид $f(A)$, где A - преобразуемый терм, либо (см. ниже случай указателя "неизвестные") вид $f(A\ B)$.

3. "корень" - приемы применяются только к корневому вхождению. Это - наиболее распространенный тип нормализаторов (в частности, к нему относятся все нормализаторы общей стандартизации). Во-первых, здесь происходит ускорение поиска приема за счет рассмотрения единственной "точки привязки" - корня преобразуемого терма. Во-вторых, упрощается ввод сопровождающих утверждений для указания области допустимых значений новых термов, возникающих в процессе преобразований (фактически такое сопровождение пока предусмотрено только для нормализаторов "корневого" типа, и при работе с числовыми выражениями, за редкими исключениями, используются только они). Недостаток таких нормализаторов - затруднения при усмотрении и реализации преобразований некорневых подтермов. Обычно эти затруднения преодолеваются за счет обработки соответствующими корневыми нормализаторами

всех операций, упомянутых в заменяющем терме теоремы приема, а также перехода от тождества $A = B$ к тождеству вида $g(\ldots A \ldots) = g(\ldots B \ldots)$, если корневой контекст $g(\ldots\ldots)$ некорневого преобразования $A = B$ прослеживается однозначно либо с малым числом вариантов.

4. "1" - результат выдается после однократного применения любого приема.

5. "неизвестные" - обращение к нормализатору имеет вид $f(A_1\ A_3)$ (при отсутствии указателя "списокпосылок") либо вид $f(A_1\ A_2\ A_3)$; наличие в A_3 элемента (неизвестные $x_1 \ldots x_n$) означает, что переменные $x_1, \ldots, x_n$ суть неизвестные той внешней задачи, из которой произошло обращение к нормализатору, причем если эта задача - на описание, то преобразуемый терм относится к ее условиям. Компилятор ГЕНОЛОГа автоматически обеспечивает создание такого элемента в наборе A_3 при обращениях к нормализаторам данного типа.

6. "стандупорядочение" - перед выдачей результата осуществляется обращение к операторному выражению "стандупорядочение", обеспечивающему стандартное упорядочение операндов операций - там, где допустимы простейшие типы перестановки операндов.

7. "разборслучаев" - предусмотрена нормализация с откатами для разбора случаев.

8. "глуб" - в комментариях нормализатора формируется элемент (глуб N), у которого N есть число внешних обращений к данному нормализатору в цепочке обращений к нормализаторам.

9. "группировка" - нормализатор, работающий в режиме отбора нескольких наиболее упрощающих выражение преобразований непересекающихся его подтермов и последующей их одновременной реализации.

10. "контрольнормализации" - используется буфер "контрольнормализации", сохраняющий результаты нескольких последних обращений к нормализатору. При трассировке "по шагам решения задачи" обращения к таким нормализаторам выводятся на экран и происходит показ их пошаговых действий. Использование буфера позволяет при программировании не слишком заботиться о экономии на повторных обращениях к обработке одного и того же терма, происходящих из различных приемов - такие обращения обычно идут друг за другом достаточно компактной группой, и размеров буфера (50 позиций) оказывается вполне достаточно для их отслеживания.

11. "коррекцияпосылок" - перед обращением к нормализатору из приема сканирования задачи в комментариях нормализатора создается накопитель (коррекцияпосылок A). Набор A (вначале пустой) заполняется тройками $(B_1\ B_2\ B_3)$, где B_1 - набор дополнительных посылок, введенных для сопровождения результата обращения к нормализатору. Эти посылки позволяют непосредственно усматривать принадлежность области допустимых значений (о.д.з.) подвыражений результата. B_3 - список исходных посылок, из которых были выведены посылки списка B_1; B_2 - либо 0, либо подмножество списка B_3, эквивалентное списку B_1 в предположении истинности остальных утверждений списка B_3. Использование указанного накопителя совершенно необходимо в тех случаях, когда предметная область требует аккуратного сопровождения термов по о.д.з. (например, в

элементарной алгебре), причем выполняемые нормализатором преобразования способны привести к появлению термов с неочевидным в исходном контексте выполнением условий на о.д.з.

Задание формата анализаторов

Формат анализатора определяется справочником "анализатор". Псевдотеорема приема этого справочника имеет вид "анализатор(f S)", где f - логический символ, являющийся названием анализатора, S - либо отсутствует, либо представляет собой один из логических символов "корень", "полный". Если S отсутствует, то анализатор сканирует все вхождения логических символов в посылки своей вспомогательной задачи, обращаясь на каждом уровне сканирования только к посылкам, вес которых не превосходит этого уровня. Если S есть символ "корень", то сканирование ограничивается лишь корневыми точками посылок. Наконец, если S есть символ "полный", то сканирование ведется по корневым точкам, а веса посылок при сканировании игнорируются.

Задание формата операторов фильтра

Формат оператора фильтра определяется справочниками "блок" и "блокпроверок". Псевдотеорема приема этих справочников - общая; она имеет вид "блок(A вход(B_1 $\dots$ B_n) выход(B_{n+1} $\dots$ B_m) C_1 $\dots$ C_k)". Здесь A - заголовок оператора; обращение к оператору имеет вид $A(y_1 \dots y_p\ x_1 \dots x_m)$. Каждое B_i - указатель типа данных соответствующей входной либо выходной переменной x_i (лог. символы "терм", "вхождение", "логсимвол", "переменная"); $y_1, \dots, y_p$ - дополнительные входные переменные, значения которых автоматически определяются (как это указано ниже) из текущего контекста. Если выходных переменных нет, то терм "выход($B_{n+1} \dots B_m$)" отсутствует. $C_1, \dots, C_k$ - указатели типа оператора структурного уровня.

Концевой пункт оглавления базы приемов, в котором хранятся указанные два приема справочников "блок" и "блокпроверок", располагается первым в подразделе, охватывающем приемы заданного оператора фильтра A. Текст этого пункта для удобства программирования автоматически выводится на экран вместо псевдотеоремы "блок" всех приемов оператора A, и его следует составлять таким образом, чтобы из него извлекалась информация о назначении вход-выходных переменных x_i.

Указатели типа оператора фильтра таковы:

1. Указатель текущего контекста, в котором происходит обращение к оператору - лог.символ "пусто" (число p равно 0), либо "решить" (текущий контекст - сканирование задачи, и тогда $p = 5$ - для программных переменных x1 - x5, определенных при сканировании), либо "замена" (текущий контекст - пакетный нормализатор, и тогда $p = 3$ - для программных переменных x1 - x3 этого нормализатора).

2. "перечисление" - указатель на наличие режима перечисления без повторений выходных наборов.

10.2.11 Приемы вывода теорем

Кроме базы приемов, в логической системе имеется также база теорем. Она необходима как источник, из которого берутся новые приемы. В свою очередь, источником

новых теорем является процедура логического вывода, архитектура которой будет описана в третьей книге данной монографии. При логическом выводе теорем используются возможности решателя, обращения к которому происходят из приемов вывода теорем. Такие приемы отличаются от обычных правил вывода прежде всего тем, что в них содержатся элементы постановки задач для решателя и элементы формирования гипотез, проверяемых решателем. Приемы вывода теорем активизируются через справочник "теорема" в процессе сканирования некоторой "текущей" теоремы, следствия из которой нужно получать. Все они реализованы на ГЕНОЛОГе и имеют заголовок "теорема". Большинство их сами основаны на теоремах какой-либо отдельной предметной области, и таким образом вывод приобретает "многоместный"характер. Для автоматического синтеза приемов вывода используется еще один реализованный на ГЕНОЛОГе справочник - "логвывод". Теоремы его приемов представляют собой шаблоны создания по теоремам соответствующих им одноместных правил вывода. Подробнее обо всех этих приемах будет рассказано в третьем томе.

10.3 Типы фильтров приемов

Логический язык, на котором формулируются фильтры приема, относится к уровню "структур данных" - в этом он весьма близок к ЛОСу. Отличие от ЛОСа состоит в том, что при задании фильтра приходится ссылаться на термы задачи через посредство "теоремных" переменных, идентифицированных с этими термами (иногда - с переменными либо логическими символами); при использовании ЛОСа такая ссылка осуществляется непосредственно, через программные переменные, которым заранее присваивается необходимое значение. С некоторым приближением, можно считать, что язык для задания фильтров - это "ЛОС в теоремных переменных". Многие операторы и операторные выражения, используемые для записи фильтров, попросту копируют соответствующие операторы и операторные выражения ЛОСа; иногда при этом удается опускать ряд их входных параметров, определяемых в контексте "по умолчанию".

При задании фильтров можно использовать, наряду с логическими связками, также и квантор существования (квантор общности сводится к нему с помощью отрицаний). Однако, этот квантор представляется здесь с помощью специальной конструкции "контекст(...)" (см. подробнее подраздел "Логические связки и квантор существования"), в которой опущена кванторная приставка - по умолчанию к ней относятся все переменные, встречающиеся внутри данной конструкции, не определенные к моменту ее рассмотрения внешним образом (в частности, не входящие в теорему). Такое соглашение укорачивает записи и практически не усложняет понимания их смысла. Другое важное отличие от обычного квантора существования состоит в том, что внутри конструкции "контекст(...)" могут использоваться условия, идентифицирующие термы задачи с некоторыми заданными термами, и в этом случае наряду с "чисто логическими" записями здесь будут встречаться технические указатели, уточняющие способ идентификации.

Как правило, порядок размещения фильтров в описании приема несущественен - компилятор размещает их программы в программе приема "по-своему", стараясь обеспечить определяемое ими отсечение как можно раньше. Однако, существует возможность влиять на размещение программы фильтра в программе приема (иногда это все же позволяет уменьшить "холостой ход" приема) - путем использования спе-

циальных сопровождающих фильтр пометок либо путем обычного их упорядочения (если в программе необходимые для выполнения проверок переменные определяются для нескольких фильтров одновременно, то их программные фрагменты размещаются в программе приема в том же порядке, что и сами фильтры в описании приема).

В большинстве случаев компилятор автоматически определяет тип значения переменной, используемой в фильтре (терм, либо вхождение, либо символ, и т.п.) - в особых случаях предусмотрено явное указание такого такого типа.

Часть встречающихся в фильтрах подвыражений называем, по аналогии с ЛОСом, операторными выражениями. Иногда они действительно оказываются тождественны операторным выражениям ЛОСа. Кроме операторных выражений, в фильтрах могут использоваться также термы предметного уровня. В отличие от операторных выражений, составленных из операций над объектами структурного уровня (термами, списками, задачами и т.п.), термы предметного уровня образованы операциями и отношениями над объектами рассматриваемой предметной области (например, числами, точками, функциями, и т.п.). Термы предметного уровня, встречающиеся в фильтрах, будем называть также термами "в теоремных переменных", так как они аналогичны термам теоремы приема. При описании фильтров уточняется, в каких случаях допустимо использование операторного выражения, а в каких - терма "в теоремных переменных".

10.3.1 Логические связки и квантор существования

В фильтрах можно использовать обычные логические связки "не(A)", "и($A_1 \dots A_n$)", "или($A_1 \dots A_n$)", а также конструкцию "альтернатива(A_1 A_2 A_3)" - в случае истинного A_1 ее истинность совпадает с истинностью A_2, а иначе - с истинностью A_3.

Роль квантора существования при задании фильтров играет конструкция "контекст($A_1 \dots A_n$)". Термы A_i в этой конструкции бывают трех типов:

а) Идентифицирующие термы - они аналогичны операторам ЛОСа, имеющим выходные переменные (в том числе перечисляющим) и определяют значения некоторых новых, до этого не определенных, переменных, используемых в других термах A_j того же списка. Вне данной конструкции "контекст(...)" эти переменные будут считаться не определенными. Изначально определенными при рассмотрении фильтров считаются все переменные, входящие в теорему, а также некоторые переменные, дополнительно определяемые в специальных указателях приема;

б) Проверочные термы - они формулируют условия на значения ранее определенных переменных и представляют собой произвольные правильно построенные фильтры;

в) Указатели идентификации - они уточняют способо применения идентифицирующих термов и идентичны аналогичным указателям для описания всего приема.

Фильтр "контекст($A_1 \dots A_n$)" выражает условие существования значений переменных, определяемых его идентифицирующими термами A_i, при которых истинны все проверочные термы. Порядок размещения термов A_i в списке, в общем, произвольный, за исключением одного ограничения: после первого же идентифицирующего терма с заголовком "вид" либо "подтерм" либо "усм" могут идти только такие идентифицирующие термы, которые имеют один из данных трех заголовков. Допускаются вложенные (с произвольной глубиной вложения) конструкции "контекст(...)".

В качестве простого примера использования фильтра "контекст(...)" рассмотрим следующую ситуацию. Пусть в теореме встречается переменная x, идентифици-

рованная с неизвестной задачи на описание, и требуется ввести фильтр, разрешающий применение приема лишь тогда, когда эта неизвестная встречается в каком-либо неравенстве из условий задачи. Этот фильтр может быть записан как "контекст(условие(A)заголовок(A меньше меньшеилиравно) входит(x A))". Здесь A - какая-либо переменная, не встречающаяся в теореме приема. Идентифицирующий терм "условие(A)" будет перечислять все условия задачи; проверочный терм "заголовок(A меньше меньшеилиравно)" - отбирать из них лишь те, заголовком которых служит отношение "меньше" либо "меньшеилиравно", а проверочный терм "входит(x A)" - проверять вхождение в условие переменной, идентифицированной с x.

Квантор общности в фильтрах не предусмотрен - он выражается через квантор существования и отрицание.

10.3.2 Равенство объектов

Условие "равно(A B)" используется для указания на равенство значений операторных выражений A и B. Если нужно сравнить два терма P, Q "предметного уровня", т.е. заданных через переменные теоремы, то в качестве A, B следует брать операторные выражения "терм(P)", "терм(Q)".

Для сокращенной записи используется также обозначение "равно(A $B_1 \dots B_n$)", означающее, что значение операторного выражения A равно хотя бы одному из значений операторных выражений $B_1, \dots, B_n$.

10.3.3 Сравнение числовых характеристик термов

Если требуется указать, что некоторый терм предметного уровня A имеет значение числовой харктеристики $f(A)$ меньшим, чем любой из термов $B_1, \dots, B_n$, то используется запись "упрощение(x_i и(A $B_1 \dots B_n$) $f(x_i)$)". Здесь x_i - новая переменная, используемая в определяющем характеристику f операторном выражении $f(x_i)$.

Если терм A должен иметь значение числовой характеристики $f(A)$ меньшим, чем хотя бы один из термов $B_1, \dots, B_n$, то используется запись "упрощение(x_i или(A $B_1 \dots$ B_n) $f(x_i)$)".

Например, если требуется применять прием лишь в том случае, когда число неизвестных задачи, встречающихся в терме, идентифицированном с переменной a, меньше числа неизвестных в любом из термов, идентифицированных с переменными b, c, d, то используется фильтр "упрощение(x и(a b c d) число(неизвестная(y) входит(y x))".

10.3.4 Ограничения на задачу

Текущий уровень задачи

Для указания допустимых значений $A_1, \dots, A_n$ текущего уровня сканирования текущей задачи используется фильтр "уровень($A_1 \dots A_n$)". При вводе описания приема, реализуемого в режиме сканирования задачи, обязательно должен быть указан такой фильтр (иначе компилятор сразу будет выдавать отказ).

Цели задачи

Фильтр "цель(A)" означает, что текущая задача имеет цель A. Здесь A - либо логический символ, либо операторное выражение, определяющее цель. Если посредст-

вом идентифицирующих термов либо операторных выражений выделена некоторая другая задача Z, то для указания на то, что она имеет цель A, применяется запись "цель(Z)".

Фильтр "цели($A_1 \ldots A_n$)" означает, что текущая задача не имеет цели, заголовок которой не входил бы в список $A_1, \ldots, A_n$.

Тип задачи

Фильтр "тип(A)" означает, что текущая задача имеет тип A. - логический символ либо операторное выражение. Фильтр "тип(Z A)" означает, что задача Z (введенная при помощи идентифицирующих термов либо операторных выражений) имеет тип A.

Текущая задача - корневая

Для указания на то, что текущая задача - корневая, т.е. выбрана непосредственно из задачника, используется фильтр "исходнаязадача". Он помогает в тех случаях, когда для корневой задачи нужно разрешить применение различного рода усиленных средств.

Неизвестные задачи

Чтобы указать, что текущая задача имеет не более одной неизвестной, используется фильтр "неизвестные(1)". В случае $N > 1$ фильтр "неизвестные(N)" применяется несколько иначе - он указывает, что текущая задача имеет не менее N неизвестных.

Раздел, к которому относится условие задачи на доказательство либо на преобразование

Фильтр "раздел(A)", где A - название раздела, используется, если нужно проверить, что условие текущей задачи на преобразование либо на доказательство содержит хотя бы один логический символ, относящийся к разделу A либо к подразделу раздела A.

Проверка наличия посылок специального вида

Во многих предметных областях бывает полезно вводить в процессе решения задач фиктивные посылки вида "актив(A)", где A - некоторое выражение, которое полезно рассмотреть в задаче, но которое пока не встречается в других ее термах. Такие посылки также упрощают обнаружение в задаче выражений специального вида и часто используются для инициализации применения приемов. Например, в геометрических задачах посылки "актив(расстояние(AB))", "актив(прямая(AB))", "актив(окружность(AB))" и т.п. выделяют для рассмотрения при сканировании различные элементы чертежа. Фильтр "См($A_1 \ldots A_n$)" используется для указания на то, что задача имеет посылки "актив(A_1)", ..., "актив(A_n)".

10.3.5 Ограничения на термы, переменные и логические символы

Простейшие свойства термов и переменных

1. Фильтр "входит(X A)" означает, что переменная либо логический символ, заданные операторным выражением X, входят в терм, заданный операторным выражением A.

2. Фильтр "переменная(X)" означает, что переменная X идентифицирована с переменной.

3. Фильтр "константа(A)" означает, что операторное выражение A задает терм без свободных переменных.

4. Фильтр "вхождениетерма(A B)" означает, что терм B входит в терм A. Здесь A, B - не операторные выражения, а термы предметного уровня, записанные непосредственно в "теоремных" переменных.

5. Фильтр "короче(A B)" означает, что терм, заданный операторным выражением A, короче терма, заданного операторным выражением B. Под длиной терма понимается число вхождений в него логических символов и символов переменных.

6. Фильтр "родобъекта(A B)" означает, что терм, определенный операторным выражением A, имеет своим заголовком операцию, тип значения которой определяется логическим символом B.

7. Фильтр "род(A B)" представляет собой усиленную версию фильтра "родобъекта", использующую при проверке не только заголовок терма A, но и посылки текущего контекста.

8. Фильтр "подобныетермы(A B)" означает, что термы, определенные операторными выражениями A и B, получены друг из друга изменением порядка операндов в коммутативных операциях.

9. Фильтр "связка(A X)" означает, что переменная, определенная операторным выражением X, входит в связывающую приставку терма, определенного операторным выражением A.

Заголовки термов

Фильтр "заголовок(A $B_1 \ldots B_n$)" означает, что терм, определенный операторным выражением A, имеет своим заголовком один из логических символов, определенных операторными выражениями $B_1, \ldots, B_n$.

Фильтры "первыйсимвол(A B)", "второйсимвол(A B)", "предпоследсимвол(A B)", "последнийсимвол(A B)" аналогичны одноименным операторам ЛОСа; здесь операторное выражение A задает некоторый терм, а операторное выражение B - логический символ.

Характеризация терма либо переменной по отношению к задаче

1. Фильтр "неизвестная(x A)" означает, что переменная x идентифицирована с неизвестной задачи A. Обычно A не указывается - тогда по умолчанию рассматривается текущая задача.

2. Фильтр "параметр(x A)" означает, что переменная x идентифицирована с несущественной неизвестной задачи на описание A (при отсутствии A - текущей задачи). Несущественные неизвестные выделяются целью "параметры $x_1 \dots x_n$"; они представляют собой подмножество неизвестных задачи. При решении задачи требуется лишь установить существование удовлетворяющих условиям значений несущественных неизвестных, не находя полного их описания и не включая найденные значения в ответ.

3. Фильтр "известно(T A)" означает, что операторное выражение T идентифицируется с термом, не содержащим неизвестных задачи A (при отсутствии A - текущей задачи).

4. Фильтр "симметрично(T_1 T_2 A)" означает, что термы T_1, T_2 входят симметричным образом в список посылок (при A = "списокпосылок") либо в условия (при A = "списокусловий") текущей задачи, либо в текущий терм задачи или нормализатора (при A = "корень").

5. Фильтр "независит(A)" означает, что терм, заданный операторным выражением A, не содержит переменных, указанных в цели "независит $x_1 \dots x_n$" текущей задачи на описание. Такая цель указывает переменные, которые не должны иметь свободных вхождений в ответ задачи.

6. Фильтр "обозначения(A)" означает, что терм, заданный операторным выражением A, содержит дополнительные переменные, возникшие при выводе в посылках и зарегистрированные в комментарии к посылкам "новаяпеременная ...". Одним из источников появления таких переменных является прием, заменяющий в посылках задачи утверждение о существовании объектов, удовлетворяющих некоторому условию P, на само условие P, в котором данные объекты переобозначены на новые переменные.

Алгебраические выражения

В ГЕНОЛОГе используются не только фильтры общего характера, но и множество специальных фильтров, связанных с различными понятиями конкретных предметных областей. Разумеется, по мере освоения новых разделов приходится пополнять запас типов таких фильтров и, соответственно, расширять компилятор ГЕНОЛОГа. Однако, эта процедура достаточно проста - она сводится к реализации на ЛОСе оператора для нового фильтра и вводу коротенькой программы приема справочника "блокпроверок", обеспечивающего создание обращений к этому оператору для проверки фильтра.

1. Фильтр "линейно(A $X_1 \dots X_n$)" означает, что операторное выражение A определяет алгебраическое выражение, которое после раскрывания скобок становится линейным относительно переменных, определяемых операторными выражениями $X_1, \dots, X_n$.

2. Фильтр "линейноеуравнение($A\ X_1 \ldots X_n$)" означает, что операторное выражение A определяет алгебраическое уравнение, которое после раскрывания скобок становится линейным относительно переменных, определяемых операторными выражениями $X_1, \ldots, X_n$.

10.3.6 Ограничения на вхождения

Символы, расположенные по заданному вхождению

Чтобы указать, что на вхождении, определенном операторным выражением t, расположен один из логических символов списка $A_1, \ldots, A_n$, используется фильтр "символ($t\ A_1 \ldots A_n$)". Этот же фильтр можно использовать, если t определяет вхождение в набор, не являющийся термом, а $A_1, \ldots, A_n$ - операторные выражения, определяющие объекты, один из которых должен находиться на заданном вхождении.

Взаимное расположение двух вхождений

1. Для указания на то, что вхождение, определенное операторным выражением t_1, расположено внутри вхождения, определенного операторным выражением t_2, используется фильтр "подчинено($t_1\ t_2$". Здесь допускается совпадение вхождений.

2. Чтобы указать, что вхождение t_1 расположено не левее вхождения t_2, применяется фильтр "постпозиция($t_1\ t_2$)".

3. Фильтр "алгебрвхождение($t_1\ t_2$)" означает, что вхождение t_2 расположено внутри вхождения t_1 (или совпадает с ним) и достижимо из него только через операции "плюс", "минус", "умножение", "дробь", "модуль", равно", меньше", "меньшеилиравно", а также через основания степеней.

4. Фильтр "обобщмножитель($t_1\ t_2$)" означает, что вхождение t_2 расположено внутри вхождения t_1 (или совпадает с ним) и достижимо из него только через операции "минус", "умножение", "дробь", "модуль", а также через основания степеней.

Свободные переменные

Фильтр "свобоперанд(t)" означает, что операторное выражение t определяет вхождение подтерма, каждая свободная переменная которого свободна во всем терме.

10.3.7 Ограничения на наборы

1. Фильтр "входит(t набор($A_1 \ldots A_n$))" означает, что объект, определяемый операторным выражением t, входит в набор, элементы которого задаются операторными выражениями $A_1, \ldots, A_n$.

2. Фильтр "длинатекста($A\ B$)" означает, что длина набора, определяемого операторным выражением A, равна символьному числу - значению операторного выражения B.

3. Фильтр "длинаменее(A B)" означает, что длина набора, определяемого операторным выражением A, меньше символьного числа - значения операторного выражения B.

10.3.8 Ограничения на новые термы, вводимые приемом

1. Фильтр "длина(A)" означает, что применение приема тождественной либо эквивалентной замены привело к получению более простого в смысле оператора A терма. Оператор A, реализованный на ЛОСе, должен иметь 5 входных переменных: x1 - задача; x2,x3,x4 - вхождение в эту задачу заменяемого терма; x5 - заменяющий терм. Истинное значение оператора означает наличие упрощения. Допускается отсутствие явного указания A, и тогда по умолчанию берется оператор "стандартизация". Возможно использование фильтра "длина(A)" и в случае нормализаторов. Тогда оператор A имеет уже 4 входных переменных: x1, x2 - заменяемый и заменяющий термы; x3 - список посылок нормализатора; x4 - список его комментариев.

2. Фильтр "контрольодз" означает, что все подвыражения заменяющего терма (в приеме замены либо в приеме нормализатора) удовлетворяют условиям на о.д.з.

3. Фильтр "контрольвывода" означает, что новое утверждение, заносимое в список посылок либо условий, не содержит выражений, до этого отсутствовавших в задаче.

10.3.9 Ограничение на способ идентификации

Если прием замены основан на тождестве либо эквивалентности с заменяемой частью $F(t_1 \dots t_n)$, где F - ассоциативно-коммутативный символ, то фильтр "полный" означает, что идентификации с $t_1, \dots, t_n$ подлежат все операнды рассматриваемого в задаче вхождения операции F.

10.3.10 Числовые предикаты

1. Для указания того, что номер логического символа либо переменной A меньше номера логического символа либо переменной B, используется фильтр "менее(A B)".

2. Если операторные выражения A, B задают десятичные числа, то для сравнения этих чисел используется фильтр "меньше(A B)".

3. Фильтры "десчисло(A)", "целое(A)", "натуральное(A)" выражают условия на то, что значением операторного выражения A служит терм - десятичная запись числа (соответственно, целого числа и натурального числа).

10.3.11 Учет комментариев задачи либо пакетного оператора

Для задания комментария в приводимых ниже фильтрах используется набор (A B_1 $\dots$ B_n), где A - логический символ, являющийся заголовком комментария; $B_1, \dots, B_n$ - операторные выражения, определяющие элементы комментария. Для B_i имеются следующие возможности:

а) B_i - логический символ либо терм C в теоремных переменных - тогда данный разряд комментария представлен, соответственно, как логический символ либо как терм. Если логический символ C нужно преобразовать к формату терма, то в качестве B_i берется запись "терм(C)".

б) B_i - служебное слово "теквхожд" (подтерм, идентифицируемый с заменяемым термом) либо "корень" (текущий терм задачи - условие либо посылка) либо "результат" (заменяющий терм приема).

в) B_i - указатель вхождения в теорему "фикс($i_1 \dots i_k$)". Здесь i_1 - номер антецедента (начиная с 1) либо указатель на консеквент (0), в котором находится вхождение; i_2 - номер операнда выделенного антецедента либо консеквента, к которому следует переходить на втором шаге, и т.д. - вплоть до перехода к требуемому вхождению. Соответствующий B_i разряд комментария есть подтерм по найденному вхождению.

г) Если B_i имеет вид "вхождение(C)", то операторное выражение C интерпретируется как задающее вхождение; если B_i имеет вид "логсимвол(C)" либо "переменная(C)", то C интерпретируется как определяющее соответствующий символ (а не терм).

д) Если B_i имеет вид "набор($C_1 \dots C_m$)", то оно определяет набор значений операторных выражений $C_1, \dots, C_m$.

При работе с комментариями используются фильтры следующих типов:

1. Фильтр "комментусловия($A\ B_1 \dots B_n$)"означает, что текущий терм задачи (для приема замены - тот, который содержит заменяемый подтерм) не имеет комментария ($A\ B_1 \dots B_n$). В зависимости от того, где расположен в задаче текущий терм, здесь имеется в виду комментарий посылки либо условия.

2. Фильтр "коммент($A\ B_1 \dots B_n$)" означает, что текущая задача либо (в случае приема пакетного оператора) пакетный оператор не имеет комментария ($A\ B_1 \dots B_n$). В случае задачи на исследование здесь рассматриваются общие комментарии к посылкам.

3. Фильтр "комментпосылки($i\ A\ B_1 \dots B_n$)" означает, что терм задачи, идентифицированный с i-м антецедентом теоремы, не имеет комментария ($A\ B_1 \dots B_n$). Вместо номера антецедента (номер здесь символьный) i может быть операторным выражением, определяющим данный терм задачи, либо логическим символом "теквхожд", указывающим терм задачи, содержащий точку привязки.

4. Фильтр "комментпосылок($A\ B_1 \dots B_n$)" означает, что текущая задача не имеет комментария к посылкам ($A\ B_1 \dots B_n$).

10.3.12 Ограничения на точку привязки

Под точкой привязки приема понимается то текущее вхождение в терм задачи либо пакетного оператора, рассмотрение которого инициализирует применение приема.

1. Фильтры "условие" и "посылка" означают, соответственно, что точка привязки приема располагается в условии либо в посылке текущей задачи.

2. Фильтр "внешзнак($A_1 \dots A_n$)" означает, что заменяемый подтерм расположен только внутри таких термов, заголовки которых принадлежат списку логических символов $A_1, \dots, A_n$.

3. Фильтр "корень" означает, что заменяемый подтерм совпадает со всем рассматриваемым термом (является "корневым" его подтермом).

4. Фильтр "отрицание" означает, что заменяемый подтерм A - корневой либо находится под корневым отрицанием, т.е. весь текущий терм имеет вид A либо "не(A)".

5. Фильтр "отр" означает, что заменяемый подтерм находится под корневым отрицанием, т.е. текущий терм задачи имеет вид "не(A)".

6. Фильтр "неизвтермы" означает, что точка привязки расположена внутри терма задачи на описание либо исследование, контролируемого при выделении повторных вхождений неизвестных. При наличии уравнений с неизвестными этот фильтр отсекает рассмотрение условий, не являющихся уравнениями.

7. Фильтр "внешоператор(i B)" означает, что i - й внешний стэковый кадр (по цепочке кадров глобального стэка - источников его текущего кадра) является кадром оператора либо операторного выражения с заголовком B, либо задачи типа B.

8. Фильтр "блокнеравенства" используется в тех случаях, когда прием выполняет преобразования, направленные на решение неравенства, и нужно проверить целесообразность таких действий в контексте задачи (например, заблокировать решение неравенства при наличии уравнений с теми же неизвестными).

10.3.13 Обращение к проверочному оператору из фильтра

Иногда бывает необходимо перед применением приема убедиться в том, что некоторое утверждение предметного уровня не является (реже - что является) очевидным в контексте задачи. Например, это может понадобиться для предотвращения вывода малоинформативных следствий. В этих случаях используется фильтр "легковидеть (A B)", где A - проверяемое утверждение, сформулированное "в теоремных переменных". B - терм "посылки(C)", определяющий дополнительные посылки C; этот терм может в фильтре отсутствовать. Фильтр используется только для таких A, которые получаются подстановкой в "шаблон" какого-либо проверочного оператора.

10.3.14 Идентифицирующие термы

Идентифицирующие термы используются в фильтрах "контекст(...)" и в некоторых специальных операторных выражениях. Они представляют собой аналог операторов, имеющих выходные переменные, и позволяют доопределять ("идентифицировать") значения новых переменных, используемых в том же фильтре.

Определение вида термов путем их идентификации с заданными термами предметного уровня

Если требуется выполнить идентификацию терма T, определенного операторным выражением A, с заданным термом B предметного уровня, то используется идентифицирующий терм "вид(A B)". Терм B может содержать новые (не встречающиеся

в теореме либо во внешних фильтрах "контекст(...)") переменные, и после идентификации этим переменным оказываются сопоставлены некоторые термы предметного уровня. Такие переменные могут встречаться в фильтрах, относящихся к той же конструкции "контекст(...)", что и "вид(A B)", и там их значения уже будут определены.

Фильтры, использующие результаты идентификации с помощью терма "вид(A B)", могут быть расположены как после данного терма, так и до него - транслятор автоматически переупорядочивает проверки. Требуется лишь, чтобы группа идентифицирующих термов "вид(...)", "подтерм(...)" и "усм(...)" (последние два типа таких термов описываются ниже) завершала список идентифицирующих термов внутри фильтра "контекст(...)". Фильтры после данной группы располагаться могут. Эта группа обрабатывается компилятором отдельно; возникающая для нее программа на ЛОСе выполняет идентификацию всей группы одновременно, учитывая имеющиеся связи между ее элементами.

В некоторых случаях для определения целесообразности применения приема замены бывает необходимо уточнить вид надтерма заменяемого терма. Чтобы идентифицировать этот надтерм с заданным термом A, в котором собственно заменяемый терм обозначен посредством служебного логического символа "теквхожд", применяется идентифицирующий терм "подтерм(A)". Как и в случае идентифицирующего терма "вид(A_1 A_2)", внутри A могут находиться новые переменные, значение которых будет определяться при идентификации.

Если требуется уточнить вид надтерма не заменяемого терма, а терма, расположенного по какому-либо другому вхождению v, то в идентифицирующем терме "подтерм(A)" вместо логического символа "теквхожд" используется терм "теквхожд (B)", где B определяет вхождение v одним из следующих способов:

а) B - переменная x, имевшая в ранее идентифицированной части единственное вхождение;

б) B - указатель вхождения в теорему "фикс(...)" (см. ниже подраздел "Операторные выражения").

Использование идентифицирующих операторов

Для косвенной идентификации различных часто встречаемых отношений между объектами предметной области создаются специальные операторы ЛОСа, называемые идентифицирующими операторами. Например, в геометрии введены: оператор, перечисляющий все точки, для которых в контексте усматривается их принадлежность заданной прямой; оператор, перечисляющий все прямые, для которых усматривается принадлежность им заданной точки; оператор, перечисляющий все прямые, для которых усматривается их перпендикулярность заданной прямой, и т.д. Эти операторы могут использовать простейшие логические переходы, извлекая новые объекты перечисления из имеющихся в контексте утверждений. Несколько подробнее о них будет сказано в разделе, посвященном указателям обработки антецедентов теоремы. Однако, такие операторы могут возникать и при обработке компилятором фильтров приема. Именно, если группа утверждений $A_1, \ldots, A_n$, определяющих отношения, допускающие обработку идентифицирующими операторами, объединяется внутри фильтра "контекст(...)" в идентифицирующий терм вида "усм ($A_1 \ldots$ A_n)", то компилятор обеспечит идентификацию новых переменных, встречающихся в $A_1, \ldots, A_n$, с помощью идентифицирующих операторов.

Идентификация с использованием операции подстановки

Для работы с операцией подстановки служит идентифицирующий терм "результ-подст(A B C D)". Здесь возможны два случая:

а) D - новая переменная, и тогда ей присваивается результат подстановки в терм, определяемый операторным выражением C, набора термов, определяемого операторным выражением B, вместо набора переменных, определяемого операторным выражением A.

б) B - новая переменная, и тогда ей присваивается такой набор термов, подстановка которого вместо набора переменных, определяемого операторным выражением A, в терм, определяемый операторным выражением C, дает терм, определяемый операторным выражением D.

Операнды и подтермы

1. Идентифицирующий терм "операнд(A B)", как и одноименный оператор ЛОСа, может использоваться в двух вариантах: либо A есть операторное выражение, имеющее своим значением вхождение в терм, и тогда новая переменная B идентифицируется как операнд этого вхождения, либо B - операторное выражение, имеющее своим значением вхождение, и тогда новая переменная A идентифицируется как вхождение, непосредственным операндом которого является вхождение B.

2. Идентифицирующий терм "позиция(A B)" позволяет идентифицировать новую переменную A с вхождением в терм либо набор, определяемый операторным выражением B.

3. Идентифицирующий терм "подчинено(A B)" аналогичен одноименному оператору ЛОСа. Если A - операторное выражение, значением которого служит вхождение v в терм, то новая переменная B идентифицируется с вхождением строгого надтерма v; если B - операторное выражение, значением которого служит вхождение v в терм, то новая переменная A идентифицируется с вхождением нестрогого подтерма v. В случае необходимости идентифицировать B с вхождением нестрогого надтерма v вместо терма "подчинено(A B)" используется "Подчинено(A B)".

4. Идентифицирующий терм "вхождениетерма(A B C)" используется для идентификации переменной C со вхождением терма, заданного операторным выражением B, в терм, заданный операторным выражением A.

5. При работе с кванторной импликацией применяются идентифицирующие термы "антецедент(A B)" и "консеквент(A B)". В них операторное выражение A задает некоторую кванторную импликацию, а переменная B идентифицируется, соответственно, с антецедентом либо консеквентом этой импликации.

Заголовки операндов

Идентифицирующие термы "символ(A B)", "первыйсимвол(A B)", "второйсимвол(A B)", "предпоследсимвол(A B)", "последнийсимвол(A B)" используются для идентификации новой переменной B с символом, расположенным по вхождению, заданному операторным выражением A либо по непосредственному операнду этого вхождения (соответственно, первому, второму, предпоследнему и последнему).

Условия на вхождения и термы, связанные с предметной областью

Такие идентифицирующие термы время от времени придется добавлять к данному списку, по мере того, как этого будут требовать новые предметные области. При этом понадобится расширять и компилятор ГЕНОЛОГа. Однако, это расширение весьма несложно - оно сводится к написанию небольших программ справочника "значение", обеспечивающего компиляцию идентифицирующих термов.

1. Идентифицирующий терм "алгебрвхождение(A B)" позволяет идентифицировать переменную B с вхождением, расположенным внутри вхождения, заданного операторным выражением A, и достижимым из него только через операции "умножение", "плюс", "минус", "модуль", "дробь", "равно", "меньше", "меньшеилиравно", а также через основания степеней. Возможна обратная идентификация: операторное выражение B задает некоторое вхождение v, а новая переменная A идентифицируется с вхождением, внтури которого расположено v и достижимым из v только через операции того же списка.

2. Идентифицирующий терм "обобщмножитель(A B)" позволяет идентифицировать переменную B с вхождением, расположенным внутри вхождения, заданного операторным выражением A, и достижимым из него только через операции "минус", "умножение", "дробь", "модуль", а также через основания степеней.

3. Идентифицирующий терм "обобщслагаемое(A B)" позволяет идентифицировать переменную B с вхождением, расположенным внутри вхождения, заданного операторным выражением A, и достижимым из него только через операции "умножение", "плюс", "минус", причем в этом смысле тупиковым.

4. Идентифицирующий терм "тригаргумент(A B)" позволяет идентифицировать переменную B с вхождением аргумента тригонометрической операции в терм, определяемый операторным выражением A.

5. Идентифицирующий терм "кратныйаргумент(A B C)" позволяет идентифицировать переменную C с целым числом (термом), при домножении на который тригонометрический аргумент - значение операторного выражения B - обращается в тригонометрический аргумент - значение операторного выражения B.

6. Идентифицирующий терм "частное(A B C)" позволяет идентифицировать новую переменную C с частным целых чисел, являющихся значениями операторных выражений A и B.

7. Идентифицирующий терм "видмногочлена(A B C)" проверяет, что терм, вхождение которого определяется операторным выражением A, при раскрывании скобок преобразуется к многочлену от переменной - значения операторного выражения B, и идентифицирует новую переменную C со степенью этого многочлена (десятичным числом).

Элементы задачи либо обращения к пакетному оператору

1. Идентифицирующий терм "посылка(A B)" идентифицирует переменную A с посылкой задачи B. Указание на задачу B может отсутствовать. Если прием с данным фильтром применяется при сканировании задачи, то в этом случае

берутся посылки текущей задачи; если же прием относится к пакетному оператору, то берутся посылки, указанные в обращении к оператору.

2. Идентифицирующий терм "условие(A B)" идентифицирует переменную A с условием задачи B. Указание на задачу B может отсутствовать, и тогда берется текущая задача.

3. Если требуется идентифицировать A с еще не идентифицированной в приеме посылкой (условием) текущей задачи, то используется терм "новаяпосылка(A)" (соответственно, "новоеусловие(A)").

4. Идентифицирующий терм "неизвестная(A B)" идентифицирует переменную A с неизвестной задачи B. Если указание на задачу B отсутствует, то берется текущая задача. Если B - логический символ "вход", то текущая задача приема есть задача на исследование, и тогда берется ее внешняя задача на описание. Данный терм может применяться также в приемах нормализаторов и анализаторов. Тогда B отсутствует; в случае нормализатора для определения списка неизвестных используется комментарий (неизвестные ...) из списка комментариев обращения к нормализатору. Компилятор обеспечивает создание таких комментариев при обращениях к нормализаторам, формат которых имеет элемент "неизвестные".

5. Идентифицирующий терм "цель(A $B_1 \dots B_n$)" идентифицирует набор ($B_1 \dots B_n$) с целью ($C_1 \dots C_n$) задачи A. При отсутствии A берется текущая задача. Некоторые элементы B_i набора ($B_1 \dots B_n$) суть операторные выражения, определяющие соответствующие разряды C_i; другие суть новые переменные, идентифицируемые с C_i. Требуется наличие хотя бы одной новой переменной (при этом возможен случай $n = 1$ - тогда B_1 просто идентифицируется с произвольной целью задачи A). По умолчанию новые переменные идентифицируются с термами (предполагается, что на соответствующем разряде набора - цели также находится терм). Если требуется уточнить тип объекта, с которым должна быть идентифицирована переменная x, то соответствующее B_i может иметь вид "переменная(x)" (здесь x идентифицируется с переменной) либо вид "терм(x)" (здесь при идентификации x преобразуется к виду терма, даже если в соответствующем разряде цели находился не терм, а символ).

6. Для идентификации переменной A с надзадачей текущей задачи, имеющей тип B, используется идентифицирующий терм "надзадача(A B)". Если B есть логический символ 0, то A идентифицируется с произвольной надзадачей текущей задачи.

7. Идентифицирующий терм "внешусловие(A)" идентифицирует переменную A с условием внешней задачи на описание, блоком анализа которой служит текущая задача на исследование.

8. Идентифицирующий терм "обл(A)" идентифицирует переменную A с утверждением из области текущего вхождения в задачу.

Комментарии задач и пакетных операторов

1. Идентифицирующий терм "комментариипосылки(i $A_1 \dots A_n$)" идентифицирует набор ($A_1 \dots A_n$) с комментарием к терму задачи, идентифицированному с i

- м антецедентом (i - символьный номер) теоремы, либо, при $i = 0$, к текущему терму задачи. Некоторые из A_j - новые переменные, которым присваиваются соответствующие разряды комментария, остальные - операторные выражения, значения которых должны совпадать с соответствующими позициями комментария. Если новая переменная x должна быть идентифицирована с переменной, то A_j берется вида "переменная(x)"; иначе, по умолчанию, новые переменные идентифицируются как термы.

2. Идентифицирующий терм "комментарий($A_1 \ldots A_n$)" идентифицирует набор ($A_1 \ldots A_n$) с комментарием к текущей задаче (если тип ее - "исследовать", то с комментарием к посылкам задачи; для приема пакетного оператора здесь берется комментарий из обращения к этому оператору). Некоторые из A_j - новые переменные, которым присваиваются соответствующие разряды комментария, остальные - операторные выражения, значения которых должны совпадать с соответствующими позициями комментария. Если новая переменная x должна быть идентифицирована с переменной, то A_j берется вида "переменная(x)"; иначе, по умолчанию, новые переменные идентифицируются как термы.

3. Идентифицирующий терм "примечание($A_1 \ldots A_n$)" идентифицирует набор ($A_1 \ldots A_n$) с комментарием к списку посылок текущей задачи. Возможно использование данного терма в приеме пакетного оператора (комментарий и в этом случае берется к списку посылок текущей задачи). Некоторые из A_j - новые переменные, которым присваиваются соответствующие разряды комментария, остальные - операторные выражения, значения которых должны совпадать с соответствующими позициями отбираемого комментария. Если новая переменная x должна быть идентифицирована с переменной, то A_j берется вида "переменная(x)"; иначе, по умолчанию, новые переменные идентифицируются как термы.

Контекст точки привязки

1. Идентифицирующий терм "внешоперанд(A)" идентифицирует новую переменную A с еще не вовлеченным в идентификацию операндом той ассоциативно - коммутативной операции, с подмножеством операндов которой идентифицирован преобразуемый терм в приеме замены.

2. Идентифицирующий терм "внешвхождение(A)" идентифицирует новую переменную A при помощи оператора "внешвхождение" с вхождением, встречающимся вне текущего вхождения в обрабатываемый нормализатором терм либо в один из термов списка B комментария "внешвхождение B" к этому нормализатору. Такие идентифицирующие термы применяются в нормализаторах, обеспечивающих выявление повторных вхождений подтермов специального вида (например, подтермов с неизвестными). Эти повторные вхождения прослеживаются во всей группе термов, получаемой добавлением обрабатываемого нормализатором терма к списку B указанного выше комментария.

3. Идентифицирующий терм "другоевхождение(A)" идентифицирует новую переменную A при помощи оператора "другоевхождение" с отличным от текущего вхождением в обрабатываемый нормализатором терм либо в один из термов списка B комментария "внешвхождение B". Применяется в нормализаторах,

обеспечивающих выявление повторных вхождений подтермов специального вида.

4. Идентифицирующий терм "внешконтекст(A)" идентифицирует новую переменную A с вхождением B_2 из комментария "внешконтекст B_1 B_2" к текущему нормализатору. Последний комментарий вводится в тех случаях, когда применение нормализатора должно учитывать внешний контекст преобразуемого терма. У него B_2 - вхождение преобразуемого нормализатором терма в некоторый "базовый" терм - условие или посылку задачи, обратившейся к нормализатору, либо терм, преобразуемый внешним нормализатором.

5. Идентифицирующий терм "внешкорень(A)" идентифицирует новую переменную A с термом, вхождение B_2 в который определяется комментарием "внешконтекст B_1 B_2" к текущему нормализатору.

Наборы

1. Идентифицирующий терм "входит(A B)" идентифицирует новую переменную A с элементом набора, определяемого операторным выражением B.

2. Идентифицирующий терм "символ(A B)" идентифицирует новую переменную B с объектом, расположенным по вхождению, определяемому операторным выражением A.

3. Идентифицирующий терм "заголовок(A B)" идентифицирует новую переменную B с логическим символом либо переменной - первым элементом терма либо набора, определяемого операторным выражением A.

4. Идентифицирующий терм "постпозиция(A B)" идентифицирует новую переменную A с вхождением в набор либо терм, расположенным не раньше вхождения в этот набор или терм, являющегося значением операторного выражения B.

5. Идентифицирующий терм "новпозиция(A B)" идентифицирует новую переменную A с вхождением в набор либо терм, расположенным строго после вхождения в этот набор или терм, являющегося значением операторного выражения B.

6. Идентифицирующий терм "ключ(A B C)" идентифицирует новую переменную C с элементом набора, являющегося значением операторного выражения A, и представляющим собой либо логический символ, определяемый операторным выражением B, либо набор, начинающийся с такого логического символа.

7. Идентифицирующий терм "биключ(A B C D)" идентифицирует новую переменную D с элементом набора, являющегося значением операторного выражения A, и представляющим собой набор, первый элемент которого есть логический символ - значение операторного выражения B, а второй элемент определяется значением операторного выражения C.

8. Идентифицирующий терм "разряд(A B C)" идентифицирует новую переменную C с вхождением в набор либо терм, определяемый операторным выражением A логического символа либо переменной, определяемых операторным выражением B.

9. Идентифицирующий терм "серия(A_1 B_1 ... A_n B_n)" идентифицирует новые переменные $A_1, \ldots, A_n$ с элементами наборов - значений операторных выражений $B_1, \ldots, B_n$ - имеющими один и тот же номер $i; i = 1, 2, \ldots$.

10. Идентифицирующий терм "контрсерия(A_1 $B_1 \ldots A_n$ B_n)" идентифицирует новые переменные $A_1, \ldots, A_n$ с вхождениями в наборы, расположенными левее вхождений в эти наборы - значений операторных выражений $B_1, \ldots, B_n$ - на одно и то же число позиций $i; i = 0, 1, \ldots$.

Логические конструкции, используемые в идентифицирующих термах

1. Идентифицирующий терм "список(A $B_1 \ldots B_n$)" идентифицирует значение новой переменной A со значением одного из операторных выражений $B_1, \ldots, B_n$.

2. Идентифицирующий терм "равно(A B)" идентифицирует значение новой переменной A со значением операторного выражения B. Если A имеет вид "значение(F X)", то идентифицируется терм, определяющий значения функции F.

3. Идентифицирующий терм "новаяпеременная(A B)" идентифицирует новую переменную B с переменной, не входящей в список переменных, определяемый операторным выражением A.

4. Идентифицирующий терм "перестановка(A B C D)" идентифицирует новые переменные C и D с парой операторных выражений A и B - в прямом либо обратном порядке.

5. Идентифицирующий терм "или($A_1 \ldots A_n$)" позволяет осуществлять идентификацию согласно одному из дизъюнктивных членов $A_i; i = 1, \ldots, n$. Каждый такой член имеет вид B_1 & ... & B_m, где $B_1, \ldots, B_m$ - набор идентифицирующих термов и фильтров, а также указателей идентификации - как для фильтра "контекст(...)". Идентификации подлежат общие для всех $A_1, \ldots, A_n$ новые переменные.

6. Идентифицирующий терм "значения(A B)" позволяет провести идентификацию для утверждения A предметного уровня посредством пакетного синтезатора, предусмотренного для утверждений такого вида. B - терм "посылки(C)", определяющий дополнительные посылки C обращения к синтезатору. Утверждение C - конъюнкция этих посылок - также относится к предметному уровню. Если дополнительные посылки не требуются, то терм B отсутствует.

10.3.15 Операторные выражения

Операторные выражения используются для задания в фильтрах приемов различных объектов "структурного" уровня (термов, наборов, символов и т.п.). Их следует отличать от выражений, задающих объекты "предметного" уровня (числа, точки пространства, функции и т.п.). Последние выражения также могут использоваться в фильтрах, однако это происходит значительно реже и лишь в специально оговоренных случаях.

Простейшие выражения

1. Любой логический символ представляет собой операторное выражение; значением его (за исключением приводимых ниже служебных слов) служит он сам.

2. Любая теоремная переменная, либо вспомогательная переменная, определенная до этого некоторым идентифицирующим термом, представляет собой операторное выражение. Тип значения этого выражения (терм, логический символ, вхождение, и т.п.) определяется компилятором ГЕНОЛОГа автоматически - в зависимости от контекста. Исключительные случаи, когда такое автоматическое доопределение типа необходимо уточнять вручную, указываются при описании различных контекстов использования операторных выражений.

3. Логический символ "теквхожд" - служебный символ, обозначающий заменяемый терм для приемов замены (в зависимости от контекста, этот символ может также обозначать вхождение заменяемого терма; далее аналогичные оговорки опускаем).

4. Логический символ "корень" служебный символ, обозначающий текущий терм задачи; этот терм содержит точку привязки приема.

5. Логический символ "вход" - служебный символ, используемый в фильтрах, сопровождающих обращение к нормализатору (см. ниже раздел, посвященный обращениям к нормализаторам), и обозначающий подлежащий нормализации терм.

6. Логический символ "текущийуровень" - служебный символ, обозначающий текущий уровень текущей задачи.

7. Логический символ "исходнаязадача" - служебный символ, обозначающий исходную задачу, т.е. фиктивную задачу на исследование, при сканировании которой запускается процедура интерфейса.

8. Логический символ "посылки" - служебный символ, обозначающий список утверждений, образующих контекст реализуемого приема.

9. Логический символ "текущаязадача" - служебный символ, обозначающий текущую задачу.

10. Логический символ "внешописать" - служебный символ, обозначающий внешнюю задачу на описание для текущей задачи на исследование.

11. Логический символ "результат" - служебный символ, обозначающий заменяющий терм приема замены. Применяется как в приемах сканирования задачи, так и в приемах нормализаторов.

Численные выражения

1. Операторное выражение "количествооперандов(A)" обозначает число (десятичное) корневых операндов терма, определяемого операторным выражением A. Служебный символ "количествооперандов" используется для обозначения числа операндов заменяемого подтерма.

2. Служебный символ "числонеизвестных" обозначает число неизвестных текущей задачи либо текущего пакетного оператора. Операторное выражение "числонеизвестных(A)" обозначает число (десятичное) неизвестных в терме, определяемом операторным выражением A.

3. Для нахождения числа наборов объектов, удовлетворяющих заданному условию, используется логическая конструкция "число(A)", аналогичная фильтру "контекст(A)". Здесь A - список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на объекты - значения новых переменных, определяемых идентифицирующими термами из A. Значение операторного выражения представлено в формате десятичного числа.

4. Если нужно не просто определить число наборов объектов, удовлтеворяющих заданному условию, но просуммировать по всем таким наборам значения заданного операторного выражения, то используется логическая конструкция "суммавсех(A t)". Здесь, как и в предыдущем случае, A есть список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на объекты - значения новых переменных. t есть операторное выражение, определяющее суммируемые величины (десятичные числа).

5. Возможно нахождение наибольшего общего делителя всех целых чисел, определяемых заданным операторным выражением для всевозможных наборов объектов, удовлетворяющих заданному условию. Здесь используется конструкция "нод(A t)"; A - список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на объекты - значения новых переменных; t - операторное выражение, определяющее указанные целые числа.

   Аналогичная конструкция "нок(A t)" используется для нахождения наименьшего общего кратного.

   Если нужно определить наибольший общий делитель либо наименьшее общее кратное двух чисел, определяемых операторными выражениями B и C, то используются, соответственно, операторные выражения "нод(B C)" и "нок(B C)". Чтобы не было путаницы с указанными выше конструкциями "нод(A t)" и "нок(A t)", список A должен начинаться с идентифицирующего терма либо с фильтра.

6. Для сложения, вычитания и умножения чисел, определенных операторными выражениями A и B, используются операторные выражения "плюс(A B)", "вычитание(A B)" и "умножение(A B)". Здесь речь идет о десятичных числах. Для выполнения этих же операций над числами в символьной записи используются операторные выражения "плюссимв(A B)", "вычитсимв(A B)" и "умножсимв(A B)". Заметим, что все перечисленные операции - строго двуместные; если число операндов более двух, то нужно использовать выражение многоместной операции через двуместные.

   Чтобы изменить знак десятичного числа, определенного операторным выражением A, используется операторное выражение "минус(A)"; аналог для символьного формата чисел здесь отсутствует.

7. Номер (в формате десятичного числа) логического символа, определенного операторным выражением A, задается операторным выражением "номерсимв(A)". Чтобы определить номер переменной, задаваемой операторным выражением A,

используется выражение "номерпеременной(A)". Этот номер представлен уже в символьном формате.

8. Операторное выражение "числзначение(A)" имеет своим значением число в десятичном формате, определяемое константным числовым термом - значением операторного выражения A.

9. Операторное выражение "длинанабора(A)" имеет своим значением константный числовой терм, задающий длину набора, определяемого операторным выражением A. Чтобы определить десятичное число разрядов набора либо терма, определяемого операторным выражением A, используется операторное выражение "Длина(A)".

10. Операторное выражение "числочленов(A B)" имеет своим значением число корневых операндов терма, определенного выражением A, если заголовок этого терма - логический символ B. Иначе это выражение имеет своим значением число 1.

11. Для нахождения числа делителей целого числа - значения операторного выражения A - используется операторное выражение "числоделителей(A)".

Переменные и логические символы

1. Для обозначения переменной, номер которой (символьное число) определяется операторным выражением A, используется операторное выражение "икс(A)". Для обозначения логического символа, номер которого (в формате десятичного числа) определяется операторным выражением A, используется операторное выражение "симвномер(A)".

2. Если требуется ввести набор переменных, не входящих в список переменных, определяемый операторным выражением A, и имеющий такую же длину, как набор - значение операторного выражения B, то используется операторное выражение "кортежпеременных(A B)".

3. Связывающая приставка терма, определяемого операторным выражением A, обозначается как "связприставка(A)".

4. Если операторное выражение A определяет терм либо набор термов, то выражение "параметры(A)" определяет список всех свободных переменных терма A (термов набора A). Выражение "списокпеременных(A)" определяет в этой же ситуации список всех вообще переменных, входящих в A (либо в термы набора A).

5. Если операторное выражение A обозначает терм, то выражение "Неизвестные(A)" обозначает список всех неизвестных, входящих в этот терм.

Наборы

1. Набор, образованный объектами - значениями операторных выражений $A_1, \ldots,$ A_n - обозначается "набор($A_1 \ldots A_n$)".

2. Для создания набора, перечисляющего без повторений объекты, удовлетворяющие заданному условию, используется операторное выражение "перечисление(A B)". Здесь A - список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на новые переменные (аналогично фильтру "контекст(A)"); B - операторное выражение с этими переменными, определяющее перечисляемые объекты. Если тип значения выражения B допускает различные варианты, то по умолчанию это значение определяется как логический символ либо символ переменной; чтобы явно указать, что рассматриваемый объект должен представлять собой терм, B берется вида "терм(C)", где C собственно и определяет данный терм.

3. Для создания набора, перечисляющего с возможными повторениями объекты, удовлетворяющие заданному условию, используется операторное выражение "выписка(A B)" - аналогично выражению "перечисление(A B)".

4. Если требуется сформировать набор заданной длины, на каждой позиции которого находится один и тот же объект, то используется операторное выражение "бланк(A B C)". Здесь операторное выражение C пределяет некоторое символьное число, причем длина формируемого набора на C больше длины набора A. Операторное выражение B указывает объект, помещаемый на каждой позиции формируемого набора.

5. Конкатенация наборов либо символов, определяемых операторными выражениями $A_1, \ldots, A_n$, обозначается "конкатенация($A_1 \ldots A_n$)".

6. Операторные выражения "вычеркивание(A B)", "пересечениесписков(A B)", "объединениесписков(A B)", как и в ЛОСе, определяют, соответственно, результат исключения из набора, определяемого операторным выражением A, элементов набора, определяемого операторным выражением B (как и в ЛОСе, при исключении учитывается кратность вхождений - из A исключается лишь столько вхождений заданного элемента, сколько их есть в B); пересечение этих наборов (в A сохраняются только те элементы, которые входят в B) и их объединение (к концу набора A присоединяются все не входившие в него элементы набора B).

7. Результат присоединения к началу набора, определяемого операторным выражением A, объекта, определяемого операторным выражением B, обозначается "префикс(B A)"; результат присоединения B к концу A - "суффикс(A B)". Результат вставки в набор A перед позицией, определяемой выражением B, значения выражения C, обозначается "вставка(A B C)".

8. Результат отбрасывания первого элемента набора, определяемого операторным выражением A, обозначается "окончание(A)". Результат исключения в наборе A разряда либо группы разрядов, определяемых выражением B, обозначается "исключение(A B)".

9. Копия набора, определяемого операторным выражением A, обозначается "копия(A)".

10. Первый элемент терма либо набора, определяемого операторным выражением A, обозначается "начало(A)"; последний элемент набора A обозначается

"конец(A)". Объект, расположенный на $i+1$ - м месте, считая от начала набора A, обозначается "левпозиция(A I)", где i - сивмольное число, являющееся значением выражения I. Объект, расположенный на $i+1$ - м месте, считая с конца набора A, обозначается "правпозиция(A I)".

11. Объект, расположенный в наборе, определяемом выражением B на позиции, соответствующей позиции объекта - значения выражения C в наборе, определяемом выражением A, обозначается "таблзначение(A B C)".

12. Набор объектов X, таких, что в наборе a имеется пара (b X), обозначается "Ключ(A B)"; здесь A, B - операторные выражения, имеющие значения a, b.

Термы

1. Чтобы задать терм T, полученный из некоторого терма A предметного уровня подстановкой в него вместо "теоремных" переменных идентифицированных с ними подтермов, используется операторное выражение "терм(A)". Если в A имеются подтермы, к которым согласно описанию приема должный применяться цепочки нормализаторов либо вспомогательных задач, то при построении T эти подтермы подвергаются данной обработке.

2. Иногда терм, который нужен в качестве значения операторного выражения, уже имеется в теореме. Чтобы не переписывать этот терм, используя приведенную выше конструкцию "терм(...)", можно сослаться на его вхождение в теорему при помощи так называемого указателя вхождения в теорему - терма "фикс(n_0 $n_1 \ldots n_k$)". Если теорема приема имеет вид "длялюбого($x_1 \ldots x_p$ если $A_1 \ldots A_m$ то A_0)", то n_0 есть номер i того утверждения A_i, в котором расположено рассматриваемое вхождение. При этом ($n_1 \ldots n_k$) - последовательность номеров операндов в A_i, определяющая путь от корня A_i к данному вхождению (операнды нумеруются начиная с 1, слева направо; сначала выполняется переход к n_1 - му операнду корневой операции A_i, затем к n_2 - му операнду данного операнда, и т.д.). Если заголовок теоремы приема отличен от квантора общности, то указатель вхождения перечисляет номера операндов прямо с корневой операции теоремы. Указатель вхождения в теорему терма A заменяет операторное выражение "терм(A)" и сам может рассматриваться как операторное выражение. Оно является даже более сильным, чем "терм(A)", так как в тех фильтрах, где нужно вхождение терма A, а не сам этот терм, в качестве значения выражения и берется такое вхождение (то, с которым было идентифицировано соответствующее вхождение A в теорему). Заметим, что набирать вручную последовательность номеров $n_0, \ldots n_k$ из указателя вхождения не нужно - в интерфейсе редактора приемов предусмотрена возможность автоматического ввода этой последовательности после выделения клавишами курсора подтерма теоремы.

3. Терм, сформированный из операндов, перечисляемых в наборе, являющемся значением операторного выражения B, при помощи заголовка - значения выражения A, обозначается "сборка(A B)". Если в случае одноэлементного набора B заголовок A не навешивается, а результатом служит сам элемент набора B, то используется операторное выражение "унисборка(A B)".

4. Если операторные выражения $A_1, \ldots, A_n$ определяют некоторые термы $a_1, \ldots, a_n$, а операторное выражение B - логический символ b, то операторное выражение "запись(B $A_1 \ldots A_n$)" определяет терм $b(a_1 \ldots a_n)$.

5. Если операторное выражение A определяет некоторый терм $f(t_1 \ldots t_n)$ либо вхождение первого символа такого терма, то операторное выражение "первыйтерм(A)" определяет терм t_1, операторное выражение "второйтерм(A)" - терм t_2, операторное выражение "предпоследтерм(A)" - терм t_{n-1} и операторное выражение "последнийтерм(A)" - терм t_n. Наконец, операторное выражение "набороперандов(A)" определяет набор термов $(t_1, \ldots, t_n)$.

6. Если операторное выражение A определяет вхождение первого символа некоторого подтерма, то сам этот подтерм задается выражением "подтерм(A)". Чтобы определить тот терм, вхождение в который задает выражение A, используется выражение "базавхождения(A)".

7. Если операторное выражение A определяет вхождение первого символа некоторого терма, а операторное выражение B - вхождение корневого операнда этого терма, то выражение "исключениеоперанда(A B)" определяет результат исключения в терме A его операнда B (если остается единственный операнд корневой операции, то корневая операция отбрасывается).

8. Если операторное выражение A определяет терм; операторное выражение B - вхождение либо набор вхождений в A (упорядоченных слева направо), а операторное выражение C - соответственно, терм либо набор термов (той же длины, что и B), то выражение "заменатермов(A B C)" обозначает результат замены в A вхождений B на термы C.

Вхождения

1. Если операторное выражение A обозначает вхождение в терм либо в набор, то выражение "буква(A)" обозначает объект, расположенный по этому вхождению.

2. Если операторное выражение A определяет набор, то выражение "левыйкрай(A)" обозначает вхождение первого элемента набора A, а "правыйкрай(A)" - вхождение последнего элемента набора A. Если, далее, операторное выражение B определяет символьный номер i, то выражение "окрестность(A B)" обозначает вхождение $i+1$ - го элемента набора A.

3. Если операторное выражение A определяет терм $f(a_1 \ldots a_n)$ либо вхождение этого терма, то выражения "первыйоперанд(A)", "второйоперанд(A)", "предпоследоперанд(A)", "последнийоперанд(A)" обозначают вхождения операндов a_1, a_2, a_{n-1}, a_n соответственно.

4. Если операторное выражение A определяет вхождение в набор, то выражение "левсосед(A)" определяет вхождение предыдущего элемента набора (если A - вхождение первого элемента, то берется то же самое вхождение A). Выражение "правсосед(A)" обозначает вхождение следующего элемента набора (если A - вхождение последнего элемента, то берется оно само).

5. Если операторное выражение A определяет вхождение в набор, являющийся значением операторного выражения B, причем операторное выражение C также определяет набор, то выражение "соотвпозиция(A B C)" обозначает вхождение в набор C разряда, имеющего тот же номер, что и разряд A набора B.

6. При идентификации антецедента вида $P(A\ B)$, где P - симметричный и транзитивный предикатный символ, может использоваться указатель "равно(i)" (i - символьный номер данного антецедента). Такой антецедент может быть усмотрен либо из явно присутствующего в контексте утверждения $P(A\ B)$, либо из двух утверждений $P(A\ C)$, $P(B\ C)$. В последнем случае C - "стандартное обозначение" выражений A, B, а также, быть может, ряда других встречающихся в контексте выражений (в первом случае роль "стандартного обозначения" играет выражение B). Появление группы утверждений $P(\ldots)$, связывающих $A, B, \ldots$ с их "стандартным обозначением", должно быть заблаговременно обеспечено специальной стандартизацией. Чтобы в описании приема можно было анализировать не только выражения A и B, но и выражение C, используется операторное выражение "выражение(i)", значением которого служит вхождение первого символа подтерма C (фактически это вхождение второго операнда в использованное при идентификации утверждение $P(A\ C)$).

7. Если i - й антецедент теоремы был идентифицирован с посылкой либо условием задачи, то операторное выражение "вхождениепосылки(i)" определяет вхождение этой посылки либо условия, соответственно, в список посылок либо условий. Здесь i - символьный номер.

Задачи

1. Если операторное выражение A обозначает задачу, то операторные выражения "цели(A)", "списокусловий(A)", "списокпосылок(A)" обозначают, соответственно, список целей, список условий (используется только в ситуации, когда заблаговременно установлено, что A - задача на описание) и список посылок задачи A. Возможно отбрасывание явного указания на задачу A - тогда рассматриваемое операторное выражение становится логическим символом. В этой ситуации по умолчанию берется текущая задача.

2. Если операторное выражение A обозначает задачу, то операторные выражения "неизвестные(A), "переменные(A)" обозначают, соответственно, список всех неизвестных задачи A (в него не включается заголовок "неизвестные" цели, перечисляющей неизвестные) и список всех переменных, встречающихся в задаче A. Возможно опускание явной ссылки на задачу, и тогда берется текущая задача.

3. Если операторное выражение A обозначает задачу, то операторное выражение "комментарии(A)" обозначает список комментариев к задаче A (которая не должна иметь типа "исследовать"). Возможно опускание явной ссылки на задачу A, и тогда берется текущая задача. Данное операторное выражение (без ссылки на A) применяется также для приемов пакетных операторов - тогда оно имеет своим значением список комментариев обращения к данному оператору. Операторное выражение "комментариипосылок(A)" определяет аналогичным образом список комментариев к посылкам задачи A.

4. Чтобы обозначить список комментариев к заданному условию задачи, определяемой операторным выражением A, используется выражение "комментариийусловия(A B)". Здесь B - либо символьный номер антецедента теоремы, идентифицированного с рассматриваемым условием, либо логический символ "корень" (указатель на условие, в котором находится текущее вхождение сканирования задачи), либо операторное выражение, имеющее своим значением рассматриваемое условие. Возможно опускание явной ссылки на задачу A, и тогда берется текущая задача.

5. Операторное выражение "комментариипосылки(A B)" совершенно аналогично выражению "комментариийусловия(A B)" - оно обозначает список комментариев к посылке B задачи A.

6. Если операторное выражение A обозначает задачу, то выражение "максимальныйуровень(A)" обозначает максимальный уровень этой задачи. Возможно отсутствие явного указания на задачу A, и тогда берется текущая задача.

7. Если операторное выражение A определяет задачу на доказательство либо на преобразование, то выражение "условие(A)" обозначает условие этой задачи. Возможно отсутствие явного указания на задачу A, и тогда берется текущая задача.

Обращения к справочникам

Результат обращения к справочнику определяется операторным выражением "справка(A B $C_1 \ldots C_n$)". Здесь A - тип справочника (логический символ); B - логический символ, к программе которого происходит обращение (может задаваться произвольным операторным выражением); $C_1, \ldots, C_n$ - операторные выражения, определяющие сопутствующие данные обращения.

Определение типа объекта

Операторное выражение "типданных(A)" определяет тип объекта, заданного операторным выражением A: 0 - логический символ либо переменная, 1 - терм, 2 - вхождение, 3 - ошибка (ссылка на пустую зону), 4 - набор, не являющийся термом.

Условное выражение

Операторное выражение "вариант(A B C)" определяет при истинном фильтре A значение, задаваемое операторным выражением B, а при ложном - операторным выражением C.

10.3.16 Очередность проверки фильтров и дополнительные действия при проверке

Организация слежения за ситуацией, в которой ложный на текущий момент фильтр может оказаться истинным

Если условие фильтра A не выполнено, но оно может оказаться выполненным при возникновении в списке посылок задачи одного из утверждений списка B, то при появлении таких посылок целесообразна повторная попытка применения приема. Ее

можно организовать, например, понизив вес текущего терма (при сканировании которого была инициирована попытка применения приема) до заданной величины P. Указанные действия - организация слежения за появлением при решении задачи посылок списка B и соответствующее переключение внимания - будут выполняться, если вместо фильтра A поместить в описании приема фильтр "пересмотр(A B P)".

Проверка истинности фильтра в конце программы приема

Проверка истинности некоторых фильтров приема оказывается достаточно трудоемкой. В таких случаях ее выполнение целесообразно предпринимать лишь в самом конце программы приема, после применения более дешевых "отсекающих" средств. Чтобы указать компилятору на необходимость размещения программы фильтра A в конце программы приема, этот фильтр преобразуется к виду "конец(A)". Таким образом можно размечать лишь "корневые" фильтры, не расположенные внутри более сложного фильтра.

Проверка истинности фильтра непосредственно перед обработкой заданного антецедента

В некоторых случаях проверку истинности фильтра следует выполнять не в начале программы приема (так как она сравнительно трудоемкая), но все же до существенно более трудоемкой обработки некоторого антецедента (например, передаваемого проверочному оператору). В этих случаях для сообщения компилятору пожелания о данном размещении фильтра A он преобразуется к виду "проверка(A i)", где i - символьный номер рассматриваемого антецедента. Таким образом размечаются лишь "корневые" вхождения фильтров.

10.4 Типы указателей приемов

Перечислим типы используемых в ГЕНОЛОГе указателей приема - различных информационных элементов, уточняющих компилятору необходимую версию программы приема.

10.4.1 Указатели идентификации

В подавляющем большинстве случаев логические конструкции, используемые в теореме приема, не встречаются в задаче в чистом виде. Для усмотрения возможности применения данной теоремы обычно приходится применять специальные преобразования встречающихся в задаче термов, "подгоняя" их под испольуемые в теореме выражения. При описании приема на ГЕНОЛОГе такие преобразования должны быть явно заданы - с помощью так называемых указателей идентификации. Компилятор ГЕНОЛОГа создает на основе этих указателей программу усмотрения замаскированных возможностей применения теоремы, существенно расширяя тем самым область применимости приема.

Вырожденные значения переменных

1. Если теоремная переменная x встречается в идентифицируемой части теоремы в качестве операнда таких операций $f(x\ Q)$, которые сводятся к Q при

специальном "единичном"значении a переменной x, то использование указателя "единица(a x)" обеспечивает трансляцию приема, при которой идентифицируемый с $f(x\ Q)$ терм задачи t, не имеющий заголовка f, будет автоматически представляться в виде $f(a\ t)$; при этом x будет идентифицироваться с a, а Q - с t. Возможно объединение нескольких указателей "единица(a x_1)", ... "единица(a x_n)" в один указатель "единица(a $x_1 \ldots x_n$)". Для использования данного указателя предварительно должен быть создан соответствующий паре f, a прием справочника "единица". Пример использования указателя "единица" - для идентификации квадратного трехчлена $ax^2 + bx + c$, у которого коэффициенты a, b могут оказаться вырожденными - равными 1. В этом случае вместо указанных в записи общего вида квадратного трехчлена произведений ax^2 и bx будут иметься некоторые выражения, не имеющие вида произведений. Чтобы компилятор мог воспринять их как произведения на единицу, вводится указатель "единица(1 a b)".

2. Иногда в теореме используется ассоциативно - коммутативная операция, некоторые операнды которой при идентификации могут отсутствовать - в этом случае определенным параметрам таких отсутствующих членов присваиваются наперед заданные константные значения (например, нулевое значение коэффициента в отсутствующем слагаемом). Чтобы обозначить такую ситуацию, используется указатель "подстановка(v x a)". Здесь v - указатель вхождения в теорему операнда ассоциативно-коммутативной операции, который может отсутствовать; x - переменная (она должна иметь единственное вхождение в идентифицируемую часть); a - логический символ, с однобуквенным термом (a) которого идентифицируется x при отсутствии операнда v. Пример использования указателя "подстановка" - при задании вида кубического многочлена $ax^3 + bx^2 + cx + d$ уточняется, что члены второй либо первой степени могут отсутствовать. Соответственно, вводятся указатели "подстановка(K_1 b 0)" и "подстановка(K_2 c 0)", где K_1, K_2 - указатели вхождения в теорему членов второй и первой степени. Такие указатели можно использовать одновременно с указателями "единица" для a, b, c. Различные указатели, затрагивающие идентификацию одной и той же переменной, обычно можно вводить одновременно - кроме случаев, когда они явно противоречат друг другу.

3. Если переменная x идентифицируется как "общая часть" нескольких вхождений ассоциативно-коммутативной операции f, то можно разрешить идентифицировать ее в случае пустой общей части с единицей этой операции. Для этого используется указатель "пустоеслово(x)". Разрешается объединять несколько таких указателей в один указатель "пустоеслово($x_1 \ldots x_n$)". Обычно данный указатель не нужен - вместо него берется указатель "единица(a x)". Однако, если в теореме имеется более двух операндов операции f, и нужно предусмотреть возможность вырожденной идентификации данной операции с выражением, не имеющим заголовка f (когда все операнды, кроме одного, обращаются в единицу), то указатель "пустоеслово" дополняет указатель "единица".

Учет внешней одноместной операции либо связки

1. Если одноместная операция f удовлетворяет тождеству $f(f(X)) = X$, то указатель "отрицание(f $x_1 \ldots x_n$)" определяет такую идентификацию переменных

$x_i; i = 1, \ldots, n$, при которой терм задачи t, идентифицируемый с $f(x_i)$, автоматически представляется в виде $f(f(t))$. Заблаговременно должен быть создан прием справочника "отрицание" для f. Пример применения данного указателя - идентификация в алгебре логики выражения, содержащего одновременно и переменную, и ее отрицание. Если выражение вида $a \vee \neg a \cdot b$ при идентификации снабдить указателем "отрицание(отр a)", то становится возможной идентификация a с $\neg c$, а отрицания a - с c. Здесь "отр" - логический символ для булева отрицания $\neg$.

2. Если переменная x имеет единственное вхождение в таком идентифицируемом подтерме $t(x)$ теоремы приема, что для одноместной операции f выполнено тождественно $f(t(X)) = t(f(X))$, то указатель "заменазнака(f x)" определяет идентификацию терма $t(x)$, при который идентифицируемый с ним терм вида $f(t(A))$ автоматически преобразуется к виду $t(f(A))$, т.е. внешний знак f передается переменной x. Подтерм $t(x)$ определяется по идентифицируемому вхождению переменной x как максимальный подтерм, для которого возможна указанная передача знака f по цепочке вложенных операций (используются приемы справочника "заменазнака" для этих операций, которые должны быть заблаговременно созданы). Возможна идентификация с передачей знака f при наличии нескольких непересекающихся подтермов $t_1(x), \ldots, t_m(x)$. Несколько указателей "заменазнака(f x_1)", ..., "заменазнака(f x_n)" могут быть объединены в один указатель "заменазнака(f $x_1 \ldots x_n$)". Пример использования указателя "заменазнака" - идентификация квадратного трехчлена $ax^2 + bx + c$, у которого перед членами второй и первой степени могут стоять знаки "минус". В этом случае указатель "заменазнака(минус a b)"обеспечивает передачу данных минусов коэффициентам a, b.

3. Если консеквент приема замены имеет вид "не(A)", то по умолчанию будет идентифицироваться вхождение утверждения A, заменяемое приемом на константу "ложь". В тех случаях, когда требуется идентифицировать именно вхождение утверждения "не(A)", заменяемое на константу "истина", применяется указатель идентификации "прямойответ". Пример применения данного указателя - приемы усмотрения истинности условий задачи на описание, имеющих вид отрицания равенства нулю. Такие вспомогательные условия обычно возникают при сопровождении основных условий по области допустимых значений. В некоторый момент решения надобность в них отпадает, и тогда предпринимается попытка отбросить данные условия, предварительно убедившись в том, что они вытекают из прочих условий. Без указателя "прямойответ" данный прием предпринимал бы также попытку усмотрения ложности имеющихся в условиях уравнений с нулем в правой части, что нецелесообразно.

Перестановки операндов

1. Если заменяемый терм приема тождественной либо эквивалентной замены (в том числе приема нормализатора) имеет вид $f(A_1\ A_2)$, причем при идентификации допустима перестановка операндов A_1, A_2 с одновременной перестановкой операндов B_1, B_2 заменяющего терма $f(B_1\ B_2)$, то используется указатель "дробь". Заменяющий терм может и не иметь вида $f(B_1\ B_2)$; тогда используется искусственное его представление в таком виде, использующее единицу операции f. Пример использования указателя "дробь" - тригонометрическое

сокращение с помощью тождества $a(\sin b)^n/c(\tan b)^n = a(\cos b)^n/c$. Если ввести данный указатель, то сокращение будет выполняться и при перестановке числителя со знаменателем.

2. Указатель "дробь(v $w_1 \ldots w_n$)" обобщает указатель "дробь". У него v - указатель вхождения в теорему идентифицируемого терма $f(A_1\ A_2)$, операнды которого A_1 и A_2 при идентификации могут быть переставлены; $w_1, \ldots, w_n$ - список указателей вхождений двуместных операций либо отношений, при идентификации либо формировании которых осуществляется синхронная с v перестановка операндов. Пример использования данного указателя - декомпозиция строгого неравенства для произведения по теореме $0 \le a \to (0 < ab \leftrightarrow 0 < a\ \&\ 0 < b)$. Указатель "дробь(фикс(0 1)фикс(0 2 2))" выделяет вхождения в теорему неравенства $0 < ab$ и неравенства $0 < b$. При его наличии прием будет выполнять также и декомпозицию неравенств вида $ab < 0$ на $0 < a, b < 0$.

3. Указатель "циклупорядочение(v)" используется в тех случаях, когда идентификация терма $f(A_1 \ldots A_n)$, расположенного по указателю вхождения v, выполняется с возможными циклическими перестановками операндов $A_1, \ldots, A_n$. Если допускается лишь однократная циклическая перестановка, при которой перечисление операндов начинается со второго операнда A_2, то используется указатель "второйоперанд(v)". Пример использования указателя "циклупорядочение" - идентификация утверждения "четырехугольник($ABCD$)", которую можно начинать с любой из расположенных по циклу вершин A, B, C, D. Пример использования указателя "второйоперанд" - идентификация утверждения "параллелограмм($ABCD$)" в той ситуации, когда достаточно различать лишь два случая - выделение пары параллельных сторон AB, CD либо пары BC, AD.

4. Указатель "контрсерия(v)" используется в тех случаях, когда идентификация терма $f(A_1 \ldots A_n)$, расположенного по указателю вхождения v, выполняется с возможным изменением порядка операндов на противоположный. Если же, кроме этого, возможны также любые циклические перестановки операндов, то используется указатель "подобны(v)". Пример использования указателя "контрсерия" - идентификация утверждения "трапеция($ABCD$)", если в приеме нужно обеспечить возможности идентификации, симметричные относительно вертикального для трапеции направления. При этом одновременно рассматриваются два различных обозначения одной и той же трапеции - $ABCD$ и $DCBA$. Пример использования указателя "подобны" - идентификация утверждения "четырехугольник($ABCD$)" в тех приемах, где нужно учесть не только возможные варианты обозначения $BCDA$, $CDAB$, ..., но и варианты $DCBA$, $CBAD$, ...

5. Указатель "множество(v)" используется в тех случаях, когда идентификация терма $f(A_1 \ldots A_n)$, расположенного по указателю вхождения v, происходит с произвольной перестановкой операндов. Если данный терм определяет набор, идентифицируемый без учета порядка элементов (этот набор может задаваться явным перечислением элементов - "набор($A_1 \ldots A_n$)" либо косвенным образом - "префикс($A\ B$)", "конкатенация(набор($A_1\ A_2$)B)", где для идентификации выделяются один либо два элемента набора), то используется указатель "список(v)". Пример использования указателя "множество" - идентификация выражения "плоскость(ABC)" без учета порядка точек A, B, C. Пример использования указателя "список(фикс(0 1 1))" - применение тождества "равно(перечень

(конкатенация(набор(a a) b)) перечень (префикс (a b)))" для исключения повторяющихся элементов в списке, определяющем конечное множество. Этот указатель выделяет вхождение выражения "конкатенация(набор(a a)b)", которое идентифицируется с набором из задачи так, что различные позиции, на которых встречаются одинаковые элементы a, - произвольны.

6. При идентификации операндов коммутативной операции по умолчанию учитывается возможность перестановки операндов. Кроме того, рассматриваются символы со специальной симметрией (см ниже), идентификация которых также по умолчанию производится с допустимыми перестановками операндов. Если же такую перестановку требуется заблокировать - для коммутативной неассоциативной операции либо операции со специальной симметрией - то используется указатель "коммутативно(v)"; v - указатель вхождения в теорему этой операции.

Альтернативные заголовки подтермов

1. В некоторых случаях теорема приема может быть модифицирована при одновременной замене заголовков операций либо отношений, расположенных на заданных вхождениях. Эти вхождения выделяются как в идентифицируемой части теоремы, так и в тех ее частях, которые определяют новые термы, создаваемые приемом. Возможность модификации определяется указателем "альтернатива($C_1 \dots C_n$)", где каждое C_i - группа операндов вида $A_1 \dots A_m$ B, причем $A_1, \dots, A_m$ - указатели вхождений в теорему термов, которые в модифицированной теореме все имеют альтернативный заголовок B. Требуется, чтобы альтернативные заголовки имели одновременно все термы по указателям вхождений из $C_1, \dots, C_n$. Если имеется несколько различных указателей "альтернатива(...)", то модификации теоремы согласно этим указателям осуществляются независимо (т.е. возможны любые их комбинации). При этом различные группы термов, выделенные в теореме данными указателями, не должны пересекаться друг с другом. В качестве примера рассмотрим теорему $0 \le \mathrm{arctg}(a) - \mathrm{arctg}(b) \leftrightarrow 0 \le a - b$, определяющую прием исключения арктангенсов в неравенствах. Чтобы прием обрабатывал как нестрогие, так и строгие неравенства, вводится указатель "альтернатива(фикс(0 1)фикс(0 2)меньше)".

2. Если в теореме выделяется вхождение операции либо отношения, для которого при идентификации допускается более одного альтернативного заголовка, то используется указатель идентификации "вариант(A $B_1 \dots B_n$)", где A - указатель данного вхождения в теорему; $B_1, \dots, B_n$ - список всех альтернативных заголовков. В качестве примера рассмотрим теорему "вершина(A, E) & линвторпорядка(E) $\rightarrow A \in E$", определяющую прием вывода следствий в задачах по аналитической геометрии. Чтобы прием мог срабатывать на кривых второго порядка конкретных типов, вводится указатель "вариант(фикс(2)эллипс гипербола парабола)".

3. Для некоторых ассоциативно-коммутативных операций f предусмотрена специальная процедура, которая может при идентификации вводить искусственное представление терма в виде $f(A_1 \dots A_n)$, даже если он первоначально не имел заголовка f. Например, степенное выражение при идентификации может рассматриваться как произведение двух других степенных выражений. Такие про-

цедуры определяются справочником "пересечениесписков" (см. об этом справочнике в начале следующего подраздела, посвященного идентификации операндов ассоциативно - коммутативных операций). Чтобы заблокировать проверку наличия у идентифицирующего терма заголовка f, в таких случаях рекомендуется применять указатель идентификации "пересечениесписков($K_1 \ldots K_m$)", где $K_1, \ldots, K_m$ - указатели вхождений в теорему идентифицируемых операций f. Отметим, что иногда такая блокировка выполняется компилятором ГЕНОЛОГа автоматически - например, при идентификации термов $f(x\ T)$, где переменная x может принимать вырожденное "единичное" значение. Как правило, специальная блокировка нужна для операций f с более чем двумя операндами.

4. Если необходимо при идентификации выражения "частнпроизв($A_1\ A_2\ A_3$)" для частной производной автоматически приводить к виду "частнпроизв($B\ 1\ C$)" выражения "производная($B\ C$)" для обычной производной, то используется указатель "частнпроизв".

Идентификация операндов ассоциативно - коммутативной операции

1. При идентификации операндов ассоциативно - коммутативных операций может использоваться специальная процедура "пересечения" списков операндов двух операций, определяющая общую их переменную. Фактически это пока сделано лишь для вещественного и комплексного умножений - здесь "пересечение" понимается как определение некоторой версии наибольшего общего делителя двух одночленов. Однако, в компиляторе ГЕНОЛОГа предусмотрена общая конструкция, позволяющая при необходимости подключать и другие процедуры такого типа. Компилятор использует для определения данной процедуры по символу f ассоциативно - коммутативной операции справочник "пересечениесписков". Эта же процедура определяет "остаточные" после определения общей части фрагменты (в случае умножения - частные от деления одночленов на их наибольший общий делитель). Такое усиление идентификации вводится по умолчанию. В тех редких случаях, когда оно не нужно и лишь замедляет работу приема, применяется указатель идентификации "набороперандов($K_1 \ldots K_n$)". Здесь $K_1, \ldots, K_n$ - указатели вхождений в теорему некоторых ассоциативно - коммутативных операций, обрабатываемых компилятором без применения указанных процедур.

2. Приведем еще два способа отключения указанной выше усиленной идентификации. Первый из них связан с идентификацией логического символа A - операнда ассоциативно - коммутативной операции f. Указатель идентификации "логсимвол($A\ f$)" означает, что A должно быть усмотрено "в чистом виде" как операнд этой операции. Если в случае умножения не применять данный указатель, то прием будет предпринимать попытку деления соответствующего одночлена на константу A. Второй способ - при идентификации переменной x из пересечения наборов операндов нескольких одноименных ассоциативно - коммутативных операций использовать указатель "операнды(x)" вместо явного перечисления в указателе "набороперандов(...)" вхождений этих операций.

3. Если требуется идентифицировать переменную X с результатом $f(B_1 \ldots B_m)$ выделения среди операндов $A_1, \ldots, A_n$ ассоциативно - коммутативной операции

f подмножества всех операндов B_i, удовлетворяющих заданному условию, то используется указатель "перечень(X $P(X)$)", где P - условие на операнд B_i, в котором этот операнд обозначен посредством самой переменной X (это условие сформулировано на том же языке, на котором определяются фильтры приемы). При использовании данного указателя предполагается, что переменная X имеет ровно одно вхождение в идентифицируемую часть - в качестве операнда некоторой ассоциативно - коммутативной операции f. В качестве примера рассмотрим теорему $a + b = c \leftrightarrow a = c - b$, используемую для перенесения в правую часть уравнения всех известных слагаемых b. Указатель "перечень(a не(известно(a)))" определяет идентификацию переменной a с суммой всех слагаемых, содержащих неизвестные; тогда сумма остальных слагаемых идентифицируется с b.

4. Если переменная X имеет несколько вхождений в идентифицируемую часть, причем все они суть операнды различных ассоциативно - коммутативных операций с одним и тем же заголовком f, то возможна идентификация ее в виде $f(B_1 \dots B_m)$, где $B_1, \dots, B_m$ - пересечение всех подмножеств операндов B указанных операций, удовлетворяющих условию $P(B)$. Для этого используется указатель "выписка(X $P(X)$)". В качестве примера рассмотрим теорему $ab + ac = a(b + c)$, применяемую в нормализаторе выявления повторяющихся вхождений выражений с неизвестными и обеспечивающую вынесение за скобку лишь известных общих множителей слагаемых. Вынесение за скобку неизвестного множителя в таком нормализаторе может разрушить произведение с неизвестными сомножителями, встречающееся повторно в другой части преобразуемого терма, и поэтому нежелательно. Здесь используется указатель "выписка(a известно(a))".

5. Возможна идентификация переменной X с коэффициентом, возникающим после приведения нескольких "подобных членов". Эта переменная должна иметь единственное вхождение в идентифицируемую часть теоремы, причем внутри терма $f(\dots g(X\ A)\dots)$, где f, g - ассоциативно - коммутативные операции; A - некоторый терм. При идентификации значением X становится терм вида $f(B_1 \dots B_n)$, где $g(A\ B_1), \dots, g(A\ B_n)$ - все операнды операции $f(\dots g(X\ A)\dots)$, множество g - членов которых включает множество g - членов терма A. Указатель идентификации здесь имеет вид "группировка(X)". В качестве примера рассмотрим идентификацию квадратного трехчлена $ad^2 + bd + c$ при разложении на множители. Сначала идентифицируется выражение d^2, и тем самым - основание степени d, а затем группируются в ad^2 все слагаемые с d^2 и в bd - все слагаемые с d. Используются указатели "группировка(a)", "группировка(b)".

6. В случае обычных сложения и умножения идентификация коэффициента X, возникающего после приведения подобных членов, может быть усилена таким образом, чтобы коэффициенты извлекались также из дробных слагаемых (например, коэффициент $2/3$ из слагаемого $2A/3$). Переменная X должна иметь единственное вхождение в идентифицируемой части - в слагаемом вида AX либо AX/B. Она идентифицируется после идентификации A и B, как сумма коэффициентов при A (соответственно, при A/B) всех слагаемых рассматриваемой суммы, представимых в таком виде. Указатель идентификации в данном случае имеет вид "коэффициент(X K)", где K - указатель вхождения в теорему слагаемого с переменной X.

7. Иногда встречается несколько другая ситуация - требуется выделить группу "подобных членов", общий коэффициент K которых уже известен. В этом случае внутри идентифицируемой части теоремы выделяется терм вида $g(\ldots f(X\ A)\ldots)$, где f, g - ассоциативно - коммутативные символы. Перед идентификацией операнда $f(X\ A)$ терм, идентифицированный с X, представляется в виде $g(t_1 \ldots t_n)$, и указанный операнд идентифицируется с группой операндов $f(t_1\ A), \ldots, f(t_n\ A)$ операции g. Указатель идентификации здесь имеет вид "группировки(X K)", где K - указатель вхождения в теорему операнда $f(X\ A)$. В качестве примера рассмотрим теорему $ab + bd = (a + d)b$, применяемую при разложении на множители в ситуации, когда выражение b, идентифицированное как сомножитель слагаемого ab, имеет вид суммы $b_1 + \ldots + b_n$, а выражение bd составляется при идентификации из слагаемых $b_1 d, \ldots, b_n d$. Используется указатель "группировки(b фикс(0 1 2))".

8. Если вместо операнда $f(X\ A)$ в предыдущем случае рассматривается операнд вида $f(X)$, который (уже после идентификации X с $g(t_1 \ldots t_n)$) должен быть идентифицирован с группой $f(t_1), \ldots, f(t_n)$ операндов операции g (при формировании этих операндов двойные применения операции f отбрасываются, т.е. предполагается истинным тождество $f(f(x)) = x$, то используется указатель идентификации "нормзнака(X f g)". В качестве примера рассмотрим теорему $a+b = 0\ \&\ a-b = 0 \leftrightarrow a = 0\ \&\ b = 0$, применяемую для тривиального упрощения системы двух уравнений. Сначала идентифицируется a, как общая часть групп слагаемых левых частей этих уравнений. Это позволяет идентифицировать b из первого уравнения - как сумму оставшихся слагаемых, а затем идентифицировать $-b$ во втором уравнении как сумму этих же слагаемых, взятых с обратными знаками. Используется указатель "нормзнака(b минус плюс)".

9. Если переменную X требуется идентифицировать с отдельным операндом ассоциативно - коммутативной операции f, не прибегая к каким-либо группировкам, то используется указатель идентификации "операнд(X K)". Здесь K - указатель вхождения в теорему данной операции. Если в идентифицируемой части теоремы переменная X имеет единственное вхождение (именно, как операнда рассматриваемой операции), то возможно указание вместо K символа операции f. В качестве примера рассмотрим теорему $b \le 0 \rightarrow -b|a| = |ab|$, используемую для вынесения из-под модуля неположительных сомножителей. Если не ввести указатель "операнд(b фикс(0 2 1))", то компилятор может идентифицировать с сомножителем не b, а переменную a, в то время как b, для которого и проверяется неположительность, будет идентифицироваться с произведением оставшихся сомножителей.

10. Если переменная X встречается в качестве операнда нескольких ассоциативно - коммутативных операций с общим заголовком f, то она по умолчанию идентифицируется как результат применения f к списку всех общих операндов этих операций, оставшихся после идентификации более явно обозначенных операндов. Можно, однако, выделить лишь подмножество M указанных ассоциативно - коммутативных операций, из рассмотрения которых и будет идентифицироваться X - как пересечение только их остаточных операндов. Для этого используется указатель идентификации "пересечение(X $K_1 \ldots K_n$)", где $K_1, \ldots, K_n$ - указатели вхождений в теорему операций подмножества M. В качестве примера рассмотрим теорему $ca^2 - 2cab + cb^2 + da - db = (a - b)(d + ac - cb)$,

применяемую для разложения на множители. Переменную c здесь достаточно идентифицировать из рассмотрения слагаемых ca^2, cb^2, что и достигается с помощью указателя "пересечение(c фикс(0 1 1) фикс(0 1 3))".

11. Иногда при идентификации ассоциативно - коммутативной операции $f(A_1 \dots A_n)$ бывает нужно идентифицировать некоторое A_i только с первым операндом соответствующей операции из идентифицируемого терма. Например, это может быть полезным при заменах рекурсивного типа - для отсечения попыток применить рекурсию с участием внутреннего операнда, если она не удалась для первого операнда. Здесь используется указатель идентификации "первыйтерм(K)"; K - указатель вхождения в теорему операнда A_i. В качестве примера рассмотрим теорему "нечетнаяфункция($\lambda_x(f(x), h(x))$) & нечетнаяфункция ($\lambda_x(g(x), h(x))$) $\rightarrow$ нечетнаяфункция($\lambda_x(f(x) + g(x), h(x))$)", применяемую для усмотрения нечетной функции, являющейся суммой нечетных функций. Нет смысла перебирать всевозможные перестановки слагаемых - достаточно провести рекурсию, рассматривая их лишь последовательно слева направо. Для этого служит указатель "первыйтерм(фикс(0 1 1))".

12. Если в приеме замены заголовком заменяемого терма служит конъюнкция либо дизъюнкция, то по умолчанию она идентифицируется так, чтобы эту конъюнкцию либо дизъюнкцию можно было усмотреть из внешнего квантора общности (возможно, при перенесении некоторых антецедентов в консеквент) либо из внешнего отрицания двойственной логической связки. Чтобы заблокировать такой режим идентификации, используется указатель "дизъюнктоперанд". В качестве примера рассмотрим теорему $a \in b \cup c \leftrightarrow a \in b \vee a \in c$, используемую для группировки известных выражений b, c при неизвестном a в условии задачи на описание. Такая группировка внутри квантора общности либо конъюнкции отрицаний принадлежности малополезна, и для отсечения этих случаев служит указатель "дизъюнктоперанд".

Одновременное изменение знаков операндов ассоциативно - коммутативной операции

Если идентифицируется выражение вида $f(A_1 \dots A_n)$, где f есть символ ассоциативно - коммутативной операции, причем одноместная операция g удовлетворяет тождеству $g(g(x)) = x$, то при идентификации можно разрешить одновременное изменение "знака" g у всех операндов $A_1, \dots, A_n$. В тех случаях, когда указанное выражение является заменяемой частью некоторого тождества, при формировании заменяющего выражения также будет происходить одновременная замена знака g у всех f - членов. В противном случае данная альтернативная идентификация на формировании заменяющих термов никак не сказывается. Указателем идентификации служит терм "знаксуммы(g K)", где K есть указатель вхождения в теорему выражения $f(A_1 \dots A_n)$. Возможно применение этого указателя внутри фильтра "контекст(...)" - тогда в качестве K берется само выражение $f(A_1 \dots A_n)$. В качестве примера рассмотрим теорему $a^2 + b^2 - 2ab = (a - b)^2$, применяемую для разложения на множители. Чтобы прием позволял разложить на множители также выражение $2ab - a^2 - b^2$, вводится указатель "знаксуммы(минус фикс(0 1))".

Идентификация кванторов и описателей

1. В задачах обычно применяется стандартизация, при которой кванторная импликация с конъюнктивным консеквентом преобразуется в конъюнкцию нескольких кванторных импликаций. Все эти кванторные импликации имеют один и тот же список антецедентов. Чтобы при идентификации автоматически выполнять "сборку" ранее расформированных кванторных импликаций, используется указатель "консеквент(A)", где A - указатель вхождения в теорему консеквента идентифицируемой кванторной импликации. Этот консеквент будет идентифицироваться с конъюнкцией консеквентов всех кванторных импликаций из области идентификации, имеющих заданный список антецедентов (к моменту идентификации кванторной импликации все переменные ее антецедентов должны быть уже идентифицированы). Единственный прием, где пока используется такой указатель - доказательства индукцией по длине набора. К моменту применения этого приема в посылках задачи должны иметься кванторные импликации, определяющие априори известные свойства i - го элемента набора, которые и группируются указанным выше образом.

2. Если теорема приема замены представляет собой эквивалентность с квантором K в заменяемой части, то можно усилить идентификацию, разрешая выделять K либо отрицание K из квантора с более длинной связывающей приставкой путем группировки всех подкванторных утверждений, зависящих от переменных некоторого подмножества этой приставки. Такое усиление обеспечивается указателем "кванторнаясвертка". В качестве примера рассмотрим теорему $a \subseteq b \leftrightarrow \forall c(c \in a \rightarrow c \in b)$, применяемую для перехода от кванторной импликации к условию включения множеств. При наличии указателя "кванторнаясвертка" прием будет выполнять такой переход в утверждении вида $\forall xy(x \in y \ \& \ y \in a \rightarrow x \in b)$, преобразуя его в утверждение $\forall y(y \in a \rightarrow y \subseteq b)$ с помощью перегруппировки $\forall y(y \in a \rightarrow \forall x(x \in y \rightarrow x \in b))$.

3. По умолчанию компилятор ГЕНОЛОГа организует идентификацию заменяемого квантора с попыткой усмотреть его отрицание в противоположном кванторе. Если такая идентификация нежелательна, то используется указатель "внешнийквантор(A)", где A - указатель вхождения данного квантора в теорему. Примеры использования указателя - приемы редактирования параметрических описаний решений тригонометрических уравнений либо неравенств, где изначально предполагается использование именно квантора существования и нет смысла тратить время на анализ возможных вхождений в задачу кванторов общности.

4. Если в теореме приема встречается квантор либо описатель, то компилятор обеспечивает проверку невхождения переменных его связывающей приставки в термы, идентифицированные с подкванторными переменными - обычными либо функциональными, т.е. выражениями вида $f(x)$, не зависящими от этой связывающей приставки. Эту проверку можно отменить, используя указатель "обобщподст(A)", где A - указатель вхождения рассматриваемого квантора либо описателя в теорему. В качестве примера рассмотрим теорему приема $\neg(b = 0) \rightarrow set_x(a/b + c = 0 \ \& \ A(x)) = set_x(a + bc = 0 \ \& \ A(x))$, применяемую для устранения знаменателей в уравнениях кривых либо плоскостей. Чтобы не перегружать теорему обозначениями $a(x), b(x), c(x)$, вместо них используются a, b, c, но зато введен указатель "обобщподст(фикс(0 1))".

5. Указанную выше отмену проверки невхождения переменных связывающей приставки можно сделать частичной. Указатель идентификации "содержится(x y_1 ... y_n)" выделяет те переменные $y_1, \ldots, y_n$, встречающиеся под квантором либо описателем по переменной x, которые могут идентифицироваться с термами, содержащими переменную, идентифицированную с x.

6. Другой способ сделать отмену проверки невхождения переменных связывающей приставки частичной - указатель "Входит(x)", определяющий ту переменную x связывающих приставок, для которой не предпринимается проверка невхождения идентифицированных с ней переменных в термы, идентифицированные с прочими переменными под кванторами и описателями.

7. При идентификации кванторов и описателей иногда бывает удобно выделить ряд конкретных переменных в связывающей приставке, а все оставшиеся переменные сгруппировать в набор, обозначаемый в теореме приема единственной (условной) переменной x. Такой набор, в частности, может оказаться пустым. Данный режим вводится указателем идентификации "кортежпеременных(x)". Заметим, что список выделяемых "штучным образом" переменных может быть пустым; в случае его непустоты такие переменные идентифицируются в связывающей приставке без учета порядка. В качестве примера рассмотрим теорему $set_x((f(x) \vee g(x))\ \&\ h(x)) = set_x(f(x)\ \&\ h(x)) \cup set_x(g(x)\ \&\ h(x))$, используемую для перехода от дизъюнкции в условии принадлежности множеству к объединению множеств. Прием имеет указатель "кортежпеременных(x)", что позволяет ему обрабатывать описания множеств наборов произвольной длины.

8. Чтобы идентифицировать теоремную переменную x с произвольным элементом связывающей приставки, без учета порядка ее переменных, применяется указатель "элемент(x)".

9. В теореме приема допускается использовать запись вида $\forall_x(f(x))$. Если сопроводить ее указателем "импликация(f)", то выражение $f(x)$ идентифицируется со вспомогательным термом "то(A_0 $A_1 \ldots A_n$)", где A_0 - консеквент; $A_1, \ldots, A_n$ - антецеденты идентифицирующей кванторной импликации, расположенной в задаче. В заменяющем терме выражение $f(B)$ может быть использовано только непосредственно под квантором, где оно расформируется на антецеденты и консеквент. Данный указатель применяется лишь в приемах вывода теорем, обычно в сочетании с указателем "вхождение(f)".

10. Если внутри теоремы приема встречается кванторная импликация $\forall_x(A_1\ \&\ \ldots\ \&\ A_n \rightarrow A_0)$, у которой некоторое A_i представляет собой функциональную переменную $F(x)$ либо обычную переменную F, и нужно идентифицировать его с отдельным антецедентом идентифицирующей кванторной импликации, то используется указатель "антецедент(F P)". Здесь P - указатель вхождения в теорему данной кванторной импликации.

11. Иногда бывает удобно выделить в идентифицируемом кванторе единственную переменную x связывающей приставки, а остальные переменные (которые могут иметься либо отсутствовать) вообще не указывать. Если в теореме формируются новые кванторные конструкции с x, то все отличные от x переменные исходной связывающей приставки переносятся в них по умолчанию. Для такого режима служит указатель "связка(x)".

Идентификация функциональных переменных

В теореме приема могут встречаться подтермы вида $f(x), f(x,y), f(x,y,z), \ldots$, где $f, x, y, z, \ldots$ - переменные. Во внутреннем логическом представлении (в скобочной записи) они соответствуют выражениям "значение(f x)", "значение(f набор(x y))", "значение(f набор(x y z))", ... Такие выражения называем функциональными переменными. Для идентификации функциональной переменной предусмотрены перечисляемые ниже указатели; при отсутствии этих указателей выражения "значение(f ...)" идентифицируются с имеющими заголовок "значение" термами задачи обычным образом.

1. Наиболее часто встречающийся случай - идентификация терма $f(x_1 \ldots x_n)$ с произвольным подтермом задачи T (здесь n может равняться 1, причем переменные $x_1, \ldots, x_n$ попарно различны). Если переменные $x_1, \ldots, x_n$ согласно другим фрагментам теоремы окажутся идентифицированы с некоторыми переменными задачи $y_1, \ldots, y_n$, то далее T начинает играть роль "шаблона": остальные подтермы теоремы вида $f(t_1 \ldots t_n)$, если они есть, при идентификации либо формировании новых термов получаются как результат подстановки в T вместо переменных $y_1, \ldots, y_n$ термов, определяемых по результатам идентификации термами $t_1, \ldots, t_n$. Указатель на идентификацию данного типа имеет вид "отображение(f)". Рекомендуется объединять несколько различных указателей такого типа в общий указатель "отображение(f $g \ldots h$)". Заметим, что если в теореме f встречается только в виде $f(x)$, то x может (до идентификации $f(x)$) быть идентифицировано не с переменной, а со списком переменных (например, со связывающей приставкой квантора либо описателя). В качестве примера рассмотрим теорему $c = \lim_{x\to a} f(x)$ & число(c) & $d = \lim_{x\to a} g(x)$ & число(d) $\to \lim_{x\to a}(f(x) + g(x)) = c + d$, используемую при вычислении предела суммы, если пределы слагаемых существуют и конечны. Наличие указателя "отображение(f g)" позволяет идентифицировать $f(x), g(x)$ с произвольными выражениями, не обязательно даже содержащими переменную x.

2. Если при идентификации терма $f(t)$ все переменные аргумента t уже идентифицированы так, что этот аргумент задает терм T "в переменных задачи", то можно выделить в текущем (идентифицируемом с $f(t)$) терме A все вхождения $v_1, \ldots, v_n$ подтерма T, и считать, что остальные вхождения в теорему термов $f(p)$ задают результаты замены в A вхождений $v_1, \ldots v_n$ на определяемый аргументом p терм P. Такой режим вводится указателем "заменатермов(K)", где K - указатель вхождения в теорему терма $f(t)$. В качестве примера рассмотрим теорему $f(y) = a \to (a$ при $x = y$, иначе $f(x)) = f(x)$, используемую для исключения условного выражения, у которого особый случай $x = y$ сводится к общему случаю. Указатель "заменатермов(фикс(0 1 3))" позволяет рассматривать выражение, идентифицированное с $f(x)$, как шаблон для построения $f(y)$.

3. Если необходимо потребовать, чтобы идентифицируемый с теоремным термом $f(t)$ терм задачи A содержал переменную X только внутри вхождений терма T, определяемого аргументом t, то используется указатель "новаргумент(f x N)". Предполагается, что к моменту идентификации $f(t)$ все переменные из t уже идентифицированы, а теоремная переменная x идентифицирована с переменной задачи X. N - либо логический символ "фикс", либо название нормализа-

тора, применяемого к A перед попыткой выделения вхождений терма T. Такой нормализатор в качестве входного комментария получает набор (новаргумент T X); он пытается преобразовать "неявные" вхождения терма T в терм A в явные. Если N = "фикс", то A используется без преобразований. После проведения идентификации терма $f(t)$ остальные вхождения в теорему термов $f(p)$ определяют результаты замены всех вхождений T в преобразованный нормализатором N терм A на терм P, определяемый аргументом p. В качестве примера рассмотрим теорему

$$\int \frac{(1-x^2)^{\frac{a-1}{2}} f(x)}{g(x)} dx = \lambda_x(h(x), \text{число}(x)) \to \int \frac{(\sin x)^a f(\cos x)}{g(\cos x)} dx =$$

$$= \lambda_x(-h(\cos x), \text{число}(x)),$$

позволяющую вычислять неопределенный интеграл путем замены $\cos x = y$. Указатели "новаргумент(f x извлечение)", "новаргумент(g x извлечение)" определяют попытки преобразования выражений A, B, идентифицируемых с $f($ $\cos x)$, $g(\cos x)$, к виду, в котором переменная интегрирования x встречалась бы только как $\cos x$. Для этого применяется нормализатор "извлечение", имеющий (в том числе) ряд несложных приемов перехода к косинусам.

4. Указатель "новаргумент", введенный в предыдущем пункте, допускает обобщение на случай идентификации "многоместного" теоремного терма $f(t_1 \dots t_n)$. Здесь он уже имеет вид "новаргумент(f набор($x_1 \dots x_m$) N)". Проверяется, что идентифицируемое с указанным термом выражение A содержит переменные, идентифицированные с переменными $x_1, \dots, x_m$, только внутри вхождений термов $t_1, \dots, t_n$, все переменные которых заранее идентифицированы. В случае $m = 1$ символ "набор" перед переменной x_1 отбрасывается. После идентификации формирование термов $f(p_1 \dots p_n)$ осуществляется на основе выделения в A вхождений термов $t_1, \dots, t_n$.

5. Еще одно обобщение указателя "новаргумент" связано с идентификацией, допускающей переобозначения связанных переменных. Это обобщение используется крайне редко (пока - лишь в приемах, связанных с решением дифференциальных уравнений порядка выше первого). По сравнению с предыдущим обобщением, здесь добавляется еще список $a_1, \dots, a_n$, каждый элемент которого - либо переменная, либо 0. Если a_i - переменная, то при выделении вхождения выражения t_i в терме A разрешается переобозначать ее внутри этого вхождения на переменную, связанную внешним по отношению к данному вхождению квантором либо описателем. Если $a_i = 0$, то никакие переобозначения внутри t_i не допускаются. После идентификации формирование термов $f(p_1 \dots p_n)$ происходит с переобозначением переменных a_i на те же связанные переменные. Указатель, вводящий данный режим идентификации, имеет вид "функаргумент(f набор($x_1 \dots x_m$) набор($a_1 \dots a_n$) N)". В качестве примера ситуации, где он может понадобиться, рассмотрим идентификацию терма $f(g'(x))$ с термом, имеющим вид второй производной $g''(x)$. Нормализатор N преобразует вторую производную к виду повторной производной $g'(g'(x))$. На логическом языке эта запись выглядит как "производная(отображение(y число(y) производная(отображение(z число(z) $g(z)$)y))x)". Внутренняя производная здесь берется в точке, обозначенной некоторой вспомогательной переменной y, связанной внешней

производной. Поэтому при идентификации этой производной с $g'(x)$ понадобится переобозначение x на y.

6. В ряде случаев требуется идентифицировать функциональную переменную с символом операции либо отношения. Здесь терм $f(t)$ либо $f(t_1 \dots t_n)$ (в скобочной записи последний представлен как "значение(f набор($t_1 \dots t_n$))") идентифицируется с термом $F(T)$ либо $F(T_1 \dots T_n)$, где F - логический символ. После этого остальные "теоремные" термы вида $f(r)$,$f(r_1 \dots r_n)$ определяют термы $F(R)$,$F(R_1 \dots R_n)$. Указателем на данный режим идентификации служит "символ(f)". Обычно этот указатель используется в приемах текстового анализатора либо в приемах логического вывода базы теорем.

7. Если выражение $f(A)$ должно идентифицироваться с многочленом от A, то используется указатель идентификации "сммногочлен(f S)". Здесь S указывает на способ определения выражения A при идентификации. Если S - символ "неизвестные", то A выявляется из рассмотрения всех содержащих неизвестные задачи или пакетного оператора множителей одночленов; если S есть теоремная переменная, то A выявляется из всех содержащих идентифицированную с S переменную множителей одночленов; если S имеет вид "терм(T)", то A определяется операторным выражением T. Наконец, S вообще может отсутствовать, и тогда A выявляется из рассмотрения всех неконстантных множителей одночленов. Во всех перечисленных случаях (кроме случая явного указания A посредством записи "терм(T)") идентификация $f(A)$ определяет как многочлен, так и выражение A. После такой идентификации остальные встречающиеся в теореме термы вида $f(B)$ определяют многочлены с теми же коэффициентами, что и $f(A)$, примененные к выражению B. Если для выделенных указателями "сммногочлен(a x)", "сммногочлен(f x)"переменных a, f имеется антецедент теоремы $a = \lambda_x(bx^c + f(x))$, причем этот антецедент - идентифицирующий (см. следующий раздел), то переменная c будет идентифицироваться со степенью многочлена, переменная b - с коэффициентом при старшем члене, а функциональная переменная $f(x)$ - с суммой оставшихся членов. В этом случае указатель "отображение(f)" не нужен. Заметим, что здесь и далее в указателях ГЕНОЛОГа под многочленами понимаются термы, имеющие вид многочленов, а не формальные многочлены, рассматриваемые в алгебре.

8. В некоторых случаях переменную f, идентифицированную обычным образом с термом A (т.е. с использованием такого ее вхождения в теорему, которое не имеет вида $f(t)$), бывает необходимо использовать после этого как функциональную переменную. Выражения $f(r)$ либо $f(r_1 \dots r_n)$ при этом будут определять результаты замены в A некоторых переменных y либо $y_1, \dots, y_n$ на термы R либо $R_1, \dots, R_n$, соответствующие $r, r_1, \dots, r_n$. Чтобы ввести такой режим, используется указатель идентификации "функция(f x)" либо "функция(f набор($x_1 \dots x_n$))". Теоремные переменные $x, x_1, \dots, x_n$ должны быть идентифицированы до формирования выражений $f(r), f(r_1 \dots r_n)$ независимым образом; они определяют указанные выше переменные задачи $y, y_1, \dots, y_n$.

9. Иногда бывает нужно идентифицировать с заданным теоремным выражением t какой-то произвольный подтерм заданного терма задачи T , безотносительно к тому, где этот подтерм расположен внутри T. Тогда в теореме используется функциональная запись $P(x, t)$, где P - вспомогательная переменная, в соче-

тании с указателем "вхождение(P)". Здесь переменная x заранее идентифицирована с переменной либо со списком переменных. Теоремный терм $P(x,t)$ идентифицируется с термом задачи T, а t - с некоторым вхождением v в T. В заменяющих фрагментах теоремы запись $P(A,B)$ далее используется для обозначения результата замены выделенного внутри T вхождения v на B, с одновременной подстановкой вместо всех остальных (не расположенных внутри v) вхождений переменной либо переменных x терма либо термов A. Чтобы в этой ситуации сослаться на результат подстановки терма A вместо x в конъюнкцию всех утверждений из области вхождения v, используется запись $P(A)$. Если в теореме не используется нетождественная подстановка A вместо x (а такие невырожденные подстановки появляются обычно в приемах вывода теорем), то вместо $P(x,t)$ может быть использована запись $P(t)$.

10. Указатель "функподст(f g)" позволяет идентифицировать теоремные термы вида $f(g(A),B)$ либо $f(g(A))$. Предварительно переменная g должна быть идентифицирована с некоторой переменной G. Проверяется, что идентифицирующий терм T содержит переменную G только в виде "значение($G\ldots$)". После идентификации формирование заменяющих термов $f(C,B)$ либо $f(C)$ происходит путем замены всех вхождений выражений "значение($G\ldots$)" в T на C. Данная идентификация применяется в приемах вывода теорем.

11. Если нужно идентифицировать терм вида $f(A,B)$ с термом вида $F(X,Y)$ либо $G(F(X,Y))$, где F,G - логические символы операций либо отношений, то применяется указатель "Бинарнаяоперация(f)". A,B идентифицируются с X,Y без учета порядка этих операндов. Переменная f, рассматриваемая отдельно, считается идентифицированной с символом F.

12. Если нужно идентифицировать терм вида $f(A_1\ldots A_n)$, у которого f - некоторая операция либо отношение, причем лишь единственный операнд A_i содержит заданную переменную x, то в теореме приема можно использовать запись $F(A)$, сопровождаемую указателем "внешфункция(F,X)". Здесь переменная F идентифицируется с логическим символом f, переменная X заранее должна быть идентифицирована с x, а выражение A будет идентифицироваться с тем операндом, который содержал x.Соответственно, термы вида $F(B)$ будут формироваться как результаты замены найденного операнда A_i на B.

Идентификация операций над конечными семействами и кванторных импликаций для конечных областей

1. Пусть одноместная операция F над семейством объектов возникла из двуместной ассоциативно-коммутативной операции f (например, "суммавсех" либо "произведениевсех"из двуместных сложения и умножения). Выражение F(отображение(i и(целое(i) меньшеилиравно(1 i)меньшеилиравно(i n))$P(i)$)), определяющее результат применения данной операции к конечному семейству (формульный редактор прорисовывает такое выражение, например, в случае суммирования, как

$$\sum_{i=1}^{n} P(i)),$$

можно идентифицировать с термом задачи вида $f(A_1\ldots A_n)$ (в случае суммирования - с суммой $A_1+\ldots+A_n$). Для этого используется указатель "развертка(K

)", где K - указатель вхождения в теорему данного выражения $F(\ldots)$. Предпринимается последовательная идентификация $P(i)$ с $A_1, \ldots, A_n$. При этом еще не идентифицированные переменные g, встречающиеся внутри терма $P(i)$ в виде $g(i)$, оказываются идентифицированы с термами "набор($B_1 \ldots B_n$)"; здесь B_j - терм, с которым была идентифицирована $g(i)$ при рассмотрении A_j. Несколько различных указателей "развертка(K_1)", ..., "развертка(K_m)" можно объединять в общий указатель "развертка($K_1 \ldots K_m$)". Чтобы при идентификации допустить вырожденный случай $n = 0$, используется дополнительный указатель "пустоеслово(K)". В качестве примера рассмотрим теорему

$$\forall_i(i \in \{1, \ldots, n\} \rightarrow \text{конечное}(B(i))) \;\&\; A = \bigcup_{i=1}^{n} B(i) \rightarrow \text{конечное}(A),$$

применяемую в проверочном операторе усмотрения конечного множества. Указатель "развертка(фикс(2 2))" позволяет идентифицировать второй антецедент с произвольным равенством вида $A = B_1 \cup \ldots \cup B_n$; при этом одновременно идентифицируются число операндов n и набор B.

2. Указатель "развертка(K)" можно использовать не только для уточнения режима идентификации, но и для того, чтобы новый терм согласно выражению F(отображение(...)) формировался как $f(A_1 \ldots A_n)$. Допускается объединение таких указателей формирования новых термов с указателями идентификации в общую запись "развертка(...)". Примером может служить теорема

$$\text{периметр}(\text{фигура}(x)) = \sum_{i=1}^{n} l(x(i)x(i(\text{mod}n) + 1)),$$

используемая для перехода к выражению периметра многоугольника, заданного набором x своих вершин, в виде сумм длин его сторон. Здесь используется указатель "развертка(фикс(0 2))", обеспечивающий формирование "конкретной" суммы длин.

3. Следующий способ применения указателя "развертка(K)" - для идентификации терма "отображение(i принадлежит(i номера(1 n)) $P(i)$)" с термом "набор(A_1 ... A_n)", определяющим конечный набор, либо для формирования второго терма по первому после идентификации из других источников всех его переменных. Заметим, что формульный редактор прорисовывает терм "отображение(...)" в виде $\lambda_i(P(i), i \in \{1, \ldots, n\})$.

4. Особо выделен случай двуместной коммутативно - ассоциативной связки "и" - аналогом этой связки для семейства утверждений служит кванторная импликация "длялюбого(i если принадлежит(i номера(1 n)) то $P(i)$)". Формульный редактор прорисовывает ее в виде $\forall_i(i \in \{1, \ldots, n\} \rightarrow P(i))$. Если такая импликация представляет собой антецедент теоремы приема, то указатель идентификации "развертка(K)" определяет режим поиска в контексте идентификации группы утверждений $A_1, \ldots, A_n$, идентифицируемых последовательно с $P(i)$. Функциональные переменные $g(i)$ из $P(i)$ при этом идентифицируются так же, как в случае термов F(отображение(...)). Указатель "развертка(K)" может ссылаться и на определяющую новый терм кванторную импликацию приведенного выше вида, которая в этом случае порождает конъюнкцию $P(1)$ $\&$ ... $\&$ $P(n)$. Наконец, рассматриваемая кванторная импликация, являющаяся

антецедентом теоремы, может быть выделена, кроме указателя "развертка(K)", также некоторым указателем обработки антецедента (см. следующий раздел), определяющим режим проверки этого антецедента при помощи пакетного оператора либо вспомогательной задачи. В этом случае к моменту проверки должны быть идентифицированы все переменные антецедента, и проверка будет заключаться в последовательности обращений к указанным оператору либо задаче для $i = 1, \ldots, n$.

5. Возможна идентификация многочлена, определенного с помощью выражения "суммавсех(отображение(i и(целое(i)меньшеилиравно(1 i)меньшеилиравно(i n))умножение(значение(A i)степень(x плюс(i минус(1))))))". Это выражение прорисовывается формульным редактором как

$$\sum_{i=1}^{n} A(i)x^{i-1}.$$

Здесь используется указатель идентификации "Многочлен(K)", где K - указатель вхождения в теорему рассматриваемого выражения. Переменная A при этом идентифицируется с термом "набор($C_1 \ldots C_n$)", задающим коэффициенты многочлена C_i, упорядоченные по возрастанию степеней; переменная n - с увеличенной на единицу степенью многочлена (термом). Все переменные выражения x должны быть определены до идентификации данного многочлена.

Идентификация матриц

Матрицы задаются термами "строки(...)" либо "столбцы(...)", перечисляющими элементы матрицы, соответственно, по строкам либо по столбцам. Чтобы идентифицировать с такой матрицей выражение $\lambda_{ij}(A(i,j), i \in \{1, \ldots, m\}\ \&\ j \in \{1, \ldots, n\})$, используется указатель "матрица(S)", где S - указатель вхождения данного выражения в теорему. Такой же указатель применяется, если на основе приведенного выше выражения нужно создать новый терм, обозначающий матрицу. Допускается объединение нескольких указателей "матрица(...)" в один, перечисляющий сразу все рассматриваемые указатели вхождений в теорему S.

Если нужно идентифицировать матрицу из коэффициентов выражений заданного вида, встречающихся, например, в некоторой сумме, то для идентификации с суммой (или подобной многоместной операцией) используется выражение вида $F(\lambda_j(g(A(i,j), B), P(j)))$, сопровождаемое указателем "развертка(...)", где F - обобщение двуместной ассоциативно-коммутативной операции f (аналога сложения) на наборы произвольной длины. Добавление указателя "коэфф(A)"обеспечивает идентфикацию A как матрицы из коэффициентов $A(i,j)$. При отсутствии в сумме некоторых членов $g(\ldots)$ значение $A(i,j)$ идентифицируется с нулем операции f.

Дополнительная идентификация

Некоторые из теоремных переменных, а также некоторые переменные, встречающиеся в фильтрах приема, могут быть идентифицированы за счет "внешних" по отношению к тексту теоремы средств. Для такой дополнительной идентификации используется указатель "контекст($A_1 \ldots A_n$)", устроенный совершенно так же, как одноименный фильтр. Единственным отличием от фильтра здесь является то, что переменные,

значения которых определяются в $A_1, \ldots, A_n$ посредством идентифицирующих термов, имеют своей областью действия не только данный указатель "контекст(...)", как это было в случае фильтра, а все описание приема. Эти переменные, в частности, могут встречаться и в тексте теоремы, где они рассматриваются как уже идентифицированные. Такая переменная может отсутствовать в идентифицирующей части теоремы, но входить в ту ее часть, на основе которой формируются новые термы. В качестве простого примера рассмотрим теорему $\neg(i - j = 0) \to \text{card}(\text{set}_x(l(x) = 2\ \&\ \text{слово}(x)\ \&\ x(i) = a\ \&\ P(x(j)))) = \text{card}(\text{set}_x(P(x)))$, используемую для сведения числа пар x, у которых одна компонента фиксирована, а другая удовлетворяет некоторому условию P, к числу объектов, удовлетворяющих условию P. Здесь выражение $x(j)$ при идентификации $P(x(j))$ должно быть уже идентифицировано. Для этого служит указатель "контекст(позиция(b теквхожд)вид(b значение(x j))не(равно(i j)))", который усматривает вхождение данного выражения внутри заменяемого терма, определяя таким образом значение j. Будет ли переменная x идентифицирована по данному указателю, или непосредственно из теоремы (где она легко усматривается из связывающей приставки в $\text{set}_x(\ldots)$) - остается на усмотрение компилятора.

Формирование вспомогательных термов при идентификации

Если к моменту идентификации некоторого теоремного терма A все его переменные уже идентифицированы, то A определяет некоторый терм B, полученный из A подстановкой вместо переменных идентифицированных с ними термов. Тогда появляется возможность не "накладывать" при идентификации терм A на соответствующий терм задачи C, а сформировать вспомогательный терм B и затем проверить совпадение термов C и B. Если подтермы терма A были снабжены указателями нормализации, то фактически C будет сравниваться не с B, а с результатом применения к B тех или иных дополнительных преобразований. Данный режим идентификации включается при наличии указателя "подтерм(K)", где K - либо указатель вхождения в теорему терма A, либо сам этот терм. В качестве примера рассмотрим теорему "таблица$\{; \lambda_i(\text{конст}(a(i), b(i)), i \in \{1, \ldots, n\})\}$+ таблица$\{; \lambda_i(\text{конст}(a(i), c(i)),$ $i \in \{1, \ldots, n\})\} =$ таблица$\{; \lambda_i(\text{конст}(a(i), b(i) + c(i)), i \in \{1, \ldots, n\})\}$". На ней основан прием сложения двух функций, заданных таблицами на одном и том же множестве точек $a(i), i = 1, \ldots, n$. Эти точки идентифицируются по первому слагаемому, а при идентификации второго слагаемого выражения $a(i)$ рассматриваются как уже известные и просто сравниваются с идентифицирующими термами. Для такой идентификации служит указатель "подтерм(фикс(0 1 2 1 1 3 1))", ссылающийся на вхождение выражения $a(i)$ во второе слагаемое.

Идентификация квазиодноместных операций с помощью вспомогательных процедур

В некоторых разделах встречаются часто повторяющиеся одноместные операции $t(x)$, возникающие на основе многоместных при подстановке вместо части их операндов констант. Например, в элементарной алгебре таковыми являются константные натуральные степени аргумента x: $x^2, x^3, x^4, \ldots$. Для идентификации таких "квазиодноместных" операций $t(A)$ могут быть использованы специальные процедуры. Эти процедуры определяются справочником "вид" по заголовку терма $t(x)$, а сам справочник используется компилятором ГЕНОЛОГа таким образом, что по умолча-

нию всегда, когда это возможно, указанные специальные процедуры применяются. В частности, при идентификации константных натуральных степеней применяется процедура "выделениестепени".

Для блокировки использования справочника "вид" при идентификации подтермов, расположенных по указателям вхождения в теорему $K_1, \ldots, K_n$, применяется указатель идентификации "вид($K_1 \ldots K_n$)".

Если необходимо передать вспомогательным процедурам, вводимым справочником "вид" для идентификации подтермов по указателям вхождений в теорему $K_1, \ldots, K_n$, некоторый комментарий, то используется указатель "выделениестепени($K_1 \ldots K_n$ $A_1 \ldots A_m$)", где набор $A_1 \ldots A_m$ определяет данный комментарий (A_1 - логический символ, являющийся заголовком комментария; возможен случай $m = 0$).

Выделение общей части нескольких термов при помощи вспомогательных процедур

Если идентифицируется группа термов $B_1, \ldots, B_n$, содержащих общую переменную x, причем каждый из них, помимо x, имеет ровно одно вхождение единственной переменной (возможно, функциональной, т.е. выражения вида $f(t)$), не встречающейся более в идентифицируемой части теоремы, то для выделения "общей части" x этих термов, а также идентификации остальных входящих в них переменных, может быть применена реализованная на ЛОСе вспомогательная процедура P. Эта процедура должна иметь формат P(x1 x2 x3), где значением входной переменной x1 служит набор термов, идентифицируемых с $B_1, \ldots, B_n$, значением выходной переменной x2 - набор термов, идентифицированных с отличными от x переменными термов $B_1, \ldots, B_n$, а значением выходной переменной x3 - терм, идентифицированный с x. Указатель идентификации, определяющий данный режим, имеет вид "общаястепень(P x $B_1 \ldots B_n$)". Примером вспомогательной процедуры указанного типа служит процедура "общаястепень(x1 x2 x3)", предназначенная для представления группы выражений в виде степеней с общим показателем x (каждое выражение B_i представляется этой процедурой в виде произведения степеней; у каждой степени определяется рациональный коэффициент показателя, и находится частное наибольших общих делителей числителей и знаменателей этих коэффициентов). Эта процедура используется, например, в приеме с теоремой

$$0 \le b \;\&\; 0 \le d \;\&\; 0 \le a \;\&\; 0 \le e \;\&\; 0 < c \to (ab^c - ed^c = 0 \leftrightarrow a^{\frac{1}{c}}b - e^{\frac{1}{c}}d = 0),$$

осуществляющем извлечение корня степени c из обеих частей равенства. Величина c определяется как общий множитель показателей степени всех неизвестных сомножителей частей этого равенства. Таких сомножителей в каждой части может быть несколько, причем каждый из них имеет вид степени. Для нахождения c вводится указатель "общаястепень(общаястепень c степень(b c) степень(d c))".

Идентификация наборов

Если в теореме встречается терм вида "префикс(A B)", то компилятор автоматически обеспечивает такой режим идентификации, при котором этот терм мог бы идентифицироваться как с выражением вида "набор(...)" (из него выбирается первый элемент для A, а остаток набора определяет B), так и с явным выражением вида "префикс(...)". Это же относится к термам вида "конкатенация(набор(A $B \ldots C$)D)"

и "суффикс(A B)". Никаких специальных указателей при этом не используется. Фактически выделение нескольких первых либо последнего элемента набора обеспечивают операторы ЛОСа: "элементнабора", "элементномер", "Элементномер", "остатокнабора" - для уточнения подробностей идентификации либо ее развития следует обращаться к программам этих операторов.

Если в указанных выше случаях термы "префикс(A B)", "конкатенация(набор(A $B \ldots C$)D)", "суффикс(A B)" выделены указателем "список(K)" (K - указатель вхождения такого терма в теорему приема), то порядок расположения элементов в наборе при идентификации игнорируется. В частности, при идентфикации терма "префикс(A B)" будут последовательно предприниматься попытки идентифицировать с A каждый из элементов набора. В качестве примера использования указателя "список" рассмотрим теорему $b \in a \to \{b;c\}\backslash a = \{;c\}\backslash a$, на которой основан прием "вычитания" множеств, заданных конечными списками. Заметим, что обозначениям формульного редактора $\{b;c\}$ и $\{;c\}$ в скобочной записи соответствуют выражения "перечень(префикс(b c))" и "перечень(c)" соответственно. Чтобы поиск элемента b, отбрасываемого из перечня ввиду его принадлежности вычитаемому множеству a, происходил по всем элементам перечня, в приеме введен указатель "список(фикс(0 1 1 1))", выделяющий вхождение подтерма "префикс(b c)".

Группировка всех слагаемых неравенства в одной части

Для идентификации неравенства вида $0 < A$ либо $A < 0$ может быть использован режим с автоматической группировкой всех слагаемых в одной части (такой же режим может быть введен и для идентификации нестрогих неравенств). Здесь используется указатель "перегруппировка($B_1 \ldots B_n$)", где $B_1, \ldots, B_n$ - указатели вхождений в теорему всех неравенств, требующих данного режима идентификации. В качестве примера рассмотрим теорему $0 < a \;\&\; c < 0 \;\&\; 0 < ax + b \;\&\; 0 < cx + d \to 0 < ad - bc$, на которой основан прием, выводящий из двух неравенств их линейную комбинацию, в которой исключена переменная x. Чтобы избежать создания приемов для различных случаев размещения x слева либо справа от знака неравенства, введен указатель "перегруппировка(фикс(3)фикс(4))".

Использование равенств из контекста идентификации

Если в теореме приема указано некоторое выражение A, которое при идентификации сравнивается с выражением B, не имеющим такого вида, то идентификация, все же, могла бы состояться, если бы в текущем контексте нашлась посылка $A = B$. Чтобы такие попытки усиления идентификации предпринимались, прием снабжается указателем "смравно(P)", где P - указатель идентифицируемого вхождения выражения A в теорему приема.

Уточнение порядка, в котором происходит идентификация

Через указатели можно в некоторой степени влиять на порядок, в котором прием будет идентифицировать значения переменных. Это может оказаться полезным для получения менее трудоемкой (с бóльшим отсечением на ранних шагах) процедуры идентификации. К числу таких указателей, уточняющих порядок идентификации, относится указатель "определено(K x)". Он блокирует идентификацию подтерма - операнда ассоциативно - коммутативной операции, расположенного по указателю

вхождения K, до тех пор, пока не будет идентифицирована переменная x. Некоторые другие указатели, от которых тоже зависит порядок идентификации, связаны с очередностью обработки антецедентов теоремы; они вводятся в следующем разделе.

10.4.2 Указатели обработки антецедентов теоремы

Антецеденты $A_1, \dots A_n$ теоремы приема $\forall_{x_1 \dots x_m}(A_1 \ \& \ \dots \ \& \ A_n \to A_0)$ предполагаются занумерованными слева направо последовательными натуральными числами $1, 2, \dots, n$. Эти номера используются в указателях обработки антецедентов для ссылок на антецеденты.

Непосредственная идентификация антецедентов

Если для обработки антецедента не предусмотрена специальная процедура, в соответствии с приводимыми далее указателями обработки, то прием пытается идентифицировать его с некоторым утверждением, содержащимся в контексте идентификации. Это называется непосредственной идентификацией антецедента.

Обычно контекст идентификации совпадает с областью вхождения в задачу точки привязки приема. Если прием был инициирован при рассмотрении вхождения в посылку, то такой контекст идентификации включает в себя все остальные посылки задачи; если прием был инициирован при рассмотрении вхождения в условие, то контекст идентификации включает в себя все остальные условия, а также все посылки задачи. Кроме того, в контекст идентификации будут входить относящиеся к области текущего вхождения подутверждения текущего терма задачи.

Можно уточнять подмножество контекста идентификации, из которого берется идентифицирующее заданный антецедент утверждение. Указатель "списокпосылок($i_1 \dots i_n$)" перечисляет номера $i_1, \dots, i_n$ тех антецедентов, которые идентифицируются с посылками задачи; указатель "списокусловий($i_1 \dots i_n$)" - номера тех антецедентов, которые идентифицируются с условиями задачи.

Проверка истинности антецедента с помощью проверочного оператора

1. Указатель обработки "блокпроверок($i_1 \dots i_n$)" указывает номера $i_1, \dots, i_n$ тех антецедентов, для проверки истинности которых будут использоваться пакетные проверочные операторы. Компилятор сам определяет заголовки этих операторов, используя справочник "легковидеть". Каждый проверочный оператор сопровождается указанием вида T утверждений, для проверки истинности которых он предназначен. При указании на обработку антецедента проверочным оператором необходимо, чтобы уже имелся оператор для проверки утверждений данного вида. Простейший пример использования проверочных операторов для обработки антецедентов - основанный на теореме $0 \le a \ \& \ 0 \le b \to (a + b = 0 \leftrightarrow a = 0 \ \& \ b = 0)$ прием декомпозиции уравнения с нулем в правой части и левой частью, представимой в виде суммы двух слагаемых одного знака. Для быстрого усмотрения неотрицательности слагаемого a и суммы остальных слагаемых b используется проверочный оператор "усмменьшеилиравно". Чтобы компилятор создал обращения к нему, вводится указатель "блокпроверок(1 2)".

2. Для некоторых типов утверждений предусмотрена усиленная версия проверочного оператора. Заголовок такого усиленного проверочного оператора опреде-

ляется при помощи справочника "проверка", аналогичного справочнику "легковидеть". Для ссылки на антецеденты, обрабатываемые усиленными проверочными операторами, используется указатель "проверка($i_1 \dots i_n$)"; $i_1, \dots, i_n$ - номера этих антецедентов.

3. В ряде случаев бывает целесообразно в зависимости от контекста корректировать заголовок антецедента, указанный в теореме. Например, вместо проверки строгого неравенства может оказаться достаточной проверка нестрогого неравенства. Чтобы прием мог выбирать необходимый проверочный оператор для такого антецедента (в указанном случае - оператор "усмменьше" либо "усмменьшеилиравно"), используется указатель "замещение($P\ i\ A$)", который означает, что при выполнении условия P заголовок антецедента с номером i, обрабатываемого проверочным оператором, заменяется на A.

4. Иногда в качестве антецедента теоремы используется переменная (обычная либо функциональная, вида $f(x)$), которая идентифицируется из других фрагментов теоремы с некоторым утверждением. Организовать непосредственную проверку истинности такого утверждения проверочным оператором невозможно, так как априори вид его не известен. Однако, здесь можно воспользоваться оператором "очевидно(...)", который с помощью справочника "легковидеть" пытается определить по виду предъявленного ему для проверки утверждения необходимый проверочный оператор и переадресует ему дальнейшую проверку. Чтобы предпринять такую обработку для антецедентов с номерами $i_1, \dots, i_n$, применяется указатель "очевидно($i_1 \dots i_n$)". В качестве примера рассмотрим теорему "семействомножеств($\lambda_a(f(a), g(a))$) & $g(x) \rightarrow \mathrm{set}(f(x))$)", на которой основан прием, усматривающий множество в элементе семейства множеств. Первый антецедент идентифицируется непосредственно; для обработки второго введен указатель "очевидно(2)".

Реализация антецедента с помощью пакетного синтезатора

Если группа антецедентов с номерами $i_1, \dots, i_n$ должна обрабатываться одним общим пакетным синтезатором, то используется указатель обработки "значения($i_1 \dots i_n$)". Компилятор определяет заголовок применяемого синтезатора с помощью справочника "значения". Как и в случае проверочного оператора, синтезатор предназначен для обработки утверждений лишь специального вида, причем обработка заключается в нахождении значений выходных переменных, при которых эти утверждения становятся истинными. Вид списка $T_1, \dots, T_n$ обрабатываемых утверждений определяется в описании формата синтезатора (см. справочник "синтезатор"). Среди переменных утверждений $T_1, \dots, T_n$ выделены входные - они должны быть уже идентифицированы к моменту обработки указанных антецедентов, а также выходные - их значения будут определяться в результате применения синтезатора. Заметим, что на "выходных" позициях антецедентов, обрабатываемых синтезатором, могут стоять только обычные переменные (недопустимы даже функциональные переменные - выражения вида $f(x)$). В то время, как ссылки на антецеденты, обрабатываемые проверочными операторами, объединяются в общий указатель "блокпроверок(...)", каждое обращение к синтезатору требует своего указателя "значения(...)". В качестве примера обработки антецедентов синтезаторами рассмотрим прием, основанный на теореме $b \leq a\ \&\ a \leq c\ \&\ d = [b]\ \&\ 0 < d - c + 1 \rightarrow [a] = d$. Этот прием предпринимает попытку определения значения целой части путем получения верхней и нижней

численных оценок b, c выражения a. Собственно нахождение этих оценок в нем выполняется синтезаторами "нижняяоценка" (первый антецедент) и верхняяоценка" (второй антецедент). Шаблон обращения к первому из них имеет вид $b \leq a$, где a - входная переменная, b - выходная; шаблон обращения ко второму имеет вид $a \leq c$, где a - входная переменная, c - выходная. Компилятор сам определяет, какой из синтезаторов для какого антецедента нужно использовать - идентифицированной на момент рассмотрения этих антецедентов является только переменная a, которая и рассматривается как входная. Указатели обработки первых двух антецедентов в данном приеме имеют вид "значения(1)", "значения(2)".

Идентифицирующий антецедент

Некоторые антецеденты теоремы приема могут вводить вспомогательный терм T и затем идентифицировать его с заданным термом B. Они называются идентифицирующими антецедентами. Самый распространенный вид идентифицирующего антецедента - "$A = B$". В этом случае терм T определяется, с учетом указателей нормализации, заданных в приеме, по терму A. К моменту обработки антецедента все переменные терма A должны быть уже идентифицированы. Расположение терма A в левой либо правой части равенства несущественно. Если терм B имеет хотя бы одну не идентифицированную переменную, то он будет идентифицироваться с термом T так же, как, например, заменяемая часть какого - либо тождества идентифицируется с подтермом текущего терма задачи. В простейшем случае, если B является переменной, то эта переменная будет идентифицирована с T. Таким образом в теоремах приемов часто вводятся вспомогательные обозначения для громоздких либо встречающихся несколько раз выражений. Если же все переменные терма B идентифицированы, то он, как и A, определяет некоторый терм T'. Тогда обработка антецедента сводится к сравнению T и T'.

Идентифицирующий антецедент может также иметь вид дизъюнкции нескольких равенств - $A_1 = B_1 \vee \ldots \vee A_n = B_n$. Здесь предпринимаются альтернативные попытки идентификации термов $B_1, \ldots, B_n$ с термами, определенными по $A_1, \ldots, A_n$. Как и выше, порядок размещения частей равенств несущественен. Наконец, возможно применение идентифицирующего антецедента "принадлежит(x номера(m k))", который последовательно идентифицирует переменную x с десятичными записями чисел $m, m+1, \ldots, k$. Такой антецедент применяется только в случаях, когда переменные m, k идентифицируются с десятичными записями целых чисел. Для выделения идентифицирующих антецедентов используется указатель обработки "идентификатор(i_1 $\ldots$ i_n)", перечисляющий все их номера $i_1, \ldots, i_n$. В качестве примера рассмотрим теорему для нахождения корней квадратного уравнения:

$d = b^2 + 4ac \rightarrow ax^2 + bx = c \leftrightarrow \neg(a = 0)$ & $0 \leq d$ & $(x = \frac{\sqrt{d}-b}{2a} \vee x = -\frac{\sqrt{d}+b}{2a}) \vee bx = c$ & $a = 0$.

Ее единственный антецедент вводит обозначение d для дискриминанта, которое далее используется в теореме несколько раз. В действительности, с учетом нормализаторов приема, в данном антецеденте происходит и вычисление дискриминанта - упрощение его и попытка разложения на множители. Указатель обработки антецедента здесь имеет вид "идентификатор(1)".

Если идентифицирующий антецедент $A = B$ применяет к сформированному по терму A утверждению T нормализаторы либо вспомогательные задачи на описание, то это утверждение может принять вид дизъюнкции нескольких подутверждений - "подслучаев". Чтобы рассматривать такие подслучаи по отдельности, вводится спе-

циальный режим идентификации терма B не со всей дизъюнкцией T, а с отдельными ее членами. Такой режим допустим для приема, выполняющего эквивалентную замену, причем итоговый заменяющий терм будет формироваться как дизъюнкция тех вариантов "теоремного" заменяющего терма, которые соответствуют отдельным подслучаям. Данный режим может быть применен сразу к нескольким перечисляющим подслучаи идентифицирующим антецедентам с номерами $i_1, \ldots, i_n$. Соответствующий указатель обработки антецедентов имеет вид "дизъюнктчлен($i_1 \ldots i_n$)". Он вводится в дополнение к указателю "идентификатор(...)", также выделяющему эти антецеденты. Примеры его использования можно найти в подразделе базы приемов, посвященном дифференциальным уравнениям. Идентифицирующий антецедент применяется для обращения к задаче на описание, решающей некоторое вспомогательное дифференциальное уравнение; по различным подслучаям решения этого вспомогательного уравнения определяются решения основного уравнения, и далее они объединяются в общую дизъюнкцию.

Иногда бывает необходимо формировать некоторый вспомогательный терм A, используемый приемом, в два этапа: сначала ввести новую переменную x и преобразовать с помощью нормализаторов либо вспомогательных задач терм $t(x)$ к требуемому виду $r(x)$ (например, разложить на множители), а затем для получения A подставить в $r(x)$ вместо x заданное выражение P. Чтобы обеспечить такой режим, вводятся дополнительный антецедент $x = P$ и указатель "свертка(x F)". Антецедент $x = P$ снабжается при этом стандартным указателем "идентификатор(i)". Переменная x будет идентифицироваться как новая переменная, причем нормализаторы, обрабатывающие термы с x, снабжаются дополнительными посылками - конъюнктивными членами утверждения F. По окончании применения таких нормализаторов в их результирующий терм вместо x будет подставляться P. Утверждение F может отсутствовать. Примером использования указателя "свертка" может служить теорема $h = bd \;\&\; i = bf \;\&\; g = \sqrt{b}e \;\&\; j = ag^3 + cg^2 + hg + i \rightarrow a\sqrt{b}e^3 + ce^2 + d\sqrt{b}e + f = j/b$, позволяющая выполнять разложение на множители выражения $a\sqrt{b}e^3 + ce^2 + d\sqrt{b}e + f$ сведением к разложению на множители многочлена $ag^3 + cg^2 + hg + i$. Новая переменная g этого многочлена введена идентифицирующим антецедентом $g = \sqrt{b}e$. Если бы вместо переменной здесь было сразу подставлено обозначаемое ею выражение, то при преобразованиях вид многочлена оказался бы утерян, и необходимые приемы не сработали бы. Прием использует указатель "свертка(g число(g))".

Программно реализуемый антецедент

Если переменные антецедента были идентифицированы с термами "константного" характера, обозначающими объекты, для представления которых предусмотрены нелогические структуры данных (например, десятичные числа), то проверка либо реализация такого антецедента могут выполняться процедурами, обрабатывающими не сами константные термы, а обозначаемые ими объекты, заданные нелогическими структурами данных. Такие антецеденты называем программно реализуемыми. Указатель, выделяющий программно реализуемые антецеденты, имеет вид "программа($i_1 \ldots i_n$)", где $i_1, \ldots, i_n$ - все их номера. В программно реализуемых антецедентах используются выражения и утверждения, для которых компилятор ГЕНОЛОГа может формировать процедуры непосредственного вычисления. К их числу, в частности, относятся выражения, построенные при помощи числовых констант, операций сложения, изменения знака, умножения, возведения в константную натуральную степень, модуля, взятия максимума либо минимума двух чисел, наибольшего общего

делителя и наименьшего общего кратного.

Приведем наиболее часто встречающиеся типы программно реализуемых антецедентов:

1. Антецедент для деления целого числа m на целое ненулевое число n с остатком: $m = an + b$. Остаток здесь берется от 0 до $|n| - 1$.

2. Антецедент, перечисляющий все целочисленные делители m, k целого числа n: $n = mk$.

3. Антецедент, перечисляющий все представления целого числа n в виде произведения целого m на заданную натуральную степень p целого k: $n = mk^p$.

4. Антецедент для проверки числового равенства либо присвоения: $a = b$.

5. Антецедент для проверки неравенства $a < b$ либо $a \leq b$.

6. Отрицание, дизъюнкция либо конъюнкция нескольких программно реализуемых антецедентов проверочного типа также является программно реализуемым антецедентом.

7. Антецедент для нахождения целочисленных решений m квадратного уравнения с целыми коэффициентами a, b, c: $am^2 + bm = c$.

8. Антецедент для перечисления всех целых чисел i из промежутка от m до n: $i \in \{m, \ldots, n\}$.

9. Антецедент для выделения наибольшей натуральной степени n целого числа a, на которую делится целое число b; определяется также частное c от деления b на эту натуральную степень: $b = a^n c$.

10. Антецедент для перечисления всех простых либо целочисленных делителей m целого n (если в фильтрах указано "простое(m)", то перечисляются простые делители, иначе - целочисленные): $m|n$.

11. Антецедент для проверки четности целого числа n: $n - even$.

В качестве примера рассмотрим теорему "$x -$ целое & $y -$ целое & $|a| \leq |b|$ & $b = ap + q \rightarrow ax + by = c \leftrightarrow \exists_z(x = z - py$ & $z -$ целое & $az + qy = c)$", используемую для решения линейного целочисленного уравнения с двумя неизвестными. Она соответствует одному шагу алгоритма Евклида. В фильтрах приема указано, что a, b, c идентифицируются с целочисленными константами, и это позволяет реализовать третий и четвертый антецеденты с помощью операторов ЛОСа, применяемых к целым числам (для четвертого антецедента выполняется деление b на a с остатком). Указатели обработки антецедентов приема - "блокпроверок(1 2)" и "программа(3 4)".

Вместо указателя "программа(...)", программно реализуемый антецедент вида $x = t$ может быть выделен указателем "выч(...)". Это происходит в тех случаях, когда все переменные выражения t оказываются идентифицированы с константными термами, причем переменную x нужно идентифицировать с численной константой,

полученной при приближенном вычислении значения t с помощью машинной арифметики для чисел формата "с плавающей запятой". Например, так делается в приемах анализатора "чертеж", позволяющего строить эскизы к планиметрическим задачам.

Список типов программно реализуемых антецедентов постоянно пополняется. Основные типы задаются приемами справочника "вычисл". Чтобы получить информацию о программной реализации антецедента, содержащего некоторый ключевой логический символ, находим в оглавлении приемов раздел данного символа, входим в его подраздел "Справочники", и далее - в подраздел "Вычисл". Анализируя приемы справочника "вычисл", определяем, какие типы программно реализуемых антецедентов существуют и на какие форматы данных они рассчитаны. Теорема справочника "вычисл" имеет вид "вычисл($s\ T\ A_1 \ldots A_n\ R$)", где s - ключевой логический символ; T - шаблон антецедента, содержащий символ s; $A_1, \ldots, A_n$ - указатели форматов входных и выходных переменных; R - оператор ЛОСа, вставляемый компилятором ГЕНОЛОГа в программу приема для обработки антецедента. Точно таким же образом справочник "вычисл" определяет компиляцию подвыражений программно реализуемых антецедентов. В этом случае некоторое A_i имеет вид "выход(f)"; f - формат значения выражения T.

Отдельный программно реализуемый антецедент приема сканирования задачи может представлять собой обращение к вычислительному пакету, реализованному на ГЕНОЛОГе и предназначенному для выполнения каких-либо трудоемких вычислений вне логического процесса. Таким образом обеспечивается подключение к логическому программированию "обычных" вычислений.

Проверка истинности антецедента при помощи решения задачи на доказательство

Для проверки истинности антецедента с помощью задачи на доказательство могут применяться различные указатели, уточняющие уровень используемых для нее средств. Эта проверка предпринимается лишь в исключительных случаях, так как она обычно гораздо более трудоемка, чем обращение к проверочному оператору. Решение вспомогательной задачи на доказательство может вызвать неоправданно большую паузу в решении основной задачи - если нет веских оснований считать, что проверяемое условие вообще выполнено. Поэтому рекомендуется сопровождать обращения к таким задачам ограничителями трудоемкости (см. указатель "Обрыв(...)").

1. Если максимальный уровень используемой задачи на доказателство равен 4, причем она снабжается комментарием "извлекается" и решается с заблокированным сканированием посылок (их веса подняты до величины, превосходящей максимальный уровень), то применяется указатель "легковидеть(i)"; i - номер антецедента (этот и другие приводимые ниже однотипные указатели U обработки обычно группируются в указатель $U(i_1 \ldots i_m)$, ссылающийся сразу на несколько антецедентов).

2. Если максимальный уровень задачи на доказательство равен 5, то применяется указатель "усматривается(i)".

3. Наконец, наиболее сильное средство проверки - использование задачи на доказательство, максимальный уровень которой полагается совпадающим с максимальным уровнем текущей задачи. Здесь используется указатель обработки

"следствие(i)". Простейший пример - прием, основанный на теореме $0 < a-b \to \neg(a = b)$ и используемый для попытки усмотреть неразрешимость уравнения $a = b$ в условии задачи на описание путем доказательства неравенства. Этот прием применяется на достаточно большом уровне, когда решение задачи обычными средствами уже зашло в тупик. Первый антецедент в нем выделен указателем "следствие(1)", причем трудоемкость попытки доказательства неравенства (может быть, придется еще проверить и противоположное неравенство) ограничена сверху с помощью указателя "лимит(3000000)".

4. Особо выделен случай, когда решается задача на доказательство, причем она фактически сводится к проверке истинности заданного антецедента. Тогда вспомогательная задача формируется при помощи оператора "спуск" (специально предназначенного для формирования задач на описание либо доказательство, к которым решаемая задача может быть сведена путем замены некоторого условия). Максимальный уровень ее полагается равным максимальному уровню текущей задачи на доказательство. Данный режим проверки вводится при помощи указателя "доказать(i)". Этот же указатель можно использовать в пакетных операторах, где он имеет несколько иной смысл - просто указывает на обращение к вспомогательной задаче на доказательство с максимальным уровнем обращения, равным 8. В качестве примера рассмотрим теорему

$$\forall_x(g(x) \to 0 < f(x)) \to 0 < \prod_{x,g(x)} f(x),$$

на которой основан прием, предпринимающий попытку доказательства положительности конечного произведения путем установления положительности его общего члена. Первый антецедент, выделенный указателем "доказать(1)", обрабатывается путем решения задачи на доказательство, полученной из текущей задачи на доказательство путем соответствующей замены условия. Заметим, что сама текущая задача при этом не изменяется, пока не будет решена вспомогательная - лишь после этого условие основной задачи будет заменено на константу "истина".

Антецеденты, реализуемые путем обращения к вспомогательной задаче на описание

Применение приема вывода следствий в задаче на исследование может быть связано с необходимостью предварительно вычислить значения группы вспомогательных выражений. Для такого вычисления обычно используется задача на описание, имеющая цель "известно ...", перечисляющую те известные параметры, через которые должны быть определены вычисляемые выражения. Чтобы создать данную задачу на описание и обратиться к ее решению, в теореме приема вводятся антецеденты $x_1 = t_1, \ldots, x_n = t_n$, перечисляющие подлежащие вычислению выражения $t_1, \ldots, t_n$ и те переменные $x_1, \ldots, x_n$, которым присваиваются вычисленные значения (т.е. выражения, не содержащие неизвестных контекста применения приема). Эти антецеденты сопровождаются указателем "вычисл(A B)". Здесь A - список их номеров; B - список дополнений: "посылки(P)" - указание конъюнкции P дополнительных посылок, используемых в задаче; "удалениепосылок(Q)" - Q есть список названий таких разделов, что содержащие их понятия посылки текущей задачи не переносятся во вспомогательную задачу на описание. Известные параметры вспомогательной задачи берутся из списка известных параметров текущей задачи на исследование.

Антецеденты, реализуемые идентифицирующими операторами

В некоторых предметных областях приходится идентифицировать антецеденты, для которых контекст идентификации не содержит явных прототипов, однако такие прототипы могут быть извлечены из утверждений данного контекста с помощью крайне небольшого числа простых логических переходов. Высокая частота попыток такой идентификации и расположение их в самом начале программ приемов делают нежелательным применение здесь обычных проверочных операторов и синтезаторов. Для ускорения идентификации в таких ситуациях применяются так называемые идентифицирующие операторы - специальные реализованные непосредственно на ЛОСе вспомогательные процедуры, выполняющие указанные выше логические переходы.

Например, для перечисления всех точек, выделенных на текущий момент на некоторой прямой, применяется идентифицирующий оператор "точкипрямой", который усматривает точки из различных источников: точки из названия самой прямой; точки из явно содержащихся в посылках условий принадлежнотси этой прямой; точки из условий принадлежности интервалам и отрезкам, концы которых принадлежат данной прямой.

Ускоренное, по сравнению с пакетными операторами, функционирование идентифицирующих операторов достигается не только за счет малого числа и простоты применямых в них логических переходов, но и за счет использования специальных адресных структур данных, группирующих посылки специального вида, необходимые для выполнения таких переходов. Эти структуры данных технически оформлены в виде комментариев к посылкам "списокпосылок ...", "выражение ..."и "отр ..." (подробнее о них см. информацию о логических символах - заголовках этих комментариев). Они автоматически корректируются в процессе преобразований списка посылок - эту работу выполняют процедуры, осуществляющие преобразования списка посылок, т.е. "вывод", "заменавхождения", и т.п.

Применение специальных адресных структур для быстрого поиска нужных посылок сближает по своей эффективности применяемое логическое представление с сетевым, где непосредственно указывались бы объекты, находящиеся в заданном отношении с рассматриваемым объектом. Такое сближение усиливается использованием буфера, в котором сохраняются результаты нескольких десятков последних обращений к идентифицирующему оператору (см. комментарий к посылкам "Буфер"). Фактически этот буфер становится в некотором смысле самоорганизующимся "расширенным контекстом", содержащим сотни готовых ответов на типовые вопросы, относящиеся к текущей ситуации. Во избежание ошибок управляющего характера, буферы идентифицирующих операторов сбрасываются после каждого преобразования посылок, которое могло бы изменить результат обращения.

Указатель обработки антецедентов с номерами $i_1, \ldots, i_n$ посредством идентифицирующих операторов имеет вид "усм($i_1 \ldots i_n$)".

Ввод нового идентифицирующего оператора осуществляется следующим образом:

1. Определяется, какого вида группа антецедентов $T_1, \ldots, T_n$ будет обрабатываться при помощи нового оператора. В этой группе антецедентов выделяются входные термы $A_1, \ldots, A_m$ (эти термы должны быть к началу идентификации уже определены - либо как выходные значения ранее примененных идентифицирующих операторов, либо через идентифицированные значения их переменных). Выделяются также выходные термы $B_1, \ldots, B_k$, которые будут определяться идентифицирующим оператором. Здесь допустим случай $k = 0$.

2. На ЛОСе создается программа идентифицирующего оператора $f(x_1 \dots x_{m+k+1})$, входные данные которой суть: x_1 - текущая задача на исследование либо на доказательство, при сканировании которой применяется идентифицирующий оператор; $x_2, \dots x_{m+1}$ - термы $A_1, \dots, A_m$. Выходные переменные $x_{m+2}, \dots,$ x_{m+k+1} перечисляют наборы термов $B_1, \dots, B_k$ (в случае $k = 0$ оператор выполняет проверку истинности соответствующих $T_1, \dots, T_n$).

3. Создаются приемы справочников "усм" и "См", основанные на псевдотеореме вида "усм(f вход($A_1 \dots A_m$) выход($B_1 \dots B_k$) усм(T_1 & ... & T_n) $C_1 \dots C_p$)". Здесь $C_1, \dots, C_p$ - информационные элементы, используемые компилятором ГЕНОЛОГа при формировании обращений к оператору f. Вид этих элементов приведен в информации о логическом символе "усм". Отметим, в частности, элемент "уровень(U)", который определяет уровень приоритетности при применении компилятором данного идентифицирующего оператора (чем меньше U, тем выше приоритет).

После этого компилятор ГЕНОЛОГа готов к использованию нового идентифицирующего оператора.

При вводе комплекта новых идентифицирующих операторов следует учитывать, что идентификация одних и тех же утверждений в разных приемах может происходить в различном порядке - с различными комбинациями определенных к моменту идентификации подтермов. Отсутствие идентифицирующего оператора, обслуживающего какую-либо из таких комбинаций, может ухудшить качество компиляции или вообще вызвать отказ компилятора.

Основной раздел, в котором были использованы идентифицирующие операторы - элементарная геометрия. Перечислим логические символы, для которых в их подразделах оглавления базы приемов указаны справочники "усм" и "См", а также созданные для этих символов идентифицирующие операторы:

1. "Прямая". Введены операторы: "прямыеточки" (перечисление прямых, проходящих через данную точку); "точкипрямой" (перечисление точек прямой); "точкапрямой" (проверка принадлежности точки прямой); "общаяточка" (определение общих точек двух прямых); "общаяпрямая" (определение прямых, проходящих через две точки); "лежатнапрямой" (определение прямых, проходящих через три заданные точки - так как нет гарантий, что эти точки действительно различны или что не совпадают две по-разному обозначенные прямые, он выдает результаты в режиме перечисления); "точкилуча" (перечисление точек луча, определяемого по двум точкам); "точкалуча" (проверка принадлежности третьей точки лучу, определяемому двумя первыми точками); "пооднусторону"; "поразныестороны" (перечисление точек, лежащих на плоскости, соответственно, по одну либо по разные стороны от данной точки относительно данной прямой); "Однасторона"; "Разныестороны" (проверка расположения двух точек по одну либо по разные стороны относительно данной прямой). Заметим, что последние два идентифицирующих оператора в итоге оказались практически неиспользуемыми, так как их вытеснили хотя и более медленные, но на порядок более мощные проверочные операторы "разныестороны" и "однасторона".

2. "Плоскость". Введены операторы: "точкиплоскости" (перечисление точек плоскости); "точкаплоскости" (проверка принадлежности точки плоскости); "плос-

коститочки" (перечисление плоскостей, проходящих через заданную точку); "общаяплоскость" (проверка принадлежности двух прямых одной плоскости).

3. "Отрезок". Введены операторы: "точкиотрезка" (перечисление точек отрезка); "точкаотрезка" (проверка принадлежности точки отрезку); "конецотрезка" (определение противоположного конца отрезка, если заданы точка отрезка и первый конец). Заметим, что идентифицирующие операторы для работы с отрезком сами по себе весьма слабые; чтобы эффективно определять взаимное расположение точек на прямой с помощью этих операторов, необходимо заблаговременно в явном виде выводить следствия - утверждения о принадлежности или непринадлежности точек отрезкам и интервалам.

4. "Расстояние". Введены операторы "расстояния" (перечисление точек, для которых введено в рассмотрение расстояние до данной точки); "смрасстояние" (проверка того, что расстояние между двумя точками введено в рассмотрение); "равныедлины" (перечисление пар точек C, D, таких, что расстояние от C до заданной точки A равно расстоянию от D до заданной точки B); "равноудалена" (перечисление пар точек B, C и прямых BC, таких, что расстояние от B до заданной точки A равно расстоянию от B до C, причем точка A лежит на прямой BC.

5. "Угол". Введены операторы: "углывершины" (перечисление пар точек B, C, для которых введен в рассмотрение угол с заданной вершиной A); "усмугол" (проверка того, что заданный угол ABC был введен в рассмотрение); "имяугла" (по заданной вершине угла A, двум прямым, проходящим через A и двум точкам, определяющим направления лучей на этих прямых, перечисляются пары точек B, C, определяющих заданный этими лучами угол BAC).

6. "Параллельны". Введены операторы: "параллелпрямые" (перечисление прямых, параллельных данной); "усмпараллельны" (проверка параллельности двух прямых); "параллели" (перечисление пар параллельных прямых, первая их которых проходит через точку A, а вторая - через точку B); "параллелплоскости" (перечисление плоскостей, параллельных данной).

7. "Перпендикулярно". Введены операторы: "перпендикуляры" (перечисление прямых, перпендикулярных данной); "перпендикулярны" (проверка перпендикулярности двух прямых); "перпендикпрямые" (перечисление пар прямых, параллельных двум заданным перпендикулярным прямым. В этом и ряде других случаев входным данным для идентифицирующего оператора служит не терм, а вхождение в терм условия перпендикулярности некоторых прямых - см. подробнее информационный элемент "перпендикулярно" из описания формата идентифицирующих операторов); "прямыеуглы" (перечисление пар прямых, параллельных двум заданным перпендикулярным прямым, и точек их пересечения); "Перпендикулярны" (проверка перпендикулярности прямой и плоскости); "Перпендикуляры" (перечисление прямых, перпендикулярных плоскости); "Перпендикпрямые" (перечисление пар "прямая - плоскость", параллельных заданной паре перпендикулярных прямой и плоскости); "Прямыеуглы" (перечисление пар "прямая - плоскость", параллельных заданной паре перпендикулярных прямой и плоскости, а также точек их пересечения).

8. "Окружность". Введены операторы: "точкиокружности" (перечисление точек окружности); "окружноститочки" (перечисление окружностей, проходящих через заданную точку); "точкаокружности" (проверка принадлежности точки заданной окружности).

В качестве примера использования идентифицирующих операторов рассмотрим теорему $l(AB) = l(BC)$ & прямая$(BD) \perp$ прямая(AC) & $D \in$ прямая$(AC) \rightarrow l(AD) = l(DC)$. На этой теореме основан прием, усматривающий медиану в высоте BD равнобедренного треугольника ABC. Применение приема начинается с обнаружения равенства $l(AB) = l(BC)$ либо пары равенств $l(AB) = a$,$l(BC) = a$, позволяющих идентифицировать точки A, B, C. Прием имеет указатель "усм(2 3)", так что далее используется идентифицирующий оператор, перечисляющий для идентификации прямой BD прямые, перпендикулярные прямой AC. Идентифицирующий оператор проверки принадлежности точки прямой отбирает ту из них, на которой лежит точка B. Наконец, применяется идентифицирующий оператор, находящий точку D пересечения прямых BD и AC.

Уточнение списка посылок, участвующих в проверке либо реализации антецедента

1. При обработке пакетным оператором либо вспомогательной задачей антецедента с номером i могут быть введены дополнительные посылки $A_1, \ldots, A_n$. Для этого используется указатель "занесениепосылки(i и$(A_1 \ldots A_n)$)". Наиболее часто дополнительные посылки вводятся для исключения кванторного антецедента теоремы. Например, на теореме

   $\forall_y(f(y) \rightarrow \neg(h(g(y)))) \rightarrow$ непересек$(\mathrm{set}_x(\exists_y(f(y)\ \&\ x = g(y))), \mathrm{set}_z(h(z)))$

   можно было бы создать прием проверки непересечения двух множеств, заданных описателями "класс". Однако, эта теорема имеет кванторный антецедент, и вместо нее можно взять теорему

   $\neg(h(g(y))) \rightarrow$ непересек$(\mathrm{set}_x(\exists_y(f(y)\ \&\ x = g(y))), \mathrm{set}_z(h(z)))$,

   полученную сохранением лишь консеквента кванторного антецедента, однако добавить указатель "занесениепосылки(1 $f(y)$)". Результат будет тот же, что и в первом случае, но запись теоремы сокращается, и экономится срабатывание приема доказательства истинности кванторной импликации, перебрасывающего ее антецеденты в посылки задачи.

2. Если при обработке пакетным оператором либо вспомогательной задачей антецедента с номером i требуется исключить посылки, удовлетворябщие заданному условию, то применяется указатель "удалениепосылок(i X $P(X)$)". Здесь X - вспомогательная переменная, обозначающая исключаемую посылку; $P(X)$ - условие на эту посылку. Обычно удаление посылок предпринимается в тех случаях, когда есть основания предполагать наличие в контексте громоздких посылок, которые заведомо не нужно использовать при проверке утверждения рассматриваемого типа.

3. Если в приеме нормализатора обработка антецедента проверочным оператором должна производиться с использованием списка посылок этого нормализатора, пополненного утверждениями из контекста текущего вхождения, то используется указатель "облнорм".

Специальная идентификация антецедентов, имеющих своим заголовком транзитивное бинарное отношение

Для посылок, заголовком которых служит транзитивное бинарное отношение, обычно предусматривается стандартизация, сжимающая все многообразие извлекаемых по транзитивности следствий к возможно меньшему числу базисных утверждений. В этой ситуации для идентификации антецедента с одним из элементов "замыкания по транзитивности" этих базисных утверждений служат специальные указатели.

Если рассматриваемое транзитивное отношение является отношением эквивалентности E, то обычно антецедент $E(X\ Y)$ идентифицируется либо с явно содержащимся в контексте идентификации утверждением $E(A\ B)$, либо с собираемым из двух таких утверждений $E(A_1\ A_2)$, $E(A_3\ A_2)$ новым утверждением $E(A_1\ A_3)$. Последнее объясняется тем, что для всех находящихся в отношении E рассматриваемых в задаче объектов выбирается некоторый эталонный объект (в данном случае его роль играет A_2), и связь с ним других объектов задается с помощью утверждений вида $E(A\ A_2)$. Указателем на данную идентификацию служит "равно(i)", где i - номер антецедента $E(X\ Y)$. Примером ее может служить обработка антецедента $l(AB) = l(BC)$ в приводившемся выше примере с высотой равнобедренного треугольника.

Для транзитивного отношения E, не являющегося отношением эквивалентности, при идентификации может быть использован оператор "транзитпереход", перечисляющий с учетом транзитивности все термы, находящиеся в отношении E с заданным термом на основе явно присутствующих в контексте идентификации утверждений $E(A\ B)$. Идентификация антецедента $E(X\ Y)$ начинается здесь после того, как были идентифицированы все переменные одного из операндов X, Y. Данный режим вводится указателем "транзитпереход($i_1 \ldots i_n$)", где $i_1, \ldots, i_n$ - номера всех реализуемых таким образом антецедентов. Например, в основанном на теореме

$b \subseteq$ прообраз$(f, a) \rightarrow$ образ$(f, b) \subseteq a$

приеме усмотрения включения прослеживаются по цепочкам включения множества b, явно указываемые в посылках, и таким образом идентифицируется правая часть включения $b \subseteq$ прообраз(f, a).

Очередность обработки антецедентов при применении приема

Для ускорения работы программы приема иногда приходится определенным образом упорядочивать обработку антецедентов: отодвигать в конец программы априори трудоемкие вычисления, и наоборот, переносить в начало быстрые проверки, позволяющие усиливать отсечение нереализуемых ситуаций.

Чтобы приблизить к началу программы обработку антецедента с номером i, используется указатель "начало($i\ x_1 \ldots x_n$)". В этом случае обработка данного антецедента будет выполняться сразу же, как только идентифицируются значения переменных списка $x_1, \ldots, x_n$.

Указатель "конец($i_1 \ldots i_n$)" отодвигает обработку антецедентов с номерами $i_1, \ldots, i_n$ к концу программы приема.

В тех случаях, когда антецедент с номером i непосредственно идентифицируется с некоторым утверждением из контекста идентификации, можно заблокировать его обработку до того момента, пока не окажутся идентифицированы заданные переменные $x_1, \ldots, x_n$. Для этого используется указатель "подчинено($i\ x_1 \ldots x_n$)".

Слежение за появлением требуемого антецедента

Если проверка истинности антецедента при помощи пакетного оператора оказалась неудачной, то можно организовать слежение за появлением этого антецедента "в чистом виде" в контексте идентификации, с переключением внимания, обеспечивающим в этом случае повторную попытку применения приема. Здесь используется указатель обработки "См(i)", где i - номер антецедента.

Ограничение трудоемкости обработки антецедента

Для установки лимита в N шагов работы интерпретатора ЛОСа, отводимых на обработку антецедента с номером i, применяется указатель "Обрыв(i N)". Если эта обработка завершается менее чем за N шагов, то данный лимит сбрасывается, иначе - происходит откат, такой же, как при нелимитированной неудачной попытке обработки.

Отмена ограничений на использование утверждений текущего контекста для непосредственной идентификации

Если некоторые антецеденты теоремы приема были непосредственно идентифицированы с утверждениями $A_1, \ldots, A_n$ из контекста идентификации, и после этого произошло обращение к проверочному оператору либо пакетному синтезатору, то утверждения $A_1, \ldots, A_n$ не будут при работе данного оператора использоваться в непосредственной идентификации антецедентов. Они регистрируются для этого в комментарии "исключение ...", передаваемом оператору. Это делается по умолчанию для предотвращения зацикливаний в рекурсивных обращениях пакетных операторов. Однако, такое ограничение можно отменить для группы непосредственно идентифицируемых антецедентов приема пакетного оператора, имеющих номера $i_1, \ldots, i_m$. Если вводится указатель "повтор($i_1 \ldots i_m$)", то данные антецеденты будут идентифицироваться с утверждениями из текущего контекста без учета комментария "исключение ...".

Выделение антецедентов, используемых для замещения условия при обратном выводе

Ранее (см. раздел "Типы заголовков приемов") были введены приемы с заголовком "подборзначений". Они осуществляют попытку подбора примера путем замены идентифицированного с консеквентом A_0 теоремы приема $\forall_{x_1 \ldots x_m}(A_1 \ \& \ \ldots \ \& \ A_k \rightarrow A_0)$ условия задачи на описание на группу антецедентов с номерами $i_1, \ldots, i_n$, выделенных указателем "подборзначений($i_1 \ldots i_n$)". Остальные антецеденты при этом обрабатываются обычным образом. В качестве примера рассмотрим теорему $0 < a \ \& \ x = a/2 \rightarrow x \in (0, a)$, используемую для указания точки, принадлежащей интервалу $(0, a)$. Она позволяет переходить от условия $x \in (0, a)$ к условию $x = a/2$, после того, как проверено, что a положительно. Прием имеет фильтр, проверяющий, что неизвестная x не входит в другие условия (кроме, быть может, условия "число(x)"). Первый его антецедент снабжен указателем "блокпроверок(1)", второй - "подборзначений(2)".

Принятие истинности антецедента по умолчанию в зависимости от контекста

Ряд антецедентов в определенных контекстах, уточняемых фильтрами приема, могут оказаться избыточными (но не во всех случаях применения приема - иначе их можно было бы вообще отбросить). Чтоы в этих ситуациях сэкономить время на проверках или избежать отказа из-за возможной некомплектности списка посылок, имеющихся в определяемой фильтром ситуации, можно применять указатель "пассив(i A)", где i - номер антецедента, проверка которого отключается, A - фильтр, определяющий условие отключения проверки.

Унифицирующий антецедент

В приемах вывода теорем используются антецеденты вида $A = B$, обрабатываемые путем обращения к процедуре унификации. Процедура обрабатывает сразу все такие равенства, унифицируя их левые части с правыми. Равенства содержат выражения $f(t_1 \dots t_k)$, соответствующие уже идентифицированным функциональным переменным $f(x_1 \dots x_k)$. Здесь каждый содержащий еще не идентифицированные переменные терм t_i представляет собой новую переменную, и значения всех таких новых переменных определяются процедурой унификации. Допускается использование для одной и той же переменной x_j нескольких различных t_j. Перед обращением к процедуре унификации в таких случаях выполняется соответствующая расклейка переменной x_j на несколько различных новых переменных, вместо которых и будут подставляться при унификации соответствующие различные t_j. Унифицирующие антецеденты выделяются указателем "унификация($i_1 \dots i_n$)", где $i_1, \dots, i_n$ - номера этих антецедентов.

10.4.3 Указатели, уточняющие тип основного преобразования

1. Если прием замены, применяемый при сканировании задачи, должен заменять не только текущее вхождение преобразуемого терма, но все его вхождения в термы задачи, для которых удается усмотреть корректность данной замены, то используется указатель "заменавхождений".

2. Если замена выполняется в задаче на доказательство либо на описание, причем нет полной уверенности в том, что она не заведет задачу в тупик, то можно организовать ввод вспомогательной задачи - копии текущей задачи, и выполнить данную замену лишь во вспомогательной задаче. Тогда, после получения отказа на такую вспомогательную задачу, решение текущей задачи будет продолжаться без выполнения рассматриваемой замены; если же на вспомогательную задачу будет получен ответ, то он будет выдан как ответ текущей задачи. Данный режим обеспечивается указателем "попытказамены".

3. Если при выполнении приема "замещениеусловий", передающего во внешнюю задачу на описание найденные в ее блоке анализа следствия, требуется сразу же вернуться к сканированию этой внешней задачи, то применяется указатель "обрывзадачи(A)". Здесь A - фильтр, определяющий условия, при которых необходим такой переход. Он может отсутствовать, и тогда указатель имеет вид "обрывзадачи".

4. Если выполняется замена согласно тождеству "вариант($P\ t_1\ t_2) = t$" (например, при усмотрении модуля), то имеется возможность автоматически обобщить это тождество до "вариант($P\ F(t_1)\ F(t2)) = F(t)$", введя указатель "извлечениеварианта". Прием будет пытаться представить находящиеся "под вариантом" термы в виде $F(t_1)$ и $F(t_2)$, используя процедуру "извлечениеварианта". Заметим, что все входящие в t_1, t_2 переменные должны быть идентифицированы из других частей теоремы (например, из условия P).

5. Если теорема приема проверочного оператора имеет вид $\forall_{y_1 \ldots y_n}(P(\ldots a \ldots)\ \&\ P(\ldots b \ldots) \rightarrow P(\ldots h(a\ b) \ldots))$, где h - ассоциативно - коммутативная операция, то есть прием предпринимает разбиение группы операндов операции h на два подмножества, то возможно предпринять разбиение этой группы операндов сразу на одноэлементные подмножества и проверять $P(\ldots a \ldots)$ последовательно для всех одиночных операндов a. Такой режим вводится указателем "дистрибразвертка(K)", где K - указатель вхождения в теорему операции $h(a\ b)$.

6. Если прием тождественной либо эквивалентной замены, либо прием нормализатора основан на тождестве (либо эквивалентности) типа $P(h(a\ b)) = f(P(a)P(b))$, с ассоциативно - коммутативными f, g, то возможно ввести режим одновременной обработки всех операндов операции h либо f, в зависимости от направления замены. В случае "разгруппировки" операции h применяется указатель "набор(первыйтерм)", иначе (при группировке операндов операции f) - указатель "набор(второйтерм)".

7. Обычно при компиляции тождества, заголовок заменяемой части которого есть ассоциативно - коммутативный символ f, вводится новая переменная x, добавляемая к f - членам обеих частей тождества. Эта переменная идентифицируется с остаточными членами операции f. Если же требуется идентифицировать заменяемую часть целиком со всем списком f - членов преобразуемого терма, не допуская остатка последних, то вводится указатель "модификатор" (в компиляторе модификатором называется упомянутая выше переменная x).

8. В некоторых предметных областях (например, в геометрии) возникает необходимость в определенных преобразованиях теоремы перед компиляцией приема. Такие преобразования обобщают определяемое теоремой описание ситуации, уменьшая потребность в средствах специальной идентификации. В качестве примера здесь можно было бы привести прием, основанный на теореме Пифагора. В этом приеме рассматривается прямоугольный треугольник ABC с прямым углом B. Последнее формулируется как условие перпендикулярности прямых AB и BC. Однако, применяемая стандартизация условий параллельности и перпендикулярности прямых выбирает для каждого класса параллельных прямых лишь один эталонный экземпляр, и все условия параллельности и перпендикулярности для этого класса формулируются с его участием. В результате может оказаться так, что в задаче не найдется условия перпендикулярности для требуемых прямых, а будет условие перпендикулярности для каких-то двух других прямых, DE и FG. Соответственно, в теореме вводятся новые переменные D, E, F, G, а условие перпендикулярности для AB,BC преобразуется в условие перпендикулярности для DE,FG и условия параллельности AB с DE и BC с FG. Такого рода преобразования теоремы выполняются до тех пор, пока не будут устранены все нежелательные связи между объектами, ограничивающие общность описания ситуации. Преобразования эти выполняются

по умолчанию (их обеспечивает процедура "развязка", к которой обращается компилятор). Однако, их можно отключить (иногда это необходимо), вводя указатель "развязка".

9. Если теорема приема имеет вид $A \to P(t_1 \dots t_n)$, где отношение P однозначно определяет значение t_i по значениям $t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n$, то возможно автомати ческое преобразование этой теоремы при компиляции к виду

$$A \to (P(t_1 \dots t_{i-1}\ x\ t_{i+1} \dots t_n) \leftrightarrow x = t_i).$$

Это делается при наличии указателя "однозначно", причем факт однозначного определения t_i распознается компилятором с помощью справочника "однозначно". В качестве примера приведем теорему "натуральное(n) $\to$ наибольший(n, set_m(натуральное(m) & $m|n$))", по которой создается прием, заменяющий утверждения вида "наибольший(A, set_m(натуральное(m) & $m|n$))" на "$A = n$".

10. Если имеется множество альтернативных приемов, которые должны срабатывать на одном и том же уровне, но среди них нужно отобрать один самый "лучший", то применяется специальный механизм, связанный с использованием указателя "оценка($u_1 A_1 \dots A_n v$)". Такая ситуация складывается, например, при выборе той формулы для площади треугольника, которая будет наиболее полезна в решаемой планиметрической задаче. Если промедлить со вводом формулы, то решатель отвлечется на посторонние действия. Поэтому распределить формулы для площади по различным уровням сканирования оказывается невозможным - они должны относиться к одному и тому же уровню. Однако, можно использовать приемы, выписывающие эти формулы, в специальном "двухуровневом режиме". Каждый из них имеет два уровня срабатывания: предварительный уровень u_1 и основной уровень u_2, больший предварительного. На предварительном уровне предпринимается лишь выбор оценки v предпочтительности применения приема (чем меньше оценка, тем выше предпочтительность). Эта оценка сохраняется в комментарии ($A_1 \dots A_n v$) к посылкам задачи. Здесь компоненты $A_1, \dots A_n$ - общие для всех альтернативных применений приемов (например, представляющие собой ссылку на треугольник, для которого нужно выписать формулу площади), а в компоненте v отбирается минимальное значение оценок, сообщаемых приемами при сканировании на уровне u_1. Таким образом, при сканировании на уровне u_2 уже можно отобрать наилучший прием: им окажется тот, чья повторно вычисляемая оценка v совпадет с оценкой w, извлекаемой из комментария ($A_1 \dots A_n w$). Если таких приемов будет несколько, то сработает первый из них. Чтобы заблокировать срабатывание прочих приемов с той же оценкой, сработавший прием может ввести специальный комментарий. Для создания описанного двухуровневого режима срабатывания приема достаточно ввести приведенный выше указатель "оценка(...)". Заметим, что перед A_1 может быть размещен терм "условие(U)", где U - фильтр, определяющий случай для выбора оценки. Чтобы находить оценку для нескольких различных непересекающихся подслучаев, вводятся несколько соответствующих им условных указателей "оценка(...)".

10.4.4 Указатели, играющие роль дополнительных фильтров либо отменяющие ограничения на применение приема

Если прием использует нормализатор A, выполняющий преобразование терма t, причем применение приема теряет смысл при сохранении терма t неизменным, то вводится указатель "различны(A t)", блокирующий в этих случаях срабатывания приема.

Обычно применение приема блокируется, если он пытается преобразовать утверждение, используемое для сопровождения по области допустимых значений других термов задачи. Однако, указатель "сопровождение" позволяет отменить это ограничение. Применять его следует лишь в исключительных случаях, когда либо продолжение сопровождения по о.д.з. становится ненужным, либо усмотрение условий на о.д.з. легко может быть осуществлено и на основе преобразованного утверждения - например, если предпринимаемое преобразование должно синхронным образом изменить и сопровождаемые данным утверждением термы задачи.

10.4.5 Указатели, уточняющие способ формирования новых термов

1. Если квантор существования в заменяющем терме приема замены либо приема нормализатора берется по конечной области, то его бывает целесообразно представить в виде конечной дизъюнкции. В этом случае используется указатель "или(A $B_1 \dots B_n$)", где A - указатель вхождения в теорему приема данного квантора существования; $B_1, \dots, B_n$ - список указателей вхождений конъюнктивных членов подкванторного утверждения, определяющих перечисление дизъюнктивных членов и компилируемых так, как компилируются программно реализуемые антецеденты. В качестве примера рассмотрим теорему "$x < n$ & натуральное(x) $\to \exists_m(m \in \{1, \dots, n-1\}$ & $x = m)$", используемую для вывода условия $x = 1 \vee \dots \vee x = n - 1$ задачи на описание, имеющей неизвестное выражение x, принимающее натуральные значения и ограниченное сверху натуральной константой n. Такое условие инициирует далее разбор случаев. Прием имеет указатель "или(фикс(0)фикс(0 2 1))", у которого первый указатель вхождения в теорему относится к вхождению квантора существования, а второй - к вхождению утверждения $m \in \{1, \dots, n-1\}$, определяющего перечисление натуральных значений m.

2. Предположим, что формируется терм вида $f(t)$ для ранее идентифицированной с некоторым термом $T(x)$ функциональной переменной $f(x)$, причем можно считать достаточно вероятным появление внутри $T(x)$ подтермов $R_1(x), \dots,$ $R_n(x)$, для которых $R_1(t), \dots, R_n(t)$ могут быть упрощены до термов $Q_1, \dots, Q_n$. Тогда можно еще до подстановки t вместо x в $T(x)$ заменить все вхождения $R_1(x), \dots, R_n(x)$ в $T(x)$ на $Q_1, \dots, Q_n$ (выделяя в первую очередь все вхождения $R_1(x)$, затем - не пересекающиеся с ними вхождения $R_2(x)$, и т.д.), и лишь после этого оставшиеся (не накрытые $R_1(x), \dots, R_n(x)$) вхождения x заменить на t. Такой режим формирования $f(t)$ обеспечивается указателем "мультизамена(K $R_1(x)$ $Q_1 \dots R_n(x)$ Q_n)", где K - указатель вхождения в теорему приема терма $f(t)$. В решателе он используется для ускоренного упрощения выражений, возникающих после замены переменной при интегрировании (подстановки Эйлера и др.).

3. Как правило, вводимые в приеме новые термы t обрабатываются цепочкой

нормализаторов, а иногда - также и вспомогательными задачами на преобразование либо описание. В тексте приема не сохраняются координаты того вхождения в теорему терма t, который должен быть подвергнут обработке - считается, что она распространяется на все его вхождения, как в теорему, так и в фильтры приема. Однако, имеется возможность часть вхождений t в теорему и в фильтры формировать с заблокированной обработкой нормализаторами (это бывает полезно, если t встречался в идентифицируемой части теоремы и был отождествлен с некоторым термом задачи T - для ссылок на этот "исходный" вид терма T). Здесь применяется указатель "копия(K)", где K - указатель того вхождения t в теорему, для которого обработка нормализаторами блокируется. Если требуется заблокировать обработку нормализаторами вхождения t в фильтр приема, то в этом фильтре вместо t помещается запись "копия(t)". Наиболее частый случай использования указателя "копия" - блокировка обработки нормализаторами некоторого терма при проверке предварительных условий. Например, в приеме предпринимается попытка разложения на множители некоторой суммы $a + b$, но лишь после того, как установлено, что она неотрицательна. При проверке неотрицательности нет необходимости использовать результат разложения на множители $a+b$, а берется исходный вид этой суммы.

4. Если переменная X была идентифицирована с выражением, заголовком которого служит ассоциативно - коммутативная операция f, то при формировании нового терма вида $f(A \ldots X \ldots B)$ корневые f - операнды X будут объединены в общую группу f - операндов с $A, \ldots, B$. В тех случаях, когда это нежелательно (например, нужно выделить явное вхождение подтерма X, чтобы впоследствии заменить все такие вхождения на новую переменную), применяется указатель "спускоперандов(X)" - он блокирует включение f - операндов идентифицированного с X выражения в группу f - операндов внешней операции.

10.4.6 Указатели, определяющие дополнительные преобразования

Вывод дополнительных посылок и условий

В некоторых случаях целесообразно дополнить выполнение основного преобразования приема (тождественную либо эквивалентную замену, вывод условия либо посылки) занесением в список посылок сопровождающих утверждений, истинность которых является следствием антецедентов теоремы, а также, быть может, ряда других специально проверяемых утверждений. Для этого используется указатель "посылка(условие(A)$B_1 \ldots B_n$ C примечание(K_1) ... примечание(K_m))". Здесь A - фильтр, определяющий условия, при которых занесение посылки целесообразно (он может отсутствовать); $B_1, \ldots, B_n$ - сформулированные в терминах переменных теоремы утверждения, истинность которых должна быть дополнительно установлена с помощью проверочных операторов перед занесением новой посылки; C - сама новая посылка (тоже в терминах теоремных переменных); $K_1, \ldots, K_m$ - комментарии, которыми сопровождается эта посылка. При выводе C проверяется, что все утверждения, использованные для проверки корректности преобразований приема и для проверки $B_1, \ldots, B_n$, относятся к списку посылок текущей задачи. Заметим, что фактически терм "примечание(K_i)" здесь имеет вид "примечание($D_1 \ldots D_p$)", так что K_i есть набор операндов $(D_1 \ldots D_p)$, начинающийся с логического символа D_1 (заголовка комментария); $p \geq 0$. Как и обычно, по умолчанию D_j при $0 < j$ компилируются

в виде термов; если требуется вместо этого иметь в комментарии переменную либо логический символ, то используется запись "переменная(D_i)". Типичный случай использования указателя "посылка(...)" - вывод в геометрическом приеме таких дополнительных утверждений, характеризующих чертеж, которые полезны лишь при выполнении специальных условий.

Особый случай - занесение дополнительных посылок, используемых при сопровождении по области допустимых значений. Такие посылки необходимо своевременно формировать в процессе преобразований, чтобы легко было усматривать выполнение условий по о.д.з. на новые термы, введенные приемом. Так как выполнение даже самых простых алгебраических преобразований связано с проверкой выполнения условий по о.д.з., отсутствие сопровождающих утверждений может полностью заблокировать решение задачи. Отличие от общего случая со вводом дополнительных посылок заключается, во-первых, в том, что сопровождающее утверждение может быть занесено не только в список посылок, но и в список условий (если прием преобразует условие задачи на описание), и во-вторых, в том, что может происходить автоматический контроль неиспользуемости для сопровождения по о.д.з. ранее введенных утверждений, с устранением их по мере возможности. Здесь используется указатель "вывод(условие(A)C эквивалентно($i_1 \dots i_n$))"; A - фильтр проверки целесообразности вывода дополнительного утверждения C; $i_1, \dots, i_n$ - номера некоторых антецедентов теоремы, истинность которых является следствием утверждения C и остальных антецедентов теоремы (подтерм "эквивалентно(...)", как и "условие(...)", может отсутствовать). Утверждения задачи, с которыми были идентифицированы указанные утверждения, если они после выполнения преобразования приема не используются в качестве сопровождающих по о.д.з., удаляются. В качестве примера ситуации, требующей сопровождения преобразований вводом дополнительной посылки с помощью указателя "вывод", рассмотрим исключение иррациональности в знаменателе выражения $a/(\sqrt{}(b) - \sqrt{}(c))$ путем домножения числителя и знаменателя на $\sqrt{}(b) + \sqrt{}(c)$. Знаменатель $b - c$ новой дроби может быть существенно видоизменен после приведения подобных членов, и для того, чтобы последующие приемы не испытывали затрудений при усмотрении отличия его от нуля, нужно сразу же вводить посылку $b - c = 0$. Так как нормализаторы применяются ко всем вхождениям терма $b - c$, то выполняемое ими приведение подобных членов будет предпринято и в новом знаменателе, и в сопровождающей посылке.

Если решается задача на описание, то появляется возможность выполнять преобразование подвыражения ее условия, используя в заменяющем терме вспомогательные обозначения, определяемые с помощью вводимых тем же приемом новых условий. Для этой и возможных других целей служит указатель "выводусловия(A $B_1 \dots B_n$)", позволяющий сопровождать основные действия приема занесением в список условий текущей задачи на описание нового условия A. Данное условие снабжается комментариями, вводимыми в термах B_i вида "комментарий(...)"; возможен случай $n = 0$.

Удаление посылок и условий

Если в результате применяемого преобразования пропадает терм, требовавший специального сопровождения по о.д.з., то прием может предпринять попытку исключить утверждения, обеспечивавшие такое сопровождение. В случае исключения посылки для этого вводится указатель "удалениепосылки(условие(A)B)"; в случае удаления условия - указатель "удалениеусловия(условие(A)B)". Здесь B - удаляемое

утверждение (в теоремных переменных); A - фильтр, проверяющий целесообразность удаления (он может отсутствовать). Если B действительно является посылкой (соответственно, условием) текущей задачи, то после выполнения определяемой приемом замены проверяется, используется ли оно все еще в качестве сопровождающего по о.д.з. утверждения, и если не используется, то удаляется. Пример использования указателя - при решении уравнения $\mathrm{tg}(x) = a$ предпринимается попытка удалить условие $\neg(\cos(x) = 0)$, обеспечивавшее до этого сопровождение по о.д.з для устраненного тангенса.

Новые переменные, вводимые приемом

Прием может вводить новую переменную - такая переменная не имеет вхождений в идентифицируемые термы приема, но встречается в формируемых приемом новых термах. Если речь идет о связанной переменной, возникающей при вводе нового квантора либо описателя, то никаких специальных указателей для нее не требуется - компилятор обеспечит ввод ее автоматически. В противном случае нужны специальные указатели, уточняющие способ ввода данной переменной. При отсутствии этих указателей компиляция приема не состоится.

1. Простейший указатель ввода новой переменной X имеет вид "новаяпеременная(X)". При его наличии компилятор просто выберет для построения нового терма с X некоторую не встречающуюся в текущем контексте переменную. Обозначение для нее будет выбираться, по мере возможности, с учетом условий на X из антецедентов теоремы и с учетом контекста задачи - для неизвестных будут выбраны $x, y, z, \ldots$, для целочисленных переменных - $m, n, k, \ldots$, для точек - $A, B, C, \ldots$, и т.д. Часто встречающаяся ситуация со вводом новой переменной - использование ее в качестве вспомогательного обозначения. Например, вычисляется производная некоторой функции f, обозначаемая новой переменной g, и в посылки задачи заносятся как утверждение "Производная(f, g)", связывающее g с f, так и равенство, явно определяющее g.

2. Иногда нужно ввести не единственную переменную, а сразу кортеж переменных, длина которого определяется в зависимости от контекста. Например, это бывает при работе с функциями многих переменных. Тогда применяется указатель "переменные(X N)", определяющий, что переменная X идентифицируется с набором новых переменных, длина которого определяется значением теоремной переменной N, идентифицированной с десятичной записью числа.

3. Если новые переменные $X_1, \ldots, X_n$ должны быть присоединены к списку неизвестных текущей задачи на описание либо исследование, то используется указатель "новыенеизвестные ($X_1 \ldots X_n$)".

4. Если прием вводит дополнительные построения - рассматривает некоторые новые объекты, условия на которые заносятся в список посылок текущей задачи, то особо должен быть учтен случай задачи на исследование - в этом случае предпринимается перенесение основных условий на новые объекты в список посылок внешней задачи на описание. Здесь используется указатель "новыйсимвол(X $K_1 \ldots K_n$)", где X - новая переменная; $K_1, \ldots, K_n$ - указатели вхождений в теорему основных утверждений о свойствах X, которые целесообразно перенести в список посылок внешней задачи на описание. Хотя в процессе обучения

решателя оказалось, что такая предусмотрительность является лишь формальностью - список посылок внешней задачи на описание фактически полностью отключен от процесса решения - указатель "новыйсимвол(...)" применяется во всех приемах по планиметрии, связанных с дополнительными построениями.

5. При решении задачи на исследование, связанной с задачей на описание вычислительного типа (например, вычислительные задачи в планиметрии), либо при решении задачи на доказательство путем вычислений в посылках, существенным является разделение переменных на неизвестные внешней задачи, неизвестные "внутреннего" характера (это просто обозначения вспомогательных объектов, которые не должны войти в окончательный ответ, например, обозначения точек на чертеже), и известные параметры. Часто для решения задач такого типа применяется прием, вводящий как бы известный новый параметр, через который далее будут выражаться прочие переменные задачи. Впоследствии либо окажется, что удалось найти выражения для неизвестных, в которых такой вспомогательный параметр "сократился", либо будут предприняты дополнительные действия по определению его значения после выражения через него неизвестных задачи. Тогда он сам поменяет статус вспомогательной "известной" на вспомогательную "неизвестную". Для первоначального ввода вспомогательного "известного" параметра X применяется указатель "вспомпараметр(X $K_1 \dots K_n$)", где $K_1, \dots, K_n$ - указатели вхождений в теорему тех утверждений о X, добавляемых при вводе этого параметра к списку посылок, которые одновременно должны быть зарегистрированы в списке посылок внешней задачи на описание. Заметим, что при вводе вспомогательного параметра все веса посылок задачи обращаются в 0; это способствует срабатыванию полезных приемов при повторном сканировании ее посылок, независимому от введения дополнительных "известных" параметров. Заметим также, что один прием может вводить несколько вспомогательных параметров (каждый требует своего указателя).

6. Другой способ придать нужную целенаправленность действиям по решению задачи на исследование, имеющей цель "известно", либо задачи на доказательство - ввести вспомогательную неизвестную X, которая в случае задачи на исследование переносится также в список неизвестных внешней задачи на описание, однако найденные ее значения не включаются в ответ такой задачи. Фильтры многих и многих приемов (например, по планиметрии) активно реагируют на связь с неизвестными внешней задачи тех или иных числовых параметров текущей задачи (расстояния, углы и проч.), так что фактически ввод вспомогательной неизвестной оказывается примерно равноценен решению вспомогательной задачи на нахождение значения такой неизвестной. Впрочем, вычисление ее значения происходит здесь в более гибком режиме - параллельно с прочими действиями по решению основной задачи, что часто гораздо более предпочтительно, чем грубое прерывание общего процесса анализа посылок и локализация внимания только на вспомогательном вычислении. Для указания на то, что прием вывода фактически вводит обозначение для новой неизвестной X, используется указатель "вспомнеизвестная(X)". Прием может вводить сразу несколько вспомогательных неизвестных, связанных между собой утверждениями, присоединяемыми им к списку посылок. В этом случае вводятся несколько отдельных указателей "вспомнеизвестная(...)". В случае задачи на

доказательство ввод вспомогательной неизвестной сопровождается понижением до 0 весов всех посылок и веса условия.

Обработка надтермов

Если применяется преобразование тождественной либо эквивалентной замены, то оно может повлечь за собой различного рода нарушения стандартизации вида термов, и как следствие - цепочку срабатываний мелких нормализующих приемов. Такого рода цепочки срабатываний увеличивают число сканирований и могут ощутимо повлиять на время решения задачи. Для борьбы с ними, в первую очередь, применяются различного рода нормализаторы, обеспечивающие необходимую стандартизацию внутри самого заменяющего терма. Однако, эти нормализаторы не выполняют восстановления стандартизации для надтермов заменяемого терма. Чтобы выполнять такую стандартизацию надтермов, в оператор ЛОСа "заменавхождения(...)", используемый для реализации тождественных и эквивалентных замен при сканировании задачи, встроена процедура, последовательно просматривающая надтермы (в направлении их увеличения) и обрабатывающая их посредством пакетных нормализаторов общей стандартизации (эти нормализаторы находятся с помощью справочника "нормализатор"). Эта процедура активизируется, если прием выделен указателем "норм". Примером использования этого режима служит группа приемов, выполняющих сложение и упрощение дробей - для проверки возможности применения последующей стандартизации дробей (деление дроби; деление на дробь; сокращение дроби и т.п.).

Аналогичный механизм стандартизации надтермов может быть включен при помощи указателя "нормализатор". Такой указатель обычно выделяет приемы общей стандартизации. После некоторых замен, нарушающих стандартизацию для надтермов замененного терма, может возникнуть длинная цепочка простых стандартизирующих преобразований. Эти преобразования, в целях уменьшения трудоемкости, выгодно применять вне сканирования. Поэтому, как только к одному и тому же терму задачи применяются подряд два приема, снабженные указателем "нормализатор" (что может служить сигналом о наличии указанной выше длинной цепочки стандартизации), автоматически включается специальная процедура, выполняющая полную обработку этого терма пакетными нормализаторами. Эта же процедура обеспечивает необходимую коррекцию структур данных, выполняющих сопровождение по о.д.з. Обращение к указанной процедуре (заголовком ее служит логический символ "блок-нормализации") происходит из программы "заменавхождения", обеспечивающей замены при сканировании задачи.

Коррекция содержимого буферов обращений к пакетным операторам

После выполнения преобразования может сложиться ситуация, когда автоматическое использование таких хранящихся в буфере обращений к заданному нормализатору результатов, которые содержат некоторый терм t, нецелесообразно - прием только что "избавился" от этого терма. Здесь применяется указатель "контрольнормализации(A t)", инициирующий исключение из буфера обращений к нормализатору A всех результатов, содержащих подтерм t. Возможно также исключение из буфера всех результатов обращений к A - для этого достаточно не указывать t.

Особые случаи сопровождения по области допустимых значений

1. Если прием выполняет замену некоторого утверждения, которое может использоваться в качестве сопровождающего по о.д.з. (при отключенной блокировке таких замен - на это отключение приходится идти при завершающем редактировании ответа), то рекомендуется сохранять информацию о том, что его исходная версия выводима из новой версии (а также из прочих обеспечивавших корректность замены утверждений из текущего контекста), вводя указатель "стандследствие". Эта информация сохраняется в комментарии "стандследствие ..." к посылкам исходной задачи и может быть использована далее оператором "стандследствие", к которому обращаются все проверочные операторы.

2. Если сопровождающие по о.д.з. утверждения нецелесообразно регистрировать в списках посылок и условий задачи, но тем не менее указания на то, что они являются следствиями конкретных посылок и условий, могут пригодиться, то применяется указатель "вычерк". Прием в этом случае будет сохранять информацию о сопровождающих утверждениях в комментарии "стандследствие ...". Фактически такой указатель был использован только для приема, выполняющего формальное интегрирование путем обращения к нормализатору "нормИнтеграл".

3. Иногда, наоборот, целесообразно зарегистрировать сопровождающие по о.д.з. условия для новых термов, даже не проверяя того, что они суть следствия старых условий и посылок (например, они могут возникнуть при вычислении пределов либо суммировании рядов как необходимые условия существования предела либо суммы ряда). Тогда вводится указатель "внешвывод" - он инициирует автоматическое перенесение всех новых утверждений из накопителя "коррекцияпосылок ..." в контекст замены (например, в список посылок).

4. В определенных случаях сопровождение по о.д.з. может все-таки нарушаться. Это не означает, что в контексте требующего сопровождения терма отсутствуют утверждения, следствием которых являются необходимые условия на о.д.з. - просто усмотрение из них этих условий неочевидно, и в комментариях явных ссылок на эти утверждения как на сопровождающие по о.д.з. нет. Такая ситуация возникает, как правило, лишь в особых случаях - после обращения к вспомогательным задачам на описание. При редактировании ответа данной задачи все сопровождающие по о.д.з. утверждения могут быть явно разрешены относительно своих параметров (такова обычная стандартизация ответов), и тогда явное сопровождение становится неявным. Если полученный таким образом ответ используется далее в задаче, то явное сопровождение необходимо восстанавливать - без него будут заблокированы многие важные приемы. Восстановление обеспечивается специальными приемами, снабженными указателем "коррекцияодз". Эти приемы суть обычные приемы вывода в посылках либо условиях; применение их начинается с усмотрения в задаче требующего сопровождения по о.д.з. терма A (например, квадратного корня) - он определяется указателем "контрольвывода(A)". Кроме регистрации в посылках либо условиях сопровождающих утверждений, прием осуществляет соответствующую коррекцию комментария "сопровождение"к текущей задаче.

Пополнение ответа задачи

Теорема приема "ответзадачи" явно перечисляет список условий на неизвестные, которые выдаются в качестве ответа. Иногда этот ответ целесообразно пополнить некоторыми утверждениями, отсутствующими в указанном списке, но являющимися его следствиями. Тогда применяется указатель "плюс(условие(A)$B_1 \dots B_n$)", где A - фильтр, определяющий целесообразность присоединения к ответу утверждений $B_1, \dots, B_n$ (фильтр может отсутствовать).

Расширение списков переменных, указанных в целях задачи

Вводимые при решении задачи новые переменные иногда бывает нужно зарегистрировать в тех или иных целях текущей задачи. Для этого можно использовать указатель "установка(A X)". Переменная X заносится тогда в список $B_1 \dots B_n$ цели, имеющей вид (A $B_1 \dots B_n$). Этот же указатель можно применять и в ситуации, когда заносимый в цель ($A \dots$) объект - не переменная, а терм. Тогда вместо X в указателе помещается запись "терм(T)", где T - заносимый терм.

Для достаточно часто возникающей необходимости пополнять новыми переменными цель "обозначение $X_1 \dots X_k$" (она перечисляет вспомогательные переменные, которые не должны входить в ответ задачи на описание либо преобразование) предусмотрен специальный указатель "обозначения($x_1 \dots x_p$)", где $x_1, \dots, x_p$ - добавляемые к цели "обозначение ..." переменные.

10.4.7 Указатели, уточняющие точку привязки приема

Приемы сканирования задачи инициализируются при рассмотрении вхождения в задачу некоторого логического символа. Это вхождение называется точкой привязки приема; для уточнения ее вида служат следующие указатели:

1. Если прием определяет вывод следствий в посылках либо условиях задачи, то инициализация попытки его применения не всегда связана с обнаружением при сканировании задачи явного вхождения одного из антецедентов теоремы приема. Иногда она может быть вызвана обстоятельствами весьма косвенного характера - обнаружением где - либо в задаче некоторого выражения либо утверждения T, при работе с которым могут понадобиться формулируемые приемом свойства входящих в T объектов. Чтобы инициализация приема начиналась с обнаружения при сканировании терма T, используется указатель "контрольвывода(T)". В качестве примера рассмотрим прием вывода следствий в посылках, записывающий соотношение для площади равнобедренного треугольника:

   $l(AB) = l(AC)$ & $D \in$ прямая(BC) & $l(BD) = l(CD) \rightarrow 2S($фигура($ABC$)$) = l(AD)l(BC)$.

   Указатель "контрольвывода(площадь(фигура(набор(ABC))))" определяет в качестве точки привязки приема вхождение выражения $S($фигура(ABC)$)$, которое в антецедентах теоремы не встречается.

2. Чтобы явно указать логический символ A, к которому должна быть отнесена программа приема - тогда его срабатывание будет начинаться с усмотрения при сканировании некоторого вхождения символа A - используется указатель "точкапривязки(A)".

3. Как правило, заменяемый терм теоремы для приема замены должен быть невырожденным, не сводящимся к обычной либо функциональной переменной, иначе трудно определить точку привязки. Однако, иногда бывает нужно идентифицировать такой вырожденный терм однозначным образом - с условием текущей задачи на преобразование. В этом случае применяется указатель "корень".

4. Если точку привязки требуется выбрать в антецеденте теоремы, имеющем номер i, то вводится указатель "теквхожд(i)". Фактически оказалось, что он стал использоваться лишь в сочетании с указателем "усм(i)" для антецедента, имеющего вид утверждения о перпендикулярности. Исключение составляет единственный случай, когда из соображений переключения внимания в приеме замены понадобилось точку привязки выбрать в обычном (идентифицируемом с посылкой задачи) антецеденте.

5. Чтобы выделить в приеме "ответзадачи" тот конъюнктивный член теоремы, который желательно использовать в качестве точки привязки, применяется указатель "смответ(i)", где i - символьный номер данного конъюнктивного члена; нумерация начинается с 1.

10.4.8 Указатели, определяющие преобразования комментариев

Ввод нового общего комментария в случае применения приема

Чтобы при срабатывании приема вводился новый комментарий к текущей задаче (в случае задач на исследование - комментарий к посылкам задачи) либо комментарий к пакетному оператору, используется указатель "замечание($A_1 \dots A_n$)". A_1 - логический символ, являющийся заголовком комментария; $A_2, \dots, A_n$ - операторные выражения для элементов комментария. Типы допустимых здесь выражений - те же, что при задании комментариев в фильтрах приемов. Возможен случай $n = 1$, и тогда комментарий состоит из логического символа A_1. Комментарий вводится лишь непосредственно перед обращением к процедуре ЛОСа, выполняющей преобразования приема. Однако, в определенных случаях блокировка срабатывания может произойти уже внутри последней процедуры - если преобразование оказалось вырожденным, либо нарушающим сопровождение по о.д.з, либо вследствие учета каких-либо общих установок на решение задачи. Если в таких случаях важно не допустить ввода комментария, то перед A_1 в указанной выше записи добавляется символ "результат". Комментарий формируется с отбрасыванием этого символа, и вводится только после фактического выполнения преобразования. Наконец, перед A_1 возможно поместить терм "условие(B)", где B - фильтр, и тогда комментарий будет вводиться только при истинности условия B.

Если применяется прием нормализатора, но комментарий требуется ввести не для нормализатора, а для текущей задачи, то перед A_1 в указателе "замечание(...)" помещается логический символ "текзадача".

По умолчанию, разрешается вводить множество копий одного и того же комментария. Иногда это используется для контроля глубины цепочки однотипных преобразований (например, применений правила Лопиталя для вычисления пределов). В тех случаях, когда требуется предотвратить дублирование комментариев, для ввода

нового комментария нормализатора либо анализатора, вместо "замечание(...)" применяется указатель "замечусловие($A_1 \dots A_n$)".

Если требуется ввести комментарий к посылкам текущей задачи (при применении приема сканирования задачи), то используется указатель "комментариипосылок($A_1 \dots A_n$)". Чтобы гарантировать ввод комментария только в случае фактического применения преобразования приема, перед A_1 помещается логический символ "результат".

Ввод комментария к новому либо преобразованному терму задачи

Если требуется снабдить комментарием новую посылку либо условие, сформированные при выводе следствий, либо преобразованный приемом терм задачи, либо новое утверждение, введенное анализатором, то используется указатель "примечание($A_1 \dots A_n$)" либо "примечание(условие(B $A_1 \dots A_n$))". Набор $A_1, \dots, A_n$ определяет комментарий; B - фильтр, указывающий дополнительное условие для ввода этого комментария.

Ввод комментария к терму задачи, идентифицированному с заданным антецедентом

Новым комментарием можно снабдить посылку либо условие задачи, идентифицированные с заданным антецедентом теоремы приема. Для этого используется указатель "примечпосылки(i $A_1 \dots A_n$)". i - номер антецедента; набор $A_1, \dots, A_n$ определяет вводимый комментарий. Формат задания комментария - такой же, как в случае "замечание(...)".

Проверка отсутствия комментария с последующим его вводом

Комментарии могут играть роль "защелки", пропуская внутрь приема при первой попытке применения и блокируя повторные попытки. В таких случаях в процессе применения приема сначала предпринимается проверка отсутствия заданного комментария, а впоследствии, в зависимости от типа примененного указателя, его ввод. Используются указатели следующих типов:

1. Простейший тип защелки - ввод комментария (к задаче, нормализатору либо анализатору) непосредственно после проверки его отсутствия. Здесь применяется указатель "ключ($A_1 \dots A_n$)", где A_1 - заголовок комментария; $A_2, \dots, A_n$ - его элементы. Формат термов $A_2, \dots, A_n$ при работе с защелками отличается от обычного, использованного в фильтрах и ранее описанных указателях. Каждый из них представляет собой либо логический символ (при этом служебный символ "теквхожд" обозначает текущий идентифицируемый в приеме подтерм), либо терм в теоремных переменных, либо определяющий вхождение терм "вхождение(C)", где C - операторное выражение.

   Проверка отсутствия комментария и его ввод происходят немедленно после идентификации всех переменных термов $A_2, \dots, A_n$. Если этот момент нужно отложить до того, как дополнительно будут идентифицированы переменные $X_1, \dots, X_m$, то к списку $A_2, \dots, A_n$ добавляется элемент "фикс($X_1 \dots X_m$)" (он никак не изменяет самого комментария). Использование в $A_2, \dots, A_n$ указателей вхождения в теорему не предусмотрено. Еще один способ отложить момент проверки отсутствия комментария и его ввода - размещение в конце списка

$A_2, \ldots, A_n$ терма "посылки($i_1 \ldots i_k$)", перечисляющего номера тех антецедентов, истинность которых должна быть установлена до указанного момента.

2. Следующий тип "защелки" - проверка отсутствия комментария и ввод его при откате, если прием оказался не реализованным. Указатель здесь имеет вид "попытка($A_1 \ldots A_n$)". Как и в предыдущем случае, проверка отсутствия комментария происходит сразу же после идентификации всех переменных, входящих в термы $A_2, \ldots, A_n$. Чтобы отложить этот момент, можно применять термы "фикс($X_1 \ldots X_m$)" и "посылки($i_1 \ldots i_k$)".

3. Если проверка отсутствия комментария и его ввод должны происходить непосредственно перед применением преобразования приема, то используется указатель "биключ($A_1 \ldots A_n$)". Записи "фикс(...)" и "посылки(...)" здесь излишни.

4. Предшествующие указатели вводили общий комментарий задачи (в случае задач на исследование - комментарий к посылкам задачи) либо пакетного оператора. Чтобы проверить отсутствие комментария к текущему терму задачи (внутри которого находится точка привязки приема) и сразу после этого ввести данный комментарий, применяется указатель "первыйключ($A_1 \ldots A_n$)". Проверка происходит сразу же после идентификации переменных термов $A_2, \ldots, A_n$; чтобы отложить этот момент, можно применять записи "фикс($X_1 \ldots X_n$)" и "посылки($i_1 \ldots i_k$)".

5. Чтобы ввести комментарий к текущему терму задачи не сразу после проверки отсутствия этого комментария, а лишь при откате, если прием не был реализован, применяется указатель "Попытка($A_1 \ldots A_n$)". Он совершенно аналогичен приведенному выше указателю "попытка($A_1 \ldots A_n$)".

Коррекция ранее введенного комментария

Чтобы изменить ранее введенный комментарий, применяется указатель "изменение($A\ B_1 \ldots B_n\ C_1 \ldots C_k$)". Здесь A - идентифицирующий терм "комментарий(...)" либо "комментарийпосылки(...)" либо "примечание(...)"; $B_1, \ldots, B_n$ - фильтры, определяющие дополнительные условия на введенные в терме A разряды преобразуемого комментария (могут отсутствовать); $C_1, \ldots, C_k$ - термы вида "замена($i\ P$)", где i - номер изменяемой позиции комментария (номер 0 имеет заголовок комментария); P - операторное выражение, значением которого является заменяющий объект.

Часто встречается ситуация, в которой комментарий используется как накопитель объектов некоторого типа. При первом появлении такого объекта комментарий должен быть введен, а при каждом последующем появлении - список объектов в уже имеющемся комментарии должен быть пополнен. В случае комментариев к посылкам задачи данный режим вводится указателем "Примечание($A\ B$)". A - логический символ, являющийся заголовком комментария. Комментарий имеет вид ($A\ N$), где набор N - накопитель объектов. Операторное выражение B должно определять логический символ либо переменную.

Удаление комментария

При выполнении приема замены, реализуемого в сканировании задачи, можно удалить заданный комментарий. Для этого используется указатель "удалениезамечания($A\ B_1 \ldots B_n$)". Здесь A - идентифицирующий терм "комментарий(...)" либо "ком-

ментарийпосылки(...)" либо "примечание(...)"; $B_1, \ldots, B_n$ - фильтры, определяющие дополнительные условия на введенные в терме A разряды удаляемого комментария (могут отсутствовать). В идентифицирующем терме A допускается явное указание комментария, без каких-либо идентифицируемых переменных. Если же такие переменные есть, то удаляются все комментарии выделенного типа.

Аналогичным образом, для удаления комментария к нормализатору используется указатель "удалениезамечания($B\ A_1 \ldots A_n$)". Здесь B - фильтр, определяющий условие, при котором происходит удаление комментария; $A_1, \ldots, A_n$ - набор, определяющий удаляемый комментарий - в стандартном формате, как для указателя "замечание(...)".

Ввод комментариев при обращении к проверочному оператору, синтезатору либо задаче на доказательство, обрабатывающим заданный антецедент

1. Если для проверки истинности заданного антецедента либо его реализации происходит обращение к вспомогательной задаче, проверочному оператору либо синтезатору, то при обращении им может быть передан заданный комментарий. Для этого используется указатель "комментарий(i условие(B) $A_1 \ldots A_n$)". Здесь i - номер антецедента; B - фильтр, определяющий условие ввода комментария; $(A_1 \ldots A_n)$ - набор, определяющий комментарий. Формат задания комментария здесь стандартный (такой же, как в указателе "замечание(...)").

2. Если происходит обращение к задаче на доказательство антецедента, имеющего вид кванторной импликации $\forall_{x_1 \ldots x_n}(P_1\ \&\ \ldots\ \&\ P_m \rightarrow P_0)$, то $P_1, \ldots, P_m$ становятся посылками данной задачи. Утверждение P_i как посылку при этом можно снабдить заданным комментарием. Для этого используется указатель "комментарий(K условие(B)$A_1 \ldots A_n$)", аналогичный приведенному выше. Однако, здесь уже K является не номером антецедента, а указателем вхождения в теорему приема утверждения P_i

3. При обращении из пакетного оператора к проверочному оператору либо синтезатору, обрабатывающему некоторый антецедент, по умолчанию происходит передача комментариев текущего оператора к подоператору. Чтобы передавать не все такие комментарии, а только имеющие заголовки $A_1, \ldots, A_n$, используется указатель "комментарии($i\ A_1 \ldots A_n$)", где i - номер антецедента. Если нужно, наоборот, удалить все комментарии с данными заголовками, то используется указатель "удалениепримечания($i\ B\ A_1 \ldots A_n$)", где B - фильтр, определяющий условия, при которых происходит удаление (может отсутствовать).

4. Наконец, предусмотрено удаление конкретных комментариев при обращении из пакетного оператора к проверочному оператору либо синтезатору. Здесь используется указатель "исключение(i условие(B)$A_1 \ldots A_n$)", где i - номер антецедента; B - фильтр, определяющий условия, при которых происходит удаление комментария; $A_1, \ldots, A_n$ - набор, задающий комментарий в стандартном формате (как для указателя "замечание(...)").

Ввод комментария к вспомогательной задаче, возникающей при попытке замены

Прием замены, имеющий указатель "попытказамены", не предпринимает, как уже говорилось выше, непосредственного изменения текущей задачи, а вводит вспомо-

гательную задачу - копию текущей, в которой уже и реализуется данная замена (при неудаче решения последней происходит откат к прерванному решению текущей задачи). Возможна передача комментария этой вспомогательной задаче. Здесь применяется указатель "Замечание($A_1 \dots A_n$)", где набор $A_1, \dots, A_n$ определяет комментарий в том же формате, как и для указателя "замечание(...)".

Специальные комментарии, используемые для учета взаимной выводимости утверждений

Во многих случаях приемы вывода следствий в действительности определяют эквивалентное преобразование одной из использованных посылок. Например, при получении линейной комбинации двух уравнений с ненулевыми коэффициентами возможен обратный логический переход от нее к одному из исходных уравнений, в предположении истинности второго уравнения. Учет такого рода обратных переходов может существенно сократить трудоемкость решения задачи, устраняя проверку исходных уравнений. Для реализации его применяется указатель "прообраз(i посылки($A_1 \dots A_n$) вывод($B_1 \dots B_m$))". Здесь $A_1, \dots, A_n$ - дополнительно проверяемые проверочными операторами утверждения. Если удалось установить их истинность, то прием дополнительно выводит следствия $B_1, \dots, B_m$ и регистрирует в комментарии "прообраз ..." тот факт, что i - й антецедент является следствием консеквента, прочих использованных при выводе утверждений из контекста идентификации, а также утверждений $B_1, \dots, B_m$. Списки $A_1, \dots, A_n$ и $B_1, \dots, B_m$ могут быть пустыми. Заметим, что фактически указатель "прообраз(...)" применялся только в приемах анализаторов.

Использование фильтра, проверяющего существование объектов некоторого типа, для сохранения в комментариях списка всех таких объектов

Если прием нормализатора должен не только проверить при оценке целесообразности своего срабатывания существование объектов, удовлетворяющих заданному условию, но также передать внешним приемам список всех таких объектов либо некоторых характеристик этих объектов, то можно разместить внутри фильтра "контекст(...)" (например, в качестве последнего операнда) указатель "буфер(A R)". В этом случае данный фильтр не ограничится отысканием единственного набора значений своих внутренних парметров, при котором его условия истинны, а станет перечислять все такие значения. Для каждого из них будет определяться значение операторного выражения R, и это значение будет регистрироваться в накопителе N комментария (A N); повторяющиеся элементы в накопитель не заносятся. При обращении к нормализатору должен быть введен его комментарий с заголовком A. Фактически данный режим был использован в нормализаторе "повторчисло", обеспечивающем выделение в явном виде повторяющихся вхождений подвыражений с неизвестными. После того, как некоторый прием этого нормализатора изменяет одно из уравнений таким образом, чтобы в нем возникло выражение с неизвестными, встречающееся в другом уравнении, ссылка на такое внешнее уравнение заносится в специальный комментарий, который по окончании работы с текущим уравнением инициирует повторное рассмотрение зарегистрированных в нем уравнений.

Ввод комментариев к внешнему нормализатору

Иногда бывает нужно передать комментарий одной из внешних процедур - согласно цепочке обращений, в конце которой находится текущая реализуемая процедура.

Например, это может быть связано с передачей сообщения о целесообразности разбора случаев, которая выяснилась лишь в текущей подпроцедуре. Если внешняя процедура представляет собой нормализатор, то для такой передачи комментария предусмотрен указатель "нормализация(A B $C_1 \ldots C_n$)". Здесь A - заголовок внешнего нормализатора; B - фильтр, определяющий условие передачи комментария; $C_1, \ldots, C_n$ - набор, определяющий комментарий в стандартном формате. В фильтре B можно ссылаться на текущий обрабатываемый внешним нормализатором A терм посредством служебного логического символа "внешкорень". Если имеется несколько внешних нормализаторов с заголовком A, для которых выполняется условие B, то комментарий передается каждому из них. Фильтр B может отсутствовать.

Предусмотрен вариант, в котором комментарий передается не всем внешним нормализаторам A (если их несколько), а лишь тому из них, который ближе всего по цепочке обращений расположен к текущей процедуре. Здесь используется указатель "учетнормализации(A B $C_1 \ldots C_n$)".

10.4.9 Указатели, определяющие переключение внимания

После того, как был применен прием, может оказаться вероятным срабатывание некоторых других приемов. Чтобы такое срабатывание стало возможным, необходимо уменьшить веса тех посылок либо условий, при рассмотрении которых эти другие приемы активизируются - "переключить внимание". Наиболее вероятные точки, для которых целесообразно осуществить переключение внимания, выявляются в процессе обучения решателя на потоке задач (например, просто сохраняются указания на переключение внимания, оказавшиеся необходимыми для решения отдельных задач обучающей выборки). Ниже приводятся типы указателей, обеспечивающих переключение внимания.

1. Наиболее часто используемый указатель - "новаяпосылка(X A N)". В этом указателе A представляет собой фильтр, определяющий условие на посылку X текущей задачи, вес которой должен быть понижен до N (если он уже не был меньшим или равным N). N - явно указываемое символьное число; X - новая переменная, используемая в контексте данного указателя - она пробегает все посылки текущей задачи. Если перед X размещается запись "условие(P)", то понижение весов будет предприниматься только при истинности фильтра P.

2. Аналогичным образом используется указатель "новоеусловие(X A N)", у которого X пробегает все условия текущей задачи на описание. В приеме явно должно быть указано, что он применяется к задаче на описание.

3. Если требуется понизить до величины N вес посылки, с которой был идентифицирован i - й антецедент теоремы, то применяется указатель "актив(i N)".

4. По умолчанию, при компиляции приема сканирования задачи, в начале его программы вставляется оператор "новый". Этот оператор блокирует применение приема в циклах повторного сканирования терма T задачи, попавшего на некоторый период в "теневую зону" сканирования, предпринимаемых перед выходом его из "теневой зоны" (т.е. когда текущий уровень задачи становится на 1 меньше веса данного терма). Однако, если срабатывание приема существенно зависит от возможности появления за период пребывания в "теневой

зоне" терма T новых посылок либо условий, такой оператор нежелателен. Устранение его обеспечивается применением указателя "новый". Заметим, что в предметных областях, характеризующихся большими длинами списка посылок (например, в геометрии) неосторожное применение указателя "новый" может существенно замедлить работу решателя. Компенсировать его отсутствие приходится явными указателями на переключение внимания.

5. Чтобы понизить до 0 веса всех условий и посылок задачи, содержащих некоторый терм T (записанный в теоремных переменных), используется указатель "внимание(T)". Если вместо T использовать служебный символ "условие", то понижается до 0 вес условия задачи. Эту возможность следует применять лишь тогда, когда в описании приема явно указывается, что он применяется для задачи на преобразование либо на доказательство, т.е. ссылка на условие задачи корректна.

6. Если прием изменяет посылку либо условие, либо вводит новую посылку или условие, то измененный либо новый терм задачи получает вес 0, цикл сканирования прерывается и возобновляется с нулевого текущего уровня. Если же прием изменяет только комментарии задачи, то такого прерывания цикла сканирования не происходит. Однако, иногда после ввода нового комментария необходимо искусственно понизить до 0 вес текущего терма задачи и возобновить сканирование с нулевого текущего уровня - чтобы сработали те приемы, которым адресован комментарий. В этом случае прием с заголовком "замечание" (изменяющий только комментарии задачи) снабжается указателем "пересмотр".

10.4.10 Ограничитель трудоемкости

Для предотвращения неоправданно длительных попыток применения приема можно использовать несколько специальных указателей, вводящих слежение за трудоемкостью попытки реализации приема и обрывающих эту попытку в случае превышения установленного лимита. В связи с обработкой заданного антецедента теоремы уже был описан ограничитель трудоемкости "Обрыв(...)". Здесь приведен еще один указатель такого типа - "лимит(N $A_1 \ldots A_n$)". У него N - число шагов работы интерпретатора ЛОСа, по превышении которого попытка применения приема обрывается. $A_1, \ldots, A_n$ - термы, используемые для уточнения той позиции в программе приема, после которой устанавливается лимит в N шагов. Именно, устанавливающий лимит оператор ЛОСа вводится сразу же, как только оказываются идентифицированы все переменные данных термов. Термы $A_1, \ldots, A_n$ могут отсутствовать.

10.4.11 Отложенная фильтрация

При применении приема не всегда удается сразу хорошо оценить действительные последствия его действий. Однако, часто можно сформулировать достаточно легко проверяемое условие P на желательные последствия. Тогда используется "отложенная" фильтрация для приема: проверка данного условия предпринимается не сразу, а лишь через заданное число шагов N. Эта фильтрация задается с помощью указателя "контроль(P лимит(N) K)". Если через N шагов не будет выполнено условие P, то произойдет откат к тому состоянию задачи, которое имелось до применения приема. При этом задача будет снабжена комментариями, определяемыми термами

"замечание(A)", перечисляемыми в списке K (этот список может быть пустым). Для организации режима отложенной фильтрации вводится комментарий (контроль ...) к списку посылок текущей задачи. Он анализируется процедурой "Контроль(...)", обращения к которой происходят при каждом выходе на программу логического символа - типа задачи при сканировании.

10.4.12 Специальные указатели пакетных операторов

Нормализаторы

1. Уровень срабатывания приема нормализатора устанавливается не фильтром, как в случае приемов остальных типов, а указателем "уровень(A)". Здесь предусматривается единственное значение уровня.

2. Если в описании формата нормализатора содержится логический символ "корень", то все его приемы будут применяться только к корневому вхождению в преобразуемый терм. Для прочих нормализаторов такое ограничение на срабатывание приема можно установить, введя указатель "корень". В частности, этот указатель может оказаться необходимым, если заменяемый терм представляет собой функциональную переменную.

3. Достаточно часто входная информация логического уровня предметной области передается нормализатору через комментарии. Например, нормализатор разложения в ряд Тейлора получает указание на переменную x, по которой ведется разложение, и на точку a, в окрестности которой оно выполняется, через комментарий (рядтейлора x a). Для дополнительной идентификации теоремных переменных через такой комментарий используется указатель "вход(A $x_1 \dots x_n$)". Здесь A - заголовок комментария, который должен иметь вид (A b_1 $\dots b_n$), причем переменные $x_1, \dots, x_n$ идентифицируются с $b_1, \dots, b_n$. Компилятор предполагает, что $b_1, \dots, b_n$ суть термы.

4. Если требуется, чтобы сразу после выполнения текущего преобразования нормализатор выдал полученный терм в качестве окончательного результата, то прием снабжается указателем "выход". В качестве примера использования такого указателя приведем приемы группировки тригонометрических выражений при разложении на множители (см. нормализатор "видумножение"). Прием, выполняющий некоторую группировку, предпринимает попытку разложить полученное выражение на множители, и если это удалось (а тогда автоматически и каждый из множителей будет подвергнут попыткам доразложения), то продолжение попыток разложения на множители результирующего выражения бесполезно.

5. Как уже отмечалось, при выполнении преобразований приемами сканирования задачи часто бывает необходимо обеспечивать специальные действия по поддержанию сопровождения по о.д.з. - иначе выполнение последующих приемов будет заблокировано из-за невозможности убедиться в выполнении самых элементарных требований на о.д.з. для рассматриваемых выражений. Аналогичные действия необходимо выполнять и в случае применения нормализатора. Для этого служит указатель "вывод(условие(P)A эквивалентно($i_1 \dots i_n$))". Этот указатель дополняет преобразование выводом следствия A, необходимого для поддержания сопровождения по о.д.з. A регистрируется в комментарии

"коррекцияпосылок B" и заносится в список посылок нормализатора (фактически заносятся конъюнктивные члены A). P - фильтр, указывающий условие, при котором предпринимается вывод следствия. $i_1, \ldots, i_n$ - номера антецедентов теоремы приема, истинность которых является следствием утверждения A и остальных антецедентов теоремы. Фильтр P и список $i_1, \ldots, i_n$ могут отсутствовать. Заметим, что комментарий "коррекцияпосылок B" по завершении применения нормализатора используется для создания во внешней задаче, обратившейся к нормализатору, необходимых дополнительных посылок и комментариев, обеспечивающих сопровождение по о.д.з.

6. Если нормализатор обращается к преобразованиям вспомогательного выражения, содержащего новую переменную x, введенную в качестве обозначения для некоторого терма t, а затем возвращается к исходным обозначениям, заменяя в результате преобразований x на t, то может понадобиться коррекция сохраненной в комментарии "коррекцияпосылок A" информации о выводимости утверждений, содержащих x. Для этого используется указатель "подствпосылки(x t)", обеспечивающий подстановку t вместо x в таких утверждениях.

7. Иногда возникает необходимость таким образом подобрать значения входящих в выражение A параметров, чтобы это выражение оказалось возможным преобразовать к некоторому специальному виду. Один из примеров этой ситуации - подбор показателя степени, при котором степенная подстановка делает правую часть дифференциального уравнения зависящей только от отношения y к x. В таких случаях целесообразно воспользоваться нормализаторами, так как подбору параметра обычно предшествует некоторая цепочка преобразований. В тот момент, когда становится "видимой" подсказка относительно возможного подбора параметра, применяется прием нормализатора, основанный на фиктивном тождестве. Левая (заменяемая) часть этого тождества идентифицируется с содержащим "подсказку" о подборе параметров фрагментом преобразуемого терма; правая часть представляет собой конъюнкцию равенств, определяющих подбор параметров. Собственно проверка допустимости данного подбора переносится в обработку антецедентов псевдотеоремы приема (например, здесь может быть предпринята явная подстановка параметров в терм и попытка преобразования его к требуемому виду, выполняемая другим нормализатором либо вспомогательной задачей). Прием оформляется как обычный прием тождественной замены. Однако, в то время как заменяемая часть его идентифицируется с подтермом преобразуемого терма, заменяющая часть вводится вместо всего преобразуемого терма. Этот режим обеспечивается указателем "нормвывод". Заметим, что обычно такие приемы являются завершающими, и в них используется также указатель "выход".

Проверочные операторы

1. Как правило, прием проверочного оператора имеет альтернативный характер: при неудаче его применения будут предприниматься попытки выполнить проверку другими средствами. Однако, иногда он сводит проверку истинности рассматриваемого утверждения к проверке истинности других утверждений (конъюнкция последних должна быть эквивалентна истинности первого), и при неудаче одной из этих проверок сразу выдается отказ. Под выдачей отказа здесь и далее понимаем, что значение проверочного оператора, как оператора ЛОСа,

становится равным "ложь"; никакого логического символа "отказ" при этом не используется. Например, проверка отличия от 0 произведения нескольких выражений сводится указанным образом к проверке отличия от 0 каждого из сомножителей. Чтобы ввести такой режим функционирования приема, служит указатель "спуск". После проверки истинности "несущественных" антецедентов теоремы, указывающих типы значений переменных и другие необходимые элементы сопровождения по о.д.з., включается установка на выдачу отказа после неудачной проверки истинности хотя бы одного из оставшихся антецедентов.

2. Последний режим может применятсья также в тех случаях, когда эквивалентность проверяемого утверждения конъюнкции подмножества антецедентов теоремы усматривается лишь после проверки истинности части других ее антецедентов. Например, если при проверке нечетности суммы двух целых чисел установлено, что первое слагаемое четное, то проверка сводится к установлению нечетности второго слагаемого. В этой ситуации применяется указатель "спуск $(i_1 \dots i_k)$", перечисляющий номера $i_1, \dots, i_k$ тех антецедентов теоремы, после установления истинности которых, а также установления истинности "несущественных" антецедентов, включается установка на выдачу отказа после неудачной проверки истинности хотя бы одного из остальных антецедентов.

3. Наконец, прием проверочного оператора может сводить выполняемую проверку к рассмотрению конкретного антецедента, и если его истинность установить не удалось, то сразу будет выдан отказ. Для этого применяется указатель "внешобрыв(i)", где i - номер данного антецедента.

4. Иногда бывает достаточно найти хоть какую-нибудь идентификацию части антецедентов теоремы приема проверочного оператора, определив при этом значения $A_1, \dots, A_n$ некоторых теоремных переменных $x_1, \dots, x_n$, и проводить оставшиеся действия внутри данного приема лишь для указанных значений. Если эти действия довести до конца не удается, то альтернативные возможности идентификации для $x_1, \dots, x_n$ не рассматриваются, и выполнение приема обрывается. Однако, в отличие от предыдущих случаев, здесь не происходит выдачи отказа - просто проверку в таких случаях целесообразно проводить другими средствами. Чтобы ввести данный режим, используется указатель "откат(i)", где i - номер того антецедента, после идентификации которого осуществляется фиксация определенных к текущему моменту теоремных переменных.

5. В определенных ситуациях возможно быстрое усмотрение заведомой ложности проверяемого утверждения. Тогда, для ускорения выхода из бесперспективной проверки, используются специальные приемы проверочного оператора, имеющие указатель "не". Консеквент теоремы такого приема имеет вид отрицания утверждения, проверяемого рассматриваемым проверочным оператором. Если удается установить истинность антецедентов теоремы и идентифицировать проверяемое утверждение с отрицанием консеквента, то сразу выдается отказ.

6. Можно обеспечить выход из проверочного оператора с выдачей отказа, если не удалось установить (также с помощью проверочных операторов) истинность антецедентов, номера которых перечислены в указателе "сброс($i_1 \dots i_m$)".

Анализаторы

Работа анализатора заключается в сканировании утверждений, занесенных в его буфер, а также посылок текущей задачи на исследование, и занесении в буфер следствий указанных посылок и утверждений буфера. Возможно преобразование специальными приемами ранее занесенных в буфер утверждений. В процессе вывода некоторые заносимые в буфер утверждения могут снабжаться комментарием "внешвывод" (вводимым с помощью стандартных указателей); по окончании работы анализатора такие утверждения будут перенесены в список посылок текущей задачи на исследование - собственно они и составляют результат применения анализатора.

Если прием анализатора выводит настолько ценное следствие, что целесообразно безотлагательное перенесение его во внешнюю задачу и возвращение к сканированию этой задачи, то используется указатель "обрыв(A)", где A - фильтр, уточняющий условие обрыва работы анализатора. В случае обрыва сформированное приемом утверждение автоматически снабжается комментарием "внешвывод" и таким образом переносится во внешнюю задачу.

Если необходимо преобразовать ранее занесенное в буфер утверждение, то используются приемы, теорема которых представляет собой эквивалентность. Левая часть этой эквивалентности идентифицируется с преобразуемым утверждением, а правая - определяет новый вид утверждения. Такие приемы снабжаются указателем "внутрпреобр".

10.4.13 Указатели, определяющие отбор и сохранение ссылок на задачи, рассмотрение которых представляет интерес для развития приема

Для усиления и оптимизации приема может оказаться полезным накопление списков ссылок на задачи, при решении которых он сработал либо, наоборот, был заблокирован тем или иным из своих фильтров. Имеется общий механизм, обеспечивающий сохранение в архиве всех ссылок на задачи, при решении которых происходило срабатывание приема. Такие данные автоматически обновляются при каждом запуске серийного решения всех задач выбранного раздела задачника, и использование их не предполагает введения в приемы каких-либо специальных указателей. Предусмотрена, однако, возможность сохранения ссылок на избранные задачи, в которых сработал прием, а также ссылок на задачи, в которых он не сработал из-за специально указанных причин.

Если требуется начать накопление закрепленного за приемом списка ссылок на задачи, в которых он сработал, причем в момент срабатывания выполнялось дополнительное условие, определяемое фильтром A, то вводится указатель "задачи(i A)". Здесь i - произвольно определяемый номер установки на отбор. Таких установок в приеме может вводиться несколько, и для каждой будет создаваться свой список ссылок.

Если требуется создать список задач, в которых прием не сработал из-за того, что срабатывание было заблокировано фильтром A, то этот фильтр помещается в описании приема внутри записи "задачи(i A)", где i - снова номер установки на отбор задач.

Для просмотра списка отобранных задач нужно выделить многоцветной указкой вхождение рассматриваемого указателя либо фильтра "задачи(...)" и нажать правую клавишу мыши либо клавишу "з".

10.4.14 Указатели, определяющие формирование информации для трассировки

Ввод новых элементов чертежа при дополнительных построениях

При решении геометрических задач, для которых изначально был создан чертеж, возможно пополнение этого чертежа в процессе дополнительных построений. Для такого пополнения используются перечисляемые ниже указатели.

1. Чтобы ввести на чертеже точку A - основание перпендикуляра, опущенного из точки B на прямую, проходящую через точки C и D, служит указатель "перпендикулярно(A B C D)". Заметим, что сам перпендикуляр при этом не прорисовывается - для его прорисовки необходимо добавить приводимый ниже указатель "отрезок(A B)".

2. Чтобы провести отрезок, соединяющий точки A и B, используется указатель "отрезок(A B)".

3. Чтобы ввести на чертеже точку A пересечения прямых, проходящих через точки B, C и D, E соответственно, используется указатель "точкапрямой(A B C D E)".

4. Чтобы выбрать произвольным образом точку A, лежащую на отрезке с концами B и C, используется указатель "точкаотрезка(A B C)".

5. Чтобы ввести точку A внутреннего касания окружности с центром B, проходящей через точку C, и окружности с центром D, проходящей через точку E, используется указатель "внутркасаются(A B C D E)". Аналогичным образом, в случае внешнего касания указанных окружностей применяется указатель "внешкасаются(A B C D E)".

6. Чтобы ввести точку A пересечения прямой, проходящей через точки B, C, с прямой, проведенной из точки D параллельно прямой, проходящей через точки E, F, используется указатель "параллельны(A B C D E F)".

7. Чтобы нарисовать окружность с центром в точке A, проходящую через точку B, используется указатель "окружность(A B)".

8. Чтобы ввести на чертеже центр A окружности, описанной около треугольника с вершинами в точках B, C, D, и прорисовать саму окружность, используется указатель "описана(A B C D)".

9. Чтобы отложить на отрезке с концами B, C точку A, отстоящую от B на расстояние, равное расстоянию между точками D, E, используется указатель "расстояние(A B C D E)".

10. Чтобы ввести на чертеже точку A пересечения перпендикуляра к прямой, проходящей через точки B и C, восставленного в точке D, с прямой, проходящей через точки E и F, используется указатель "перпендикулярны(A B C D E F)".

11. Чтобы ввести точку A на луче BC, расстояние которой до точки B, выраженное в пикселях, равно символьному числу r, используется указатель "луч(A B C r)". Явное указание расстояния означает, что выбор на луче точки A в некотором

смысле произволен (например, вводится конец координатного вектора), а константа r подбирается из соображений достаточной различимости точек на чертеже.

12. Тесно примыкает к предыдущему указатель "прямкоорд(A B C D r)", вводящий точку A на перпендикуляре к прямой BC, восставленном в точке B, лежащую от прямой BC по ту же сторону, что и точка D; расстояние от A до B, выраженное в пикселях, равно символьному числу r.

Ввод контрольного прерывания перед применением приема

Чтобы при отладке проще было отслеживать моменты срабатывания приема, можно использовать указатель "стоп". Его наличие обеспечивает выход в отладчик ЛОСа перед моментом выполнения преобразований. Разумеется, после отладки этот указатель следует удалить.

Шаблоны и указатели текстформульного сопровождения приема

Если в описании приема не предусмотреть специальных средств для вывода на экран пояснений при его срабатывании, то по умолчанию будет выводиться текст того концевого пункта оглавления базы приемов, в котором размещен этот прием. Во многих случаях это, в сочетании с возможностью просмотра описания приема на ГЕНОЛОГе, достаточно наглядно объясняет смысл выполняемых действий. Чтобы получить более качественное сопровождение пошаговых действий решателя, предусмотрена возможность создания в описании приема специальных текстформульных шаблонов выводимых на экран пояснений.

Собственно текстовая часть шаблона выводится при просмотре описания приема на экран при нажатии клавиши "6". Эта часть распадается на несколько независимых фрагментов, каждый из которых начинается с символа ¬, вслед за которым идет номер фрагмента (последовательность цифр, завершающаяся пробелом). Нумерация начинается с 1; если фрагмент всего один, то запись ¬1 отбрасывается. Для указания мест, на которые должны вставляться формулы, используются пары расположенных подряд скобок (). Если фрагмент единственный и запись ¬1 отброшена, то при вводе текста следует отступить от левого края на одну позицию, иначе первая буква пропадет. Напомним, что для ввода символа ¬ текстовым редактором служит клавиша "Ctr-n".

Чтобы указать, какие именно формулы следует вставлять, и какие именно фрагменты в каких случаях использовать, служат указатели "титр(P)", размещаемые вместе с прочими указателями приема. Здесь P - выражение, построенное при помощи условных выражений "вариант(...)" из атомарных указателей текстформульного сопровождения. Каждый такой атомарный указатель имеет вид "набор($A\,B_1 \ldots B_n$)", где A - символьный номер используемого фрагмента текстового шаблона, связанного с приемом; $B_1, \ldots, B_n$ - информация о заполнении текстового шаблона. Если в некотором подслучае нужно отменить отображение срабатывания приема, то используется указатель "стоп". Условная конструкция "вариант(...)" может отсутствовать, и тогда во всех случаях пояснение будет строиться по одному и тому же шаблону; в противном случае для разных ситуаций будут извлекаться различные шаблоны.

Информационные элементы B_i, уточняющие использование текстового шаблона для создания текстформульного сопровождения срабатывания приема, бывают следующих двух типов:

1. терм($C_1 \dots C_n$) - $C_1, \dots, C_n$ суть операторные выражения, определяющие термы, подставляемые последовательно в текстовый шаблон вместо пар скобок (). Для указания на теоремный терм T, не подлежащий обработке нормализаторами, внутри C_i можно использовать запись "фикс(T)".

2. параметры($X_1 \dots X_m$ D) - $X_1, \dots, X_m$ суть теоремные переменные, для которых при подстановке в термы $C_1, \dots, C_n$ информационного элемента "терм($C_1 \dots C_n$)" вводятся новые переменные, обозначающие объекты типа D (при отсутствии D берутся первые неиспользуемые большие буквы).

Чтобы сопровождать пояснениями обращения к нормализаторам и другим пакетным операторам, эти обращения сопровождаются комментариями "титр(P)", аналогичными приведенному выше указателю "титр(P)" (например, при обращении к нормализатору используется запись "замечание(титр(P))".

Для боле гибкого учета различных ситуаций можно использовать в одном и том же описании приема несколько указателей "титр(...)", вводящих располагаемые подряд независимо определяемые текстформульные фрагменты.

10.5 Нормализаторы приема

При применении приема формируются различные термы - как вспомогательные (например, необходимые для принятия решения или представляющие собой промежуточные результаты вычислений), так и основные (входящие в новую посылку или в новое условие, либо в заменяющий терм при тождественной или эквивалентной замене, и т.п.). Любой из таких термов может быть преобразован цепочкой обращений к пакетным нормализаторам либо вспомогательным задачам. Эта цепочка обращений определяется специальной группой служебных термов ГЕНОЛОГа, называемых нормализаторами приема. Каждая группа нормализаторов связана с обработкой определенного терма A, который выделяется либо непосредственно в теореме приема, либо в некотором фильтре приема (в том числе в фильтре, встречающемся внутри указателя). Заметим, что обработке заданной группой нормализаторов подлежат все термы, равные A, независимо от их вхождения в теорему или в фильтры - нет необходимости связывать ее с каждым из таких вхождений A (для выделения вхождений терма A, не подлежащих нормализации, используется, как отмечалось выше, указатель "копия"). Собственно для выделения A служит специальный интерфейс: сначала в теореме или в фильтре многоцветной указкой выбирается какое-либо вхождение A; нажимается Enter, и в возникающем при этом окне текстового редактора набирается группа нормализаторов, относящихся к A.

Каждый нормализатор приема представляет собой либо обращение к пакетному нормализатору, либо обращение к задаче, либо указатель коррекции посылок, относящийся сразу ко всей цепочке преобразований (можно вводить либо удалять заданные посылки, в предположении истинности которых будут проводиться преобразования). Преобразования выполняются в соответствии с указанным порядком, слева направо. Разделители между нормализаторами - пробелы.

10.5.1 Обращения к пакетным нормализаторам

Каждое обращение к пакетному нормализатору либо состоит из названия A этого нормализатора, либо имеет вид $A(B_1 \dots B_n)$. $B_1, \dots, B_n$ - специальные термы, назы-

ваемые указателями обращения к нормализатору. Возможные типы таких термов приводятся ниже:

1. "замечание($A_1 \dots A_n$)" - вводится, если его не было, комментарий ($A_1 \dots A_n$). Перед A_1 возможно размещение терма "условие(B)", уточняющего условие, при котором вводится комментарий. В этом случае после A_n можно разместить терм "альтернатива($C_1 \dots C_m$)", указывающий комментарий ($C_1 \dots C_m$), вводимый при невыполнении условия B. Формат, в котором здесь задаются комментарии - тот же, что и для указателя приема "замечание(...)". Кроме того, можно явно задавать комментарий посредством выражения "префикс(...)"; в этом случае указатель обращения имеет вид "замечание(префикс(...))". Заметим, что по умолчанию все комментарии текущего пакетного оператора передаются нормализаторам, используемым в приеме этого пакетного оператора.

2. "примечание($A_1 \dots A_n$)" - вводится (даже если такой комментарий уже был) новый экземпляр комментария ($A_1 \dots A_n$). В остальном аналогично указателю "замечание(...)".

3. "удалениезамечания($A_1 \dots A_n$)" - удаляется комментарий ($A_1 \dots A_n$). Перед A_1 возможно размещение терма "условие(B)", уточняющего условие, при котором происходит удаление.

4. "исключкоммент($A_1 \dots A_n$)" - при обращении к нормализатору отбрасываются все комментарии с заголовками $A_1, \dots A_n$

5. "условие(P)" - P есть фильтр, накладывающий дополнительные ограничения на применение данного пакетного нормализатора A.

6. "вариант(A_1 A_2)" - при выполнении условия A_1 берется альтернативный пакетный нормализатор, имеющий заголовок A_2.

7. "комментарии($A_1 \dots A_n$)" - при обращении нормализатору передаются все комментарии текущей задачи, имеющие заголовки $A_1, \dots, A_n$.

8. "лимит(N)" - при превышении N шагов обращение к нормализатору обрывается (предпринимается откат к установленному перед обращением оператору "лимит(...)"). Если обращение к нормализатору не превысило N шагов, то после него установленный лимит сбрасывается. Этот указатель не отменяет слишком долгого процесса нормализации, передавая далее ненормализованный терм, а просто блокирует в этом случае дальнейшее применение приема.

9. "расширениепосылок" - к списку посылок нормализатора присоединяются следствия данного списка, введенные ранее использованными в приеме нормализаторами. Такое пополнение списка посылок позволяет нормализатору использовать утверждения, введенные предшествующими нормализаторами для сопровождения по о.д.з. измененных выражений (эти утверждения извлекаются из сквозного накопителя "коррекцияпосылок ...", заполняемого нормализаторами приема).

10.5.2 Обращения к вспомогательным задачам

Нормализатор приема, обеспечивающий обращение к вспомогательной задаче для преобразования нормализуемого терма, имеет вид "задача($A\ B_1 \dots B_n$)", где A - уровень обращения к вспомогательной задаче; $B_1, \dots, B_n$ - указатели на формирование задачи. Те B_i, которые суть логические символы (кроме указываемых далее служебных слов), определяют цели новой задачи. Прочие типы указателей перечисляются ниже.

1. По умолчанию, вводится задача на преобразование. Чтобы ввести задачу на описание, используется указатель "тип(описать)".

2. Чтобы перенести в новую задачу все цели текущей задачи, имеющие заголовки $A_1, \dots, A_n$, используется указатель "цель(повторение($A_1 \dots A_n$))".

3. Если логический символ A_2 должен быть введен в качестве цели новой задачи лишь при выполнении фильтра A_1, то используется указатель "цель(условие(A_1 A_2))".

4. Чтобы ввести цель (неизвестные $x_1 \dots x_n$), в которой перечисляются все переменные $x_1, \dots, x_n$, идентифицированные с переменными $X_1, \dots, X_m$ либо встречающиеся в идентифицированных с ними списках переменных, служит указатель "цель(неизвестная($X_1 \dots X_m$))". Другую возможность ввести цель (неизвестные $x_1 \dots x_n$) дает указатель "цель(неизвестные($X_1 \dots X_m$))". Здесь $x_1, \dots, x_n$ суть все неизвестные текущей задачи на описание, являющиеся параметрами подтермов, идентифицированных с переменными $X_1, \dots X_m$.

5. Аналогичным образом, чтобы ввести цель (параметры $x_1 \dots x_n$), в которой перечисляются все переменные $x_1, \dots, x_n$, идентифицированные с переменными $X_1, \dots, X_m$ либо встречающиеся в идентифицированных с ними списках переменных, используется указатель "цель(параметры($X_1 \dots X_m$))".

6. Чтобы была введена цель ($A_1 \dots A_n$), используется указатель "цель(набор($A_1 \dots A_n$))". Формат описания цели здесь такой же, как при описании комментариев.

7. Для передачи вспомогательной задаче списка комментариев A текущего пакетного нормализатора, к которому относится реализуемый прием, используется указатель "цель(нормализация)". Он вводит в формируемую задачу цель (нормализация A).

8. Иногда бывает нужно разрешить нормализуемое утверждение относительно заданного списка выражений $t_1, \dots, t_n$, которые были идентифицированы с теоремными переменными $x_1, \dots x_n$. Для этого используется указатель цели вспомогательной задачи на описание "цель(вспомописание($x_1 \dots x_n$))". Условие этой задачи получается из нормализуемого утверждения заменой в нем выражений $t_1, \dots, t_n$ на вспомогательные неизвестные $z_1, \dots, z_n$, относительно которых и решается задача. В найденный ответ вместо переменных $z_1, \dots, z_n$ подствляются выражения $t_1, \dots, t_n$, и таким образом получается результат нормализации.

9. Для передачи формируемой задаче комментария ($A_1 \dots A_n$) используется указатель "комментарий(условие(P)$A_1 \dots A_n$)". Здесь P - фильтр, определяющий

условие ввода комментария; подтерм "условие(P)" может отсутствовать. Комментарий задается в стандартном формате - как в указателе приема "замечание($A_1 \dots A_n$)".

10. Указатель "лимит(N)" определяет число шагов N, по превышении которого выдается отказ на задачу. Как и в случае пакетного нормализатора, такой отказ означает блокировку дальнейшего выполнения приема и откат.

11. Если происходит обращение к вспомогательной задаче на преобразование из приема пакетного нормализатора, то для регистрации в списке посылок нормализатора всех новых посылок, введенных при решении этой задачи, служит указатель "учетодз".

12. Если нужно заблокировать ввод при обращении к вспомогательной задаче комментария "коррекцияпосылок A" - накопителя информации, используемой для коррекции сопровождения по о.д.з. после ее решения, то используется указатель "коррекцияпосылок".

10.5.3 Указатели коррекции посылок

Нормализатор приема "посылки($A_1 \dots A_n$)" используется для указания на то, что все обращения к пакетным операторам и вспомогательным задачам, определяемые данной цепочкой нормализаторов приема, выполняются при дополнительных посылках $A_1, \dots, A_n$. Здесь $A_1, \dots, A_n$ - утверждения в теоремных переменных.

Для исключения заданных утверждений из списков посылок всех пакетных операторов и вспомогательных задач, определяемых рассматриваемой цепочкой нормализаторов приема, используется нормализатор приема "удалениепосылок(x A)". Здесь A - фильтр, определяющий условие на исключаемое утверждение, обозначенное в нем переменной x.

Глава 11

Редактор приемов ГЕНОЛО́ГА

11.1 Просмотр описаний приемов

11.1.1 Вход в редактор приемов

Вход в просмотр и редактирование приемов, реализованных на ГЕНОЛОГе, осуществляется через оглавление приемов. Войти в это оглавление можно из главного меню, нажав клавишу "г" или выбрав левой кнопкой мыши окно "Оглавление приемов". Приемы распределены в оглавлении, в целом, по тематическому принципу. Это распределение нужно лишь для удобства поиска приема вручную и самим решателем никак не используется.

В корневом меню оглавления приемов перечисляются основные разделы, в которых происходило обучение. Начинается перечисление с раздела "Логические приемы", в котором собраны реализованные на ГЕНОЛОГе общелогические приемы (подраздел "Общие приемы"), а также приемы вывода теорем и проч. Одним из последних размещается раздел "Словарь", в котором собраны приемы текстового анализатора. Он организован по принципу обычного словаря: подразделы обозначены буквами от "А" до "Я". В зависимости от того, были ли сохранены или сброшены данные о последних изменениях в базе приемов, корневое меню может завершаться разделом "Буфер". В трех его подразделах "Новые приемы", "Измененные приемы" и "Удаленные приемы" указываются имевшие место изменения. Степень проработки различных разделов, представленных в базе приемов, весьма различна. В некоторых случаях проработка лишь начата на простейших единичных примерах, в других - доведена до более-менее устойчивого решения стандартных задач.

После выбора нужного концевого пункта оглавления приемов нажимается клавиша "курсор вправо" или Enter, и на экране появляется запись приема, относящегося к выбранному пункту.

Для выхода из просмотра приема обратно в оглавление базы приемов нажимается клавиша "курсор влево" либо Esc.

К одному и тому же концевому пункту оглавления могут относиться несколько приемов; смена этих приемов осуществляется клавишами "курсор вверх" - "курсор вниз".

11.1.2 Отображение приема на экране

Описание приема размещается в четырех окнах, отделенных друг от друга горизонтальными линиями; слева расположены номера этих окон. В первом окне находится

теорема приема; во втором - заголовок приема; в третьем - список фильтров приема, и в четвертом - список указателей приема. Обращения к нормализаторам выводятся на экран с помощью описываемого ниже специального интерфейса и при общем просмотре не видны. Если одно из окон пусто (таким может быть либо третье, либо четвертое окно), то номер его не прорисовывается, а верхняя и нижняя горизонтальные линии прорисовываются на расстоянии двух-трех пикселей друг от друга. В исключительных случаях могут оказаться пустыми оба (третье и четвертое) окна - таких случаев при создании новых приемов следует избегать, так как здесь описание приема (та его часть, которая остается за вычетом теоремы приема и его заголовка) может оказаться пустым, и прием не будет сохранен в файлах.

В верхней части первого окна может быть размещен чертеж. Это делается лишь для большей наглядности теоремы приема; компилятором ГЕНОЛОГа чертеж не используется.

Если описание приема слишком велико, то содержимое третьего и (или) четвертого окон разрезается на страницы. При просмотре будет выдаваться лишь одна из таких страниц, причем интерфейс позволяет перелистывать страницы двух окон независимо друг от друга. Признаком наличия предыдущей страницы служит запятая в начале текста окна; признаком наличия следующей страницы - запятая в конце окна.

Кроме того, для работы с большими описаниями приемов можно временно убирать с экрана одно или несколько окон. i - е окно убирается и восстанавливается на экране при нажатии клавиши Ctr-Fi.

Иногда текстовый (в скобочной записи термов) вид теоремы приема занимает меньше места, чем стандартный, формульный. Тогда для уменьшения размеров описания приема можно перейти к текстовому режиму отображения теоремы. Это делается путем нажатия клавиши "т". Для восстановления формульного режима нажимается "ф".

Если для каких-либо логических символов или сочетаний таких символов не предусмотрен режим отображения формульным редактором, то теорема приема прорисовывается текстовым редакторов в скобочной записи.

11.1.3 Просмотр компонент описания приема

Применение цветовой указки для выделения элементов описания приема

Обычно все элементы описания приема размещены на экране одновременно; это упрощает восприятие приема "в целом", однако при чтении сколь-нибудь громоздких фрагментов описания бывает удобно применять многоцветную указку, аналогичную той, которая применялась при чтении программ ЛОСа. Эта же указка позволяет получить справочную информацию относительно тех или иных терминов, использованных в описании приема.

Чтобы войти в режим применения многоцветной указки, следует прежде всего выделить то окно, в котором предполагается ее использовать. Для этого существует два способа - либо нажать клавишу "курсор вправо", и тогда будет выбрано окно номер 1 (номер выбранного окна перекрашивается в малиновый цвет), либо сразу нажать клавишу i для номера требуемого окна; i - 1,2,3 либо 4. Смену номера выбранного окна можно выполнять клавишами "курсор вверх - курсор вниз". После того, как нужный номер выделен, нажатие клавиши "курсор вправо" переводит собственно в просмотр элементов окна с применением цветовой указки. Для третьего и

четвертого окон такой просмотр совершенно аналогичен применявшемуся в редакторе программ ЛОСа: смена терма выполняется клавишами "курсор вправо - курсор влево"; вход в первый операнд текущей операции - "курсор вниз"; выход обратно - "курсор вверх". По достижении "корневого" уровня выход из просмотра текущего окна и возвращение к режиму выбора номера окна - тоже клавишей "курсор вверх". Во втором окне многоцветная указка включается лишь тогда, когда заголовок приема более чем односимвольный.

Просмотр теоремы приема, записанной в текстовом (скобочном) виде, происходит так же, как просмотр термов третьего и четвертого окон. Если же теорема изображена в формульном виде, то для выделения ее подтермов используется одноцветный режим: выделенная часть перекрашивается в малиновый цвет. В остальном принципы использования клавиатуры - те же, что для многоцветной указки: вход в первый операнд операции - "курсор вниз"; смена операнда - "курсор влево - курсор вправо"; выход в надоперанд - "курсор вверх".

Для быстрого выхода из режима цветовой указки используется клавиша "пробел" - нажатие ее в любом контексте использования цветовой указки сразу возвращает в режим общего просмотра приема.

Для быстрого выделения конкретного элемента описания приема, расположенного в третьем либо четвертом окнах, можно также использовать мышь: если ее курсор разместить на представляющем интерес элементе и нажать левую клавишу, то он будет выделен многоцветной указкой. Этот же режим можно применять и для первого окна. Если теорема записана в формульном виде, то мышью бывает трудно выделить нужный подтерм для инфиксных операций - вместо него выделяется какой-либо подфрагмент, например, переменная. В таких случаях можно сначала выделить мышью что-то близкое к нужному подтерму, а затем скорректировать выбор подфрагмента клавишами курсора.

Нажатие левой клавиши мыши вне элементов описания приема, как и нажатие пробела, возвращает к режиму общего просмотра приема.

Если третье либо четвертое окно разрезано на несколько страниц, то для перехода к следующей или предыдущей его странице нужно войти в режим цветового выделения элементов этого окна и использовать клавиши "PageDown", "PageUp". Можно также переместить текущий указатель к запятой, расположенной в конце либо в начале окна, и нажать клавишу "курсор вправо" либо, соответственно, "курсор влево". Смену страницы можно выполнить, не входя в просмотр элементов окна - для этого нужно подвести курсор мыши к соответствующей запятой и нажать левую кнопку мыши.

Смена формульного и текстового режимов просмотра теоремы

Для просмотра теоремы приема предусмотрены два режима - обычный формульный и технический текстовый, при котором отображается используемая программами ЛОСа скобочная запись. Чтобы ввести формульный режим, следует нажать клавишу "ф"; чтобы ввести текстовый режим - нажать клавишу "т". Выбранный режим сохраняется на все время пребывания в оглавлении приемов и сбрасывается при выходе из него; по умолчанию при входе в оглавление устанавливается формульный режим.

Просмотр нормализаторов приема

Для просмотра нормализаторов приема нажимается клавиша "5". После этого под горизонтальной линией, завершающей четвертое оконо приема, возникает набор нормализаторов приема, обрабатывающих некоторый его терм A. Одновременно терм A выделяется цветовой указкой в теореме приема либо, если он встречался не в теореме, а в фильтрах либо указателях, в соответствующих текстах третьего либо четвертого окон. Чтобы перейти к просмотру цепочки нормализаторов, связанных со следующим термом A, нажимается клавиша "PageDown"; чтобы вернуться обратно - клавиша "PageUp". Выйти из данного режима просмотра цепочек нормализаторов, связанных с различными термами A, можно только одним способом - нажатием клавиши "End".

Если требуется внести изменения в текущую цепочку нормализаторов, нажимается клавиша "н" - тогда в начале записи нормализаторов возникает курсор текстового редактора. После завершения редактирования (нажатие Enter либо, в случае отказа от изменения записи, - Esc) - восстанавливается указанный выше режим просмотра. Вносимые здесь и в других случаях редактирования фрагментов описания приема изменения не затрагивают хранящейся в файлах записи приема; они лишь изменяют копию этой записи, используемую для текущего отображения на экране. Для фактического изменения приема нужно после возвращения в общий режим просмотра нажать F3 либо F4. Другая возможность изменять нормализаторы приема (по существу, основная) будет описана ниже - в разделах, относящихся к редактированию приема.

Если при просмотре нормализаторов приема нажать клавишу Ctr-Del, то произойдет удаление текущей цепочки нормализаторов - как и выше, без изменения записи приема в файлах до тех пор, пока не будет нажата F3 либо F4.

Просмотр текстформульного шаблона, используемого для формирования поясняющего срабатывание приема текста

Для входа в просмотр и редактирование текстформульного шаблона, используемого при отображении срабатывания приема в протоколе решения, нажимается клавиша "6". После этого в области, расположенной под нижней горизонтальной линий четвертого окна, отображается текст ранее введенного текстформульного шаблона (если он имелся) и включается обычный режим текстформульного редактора.

11.1.4 Получение справочной информации при просмотре приема

Чтобы получить справочную информацию о том или ином логическом символе, встречающемся в третьем либо четвертом окне описания приема, достаточно выделить этот символ цветовой указкой (с помощью клавиатуры либо поместив на него курсор мыши и нажав левую клавишу) и далее нажать правую клавишу мыши. Вместо последнего действия можно нажать: в случае третьего окна - Ctr-у (здесь у - кириллица),а в случае четвертого окна - Ctr-х (х - кир.). Справочный текст появится под нижней линией четвертого окна; если для него не хватило места, то следует убрать одно или несколько неиспользуемых окон с помощью Ctr-Fi; i = 1,2,3,4 - как указано выше.

Иногда для одного и того же логического символа имеется несколько различных

справочных текстов, соответствующих различным случаям его использования. Эти тексты могут быть двух типов - пояснения к фильтрам и пояснения к указателям приема. Переходить от одного такого текста к другому внутри текстов одного типа, после выдачи на экран некоторого из них, можно с помощью клавиш "курсор вверх - курсор вниз". Переход к текстам следующего типа - нажатием любой другой клавиши (например, "пробел") либо кнопкой мыши. При использовании клавиш Ctr-y, Ctr-x такого рода неоднозначность уменьшается, так как в первом случае выдаются только тексты, связанные с фильтрами приема, а во втором случае - только тексты, связанные с указателями.

Справочный текст убирается нажатием любой не используемой при его просмотре клавиши (например, клавиши "пробел") либо нажатием клавиши мыши вне окон приема.

Чтобы получить во время создания нового приема необходимую информацию о конструкциях, используемых в ГЕНОЛОГе, нужно выйти в исходную ситуацию просмотра приема либо нажать клавишу F1 (тогда появится оглавление общего описания системы), либо нажать одну из клавиш Ctr-з (оглавление заголовков приемов), Ctr-y (оглавление фильтров приемов), Ctr-x (оглавление указателей приемов).

Чтобы при просмотре приема получить доступ к информационным текстам, связанным с заданным логическим символом, нужно нажать клавишу "c". Тогда появляется курсор текстового редактора, с помощью которого набирается название данного символа. После набора символа нажимается Enter, и на экране появляется первая страница указанных информационных текстов. Эти тексты - те же самые, что и просматриваемые из редактора программ ЛОСа. При нажатии любой не используемой в интерфейсе просмотра информационных текстов клавиши - возвращение в просмотр приема.

Начато создание сводного оглавления логического языка решателя. С его помощью можно будет получать информацию о логических символах, встречающихся в теоремах приемов. Вызов оглавления из просмотра приема осуществляется нажатием клавиши "Ctr-я". Если выделить подфрагмент теоремы приема в режиме формульного просмотра и нажать правую клавишу мыши, то внизу появится справочная информация относительно заголовка подфрагмента, при условии, что она уже имеется в данном оглавлении.

С приемом можно связывать набираемые текстовым редактором примечания, облегчающие регулировку системы приемов "вручную". Для входа в просмотр таких примечаний следует нажать клавишу "э". Если в примечания нужно внести изменения, то после этого нажимается Enter (возникает курсор текстового редактора), Нажатие Enter по окончании редактирования приводит к немедленному сохранению изменений в примечаниях.

11.1.5 Указатели на степень готовности приема

Если прием только что введен либо изменен, но изменения не сохранены в файлах, хранящих описания приемов, то нижняя линия четвертого окна имеет голубой цвет. В этой ситуации нельзя выходить куда-либо из просмотра приема - все изменения (а в случае нового приема и сам прием) будут утеряны.

Если прием (новый либо после изменения старого) сохранен в файлах, хранящих описания приемов, но не откомпилирован в программу ЛОСа, то нижняя линия четвертого окна имеет красный цвет. В этом случае можно выходить из редактора приемов в другие разделы и выключать систему, не теряя описания приема.

Наконец, если описание приема сохранено в файлах и прием оттранслирован в программу ЛОСа, то нижняя линия четвертого окна имеет черный цвет. Заметим, что если изменения в описание были внесены для уже откомпилированного приема, то при сохранении изменений в файлах без перекомпиляции приема (это достигается нажатием клавиши F4) нижняя линия четвертого окна останется черной, хотя фактически программа приема на ЛОСе не будет соответствовать новой версии описания приема. Чтобы не создавать такого рода рассогласований, нужно либо использовать при изменениях приема клавишу F3 сохранения с перекомпиляцией, либо изменять отключенный прием с удаленной ЛОСовской программой (она удаляется нажатием клавиши F7).

11.1.6 Переход от просмотра описания приема к просмотру других разделов системы

От описания откомпилированного приема можно перейти к просмотру его программы на ЛОСе, нажав клавишу Home. Далее можно произвольным образом перемещаться по фрагментам ЛОСовской программы (не выходя в главное меню системы) - из любой текущей позиции для возвращения в просмотр приема достаточно нажать клавишу End.

Для быстрого возвращения в главное меню системы из просмотра описания приема предусмотрена клавиша End. При этом в оглавлении базы приемов будет сохранен путь, ведущий к просматриваемому приему, и при возвращении в это оглавление будет выделен синим цветом содержащий данный прием концевой пункт. Если в концевом пункте было несколько приемов, то при входе в просмотр данного пункта будет выдан прием, из которого в последний раз был предпринят выход по End.

Если несколько приемов созданы на основе одного и того же представления теоремы в файлах (такое представление называется узлом теоремы в базе приемов), то для перехода от одного из них к другим служат клавиши PageDown, PageUp. Впрочем, в решателе крайне редко один и тот же узел теоремы обслуживает несколько различных приемов; даже если теоремы двух приемов совпадают, они, как правило, имеют различные узлы.

Возможен прямой переход от просмотра приема к оглавлению программ либо к оглавлению задачника. В первом случае нажимается Shift-1; во втором случае Shift-3.

Для возвращения в данный прием из просмотра другого приема предусмотрен буфер перехода. Чтобы занести в него ссылку на прием, достаточно при просмотре приема нажать клавишу "минус". Для возвращения к приему из просмотра другого приема нажимается клавиша "равно". Одновременно последний прием сам заносится в буфер перехода, так что еще одно нажатие "равно" восстановит ситуацию.

Предусмотрена возможность связывать прием с той теоремой из базы теорем, на основе которой он создан. Это можно делать не для всех приемов - некоторые из них основаны на псевдотеоремах, не имеющих ничего общего с теоремами предметной области и представляющих собой чисто технические записи. Однако, большинство приемов действительно имеют своим источником конкретную "обычную" теорему. В этих случаях теорема приема возникает из соответствующей ей теоремы предметной области несложными преобразованиями, в настоящей версии системы автоматизированными лишь частично. Имеется интерфейс для установления соответствия между теоремой приема и ее прототипом в базе теорем вручную. Установление такого соответствия является одним из первых шагов в расшифровке процедур автоматического синтеза приемов, дающим исходный материал для анализа. Чтобы перейти от про-

смотра приема к просмотру той теоремы в базе теорем, которая служит источником приема, служит клавиша F9. Если источник еще не был выбран, то после нажатия F9 возникает оглавление базы теорем, в котором можно выбрать нужный источник - войти в просмотр теоремы и нажать "И".

11.2 Редактирование приемов

11.2.1 Ввод нового приема

Инициализация приема и ввод его теоремы

Для ввода нового приема следует прежде всего найти в оглавлении базы приемов тот концевой пункт, в котором предполагается зарегистрировать прием, либо создать такой концевой пункт. Затем следует войти в просмотр данного концевого пункта (клавиша "курсор вправо"). Если в пункте уже имелись какие-либо приемы, то на экране будет отображен один из них, и для создания нового приема нужно нажать клавишу Ctr-п. В случае пустого концевого пункта клавиша Ctr-п не нажимается. Далее в обоих случаях начинается заполнение окон нового приема.

Если прием должен сопровождаться чертежем, то прежде всего нажимается клавиша Ctr-ч, которая вводит в геометрический редактор. По окончании набора чертежа нажимается Enter, если теорема будет набираться формульным редактором, и Ctr-Enter, если ее нужно набрать текстовым редактором. Вслед за этим возникнет курсор формульного либо текстового редактора для набора теоремы приема (чертеж на экране сохраняется).

Если чертеж не создается, то для начала ввода теоремы приема следует либо нажать клавишу Ctr-ф (тогда теорема будет вводиться формульным редактором), либо нажать клавишу Ctr-т (тогда теорема будет вводиться в скобочной записи текстовым редактором; здесь "т" - кириллица). По окончании ввода теоремы нажимается Enter - под теоремой прорисовывается горизонтальная линия. Если теорема вводилась текстовым редактором, но возможна ее прорисовка в формульном виде, то эта прорисовка осуществляется вместо скобочной записи. В левом верхнем углу теоремы появляется номер окна - цифра 1.

Если теорема выдана в формульном виде, то непосредственно под ней размещается выделенная голубым цветом строка обозначений переменных - пар вида (x - хn), где x - обозначение переменной в формульной записи, а хn - обозначение этой же переменной во внутреннем скобочном представлении. Эти обозначения необходимы для ввода фильтров и указателей приема, так как в них используется только скобочное представление термов.

После набора теоремы под ней проводится нижняя горизонтальная линия. Далее возможен ручной ввод описания приема либо автоматическое создание его исходной версии. В первом случае следует нажать Enter - после этого появляются цифра "2" - номер второго окна приема, а также курсор текстового редактора для ввода заголовка приема. Во втором случае, для входа в оглавление типов приемов, нажимается "а" (кир.). Заметим, что пока автоматическое создание приема используется только для приемов стандартных справочников, хотя существующие процедуры могут предлагать какие-то исходные версии приема и в других случаях.

Интерфейс ручного ввода нового приема устроен так, что по окончании ввода каждого из окон 1 - 3 появляется приглашение для ввода следующего окна (в левом углу прорисовывается его номер, и возникает курсор текстового редактора для набо-

ра содержимого окна). Этот режим, однако, можно в любой момент прервать, нажав клавишу Esc. Уже набранные к этому моменту окна приема будут сохранены на экране (но пока не в файлах, так что выход из просмотра приема обратно в оглавление либо смена приема клавишами "курсор вверх - куроср вниз" ПРИВЕДУТ К ПОТЕРЕ набранной его части). Для возобновления набора приема далее нужно поступать так же, как при изменении ранее введенного приема - нажать клавишу Ctr-i, где i - номер того окна, в котором предполагается продолжать набор приема; $i = 2, 3, 4$.

Ввод заголовка приема

Заголовок приема набирается текстовым редактором обычным образом. Если для выбора заголовка здесь понадобится справочная информация, то ее можно получить, нажав клавишу Ctr-з. Тогда появляется оглавление типов заголовков приемов. Выбрав нужный концевой пункт в этом оглавлении, следует нажать клавишу "курсор вправо" - текст шаблона заголовка приема будет прорисован во втором окне автоматически. В нижней части экрана появится текстовое пояснение к этому шаблону. После преобразования шаблона заголовка в заголовок нажимается Enter. Автоматически появляется приглашение к набору 3-го окна.

Ввод фильтров и указателей приема

Третье и четвертое окна (соответственно, фильтры и указатели) набираются текстовым редактором. Если при наборе третьего окна нажать клавишу Ctr-у (кир.), то появляется оглавление фильтров; если при наборе четвертого окна нажать клавишу Ctr-х, то появляется оглавление указателей. При выборе нужного концевого пункта одного из этих оглавлений и нажатии на нем клавиши "курсор вправо" произойдет прорисовка шаблона соответствующего фильтра либо указателя начиная с той позиции окна, на которой находился курсор при обращении к оглавлению (содержимое окна при выходе из оглавления восстанавливается). По окончании набора окна нажимается Enter.

Если при наборе третьего либо четвертого окна появляется необходимость ввести указатель вхождения в теорему, то нажимается Shift-1. Вслед за этим в первом окне включается режим выбора подтерма теоремы цветовой указкой. Выбранный подтерм прорисовывается малиновым цветом в случае формульной прорисовки теоремы либо, при скобочной прорисовке, выделяется группой цветов, как в редакторе программ ЛОСа. После выбора подтерма, расположенного на нужном вхождении, нажимается "ф". Тогда, начиная с той позиции, на которой во время нажатия Shift-1 находился курсор, прорисовывается указатель вхождения "фикс(...)". Если при просмотре подтермов стало ясно, что указатель вхождения не нужен, то нажимается Esc. Следует заметить, что в особых случаях формульный редактор не позволяет непосредственно выходить на цветовое выделение части подтермов теоремы; тогда для автоматического набора указателя вхождения рекомендуется нажатием Enter выйти из редактирования окна, поменять формульный режим отображения теоремы на текстовый, и снова войти в редактирование окна.

По окончании набора предварительной версии приема нажимается клавиша F4. Это приводит к сохранению в файлах набранной части приема. Далее можно начинать цикл коррекций исходной заготовки приема, присоединяя к ней необходимые фильтры, указатели, нормализаторы и т.п.

Ввод нормализаторов приемов

Для ввода нормализаторов приема необходимо выделить цветовой указкой в теореме либо в некотором терме третьего или четвертого окон нормализуемый терм. Если после этого нажать Enter, то под нижней линией четвертого окна появляется курсор текстового редактора. Если для выделенного подтерма уже были введены нормализаторы, то они будут прорисованы начиная с позиции курсора. Далее предпринимается ввод либо изменение нормализаторов и нажимается Enter (в случае отказа от редактирования - Esc). После этого восстанавливается режим цветовой указки, и можно переходить к выбору нового нормализуемого подтерма.

Как и в случае автоматического набора указателя вхождения, следует учитывать, что не во всех случаях возможно корректное выделение подтерма теоремы в режиме формульного редактора. В тех случаях, когда оно не обеспечено, следует переходить к текстовому режиму отображения теоремы.

Группа нормализаторов приема, связанная с обработкой некоторого подтерма, вводится лишь однократно - по произвольному вхождению этого подтерма в теорему, фильтр либо указатель.

Иногда бывает нужно связывать группу нормализаторов приема с термом A, встречающимся в других нормализаторах приема (например, в дополнительных посылках либо в фильтрах, встречающихся в нормализаторах приема). Если такие термы не встречаются в 1,3,4 окнах, то в 4-м окне вводится фиктивный указатель "фикс(A)", и после этого для A указывается цепочка нормализаторов. Заметим, что данный фиктивный указатель при сохранении приема в файлах отбрасывается, и при последующих просмотрах приема (если его не ввести снова) появляться не будет.

Как и для любых изменений приема, после коррекции нормализаторов нужно нажать клавишу F4 для сохранения изменений в файлах либо клавишу F3 для сохранения изменений и перекомпиляции.

11.2.2 Сохранение описания приема и компиляция

По окончании ввода приема следует по крайней мере сохранить его описание в файлах системы (оно сохраняется в 8-м информационном блоке). Для этого нажимается клавиша F4. Если введенная версия приема не требует какой-либо доработки, то вместо F4 можно нажать F3 - тогда кроме сохранения описания приема сразу же будет выполнена компиляция его в программу на ЛОСе.

В целях отладки иногда бывает нужно временно удалять программу приема, сохраняя его описание в файлах. Для этого используется клавиша F7. Чтобы снова откомпилировать программу такого приема, нажимается F5.

Для отладки компилятора ГЕНОЛОГа предусмотрен вход в компиляцию с установкой прерывания через оглавление программ ЛОСа. Здесь используется клавиша Ctr-F5, после нажатия которой возникает некоторый пункт оглавления программ. В этом оглавлении нужно войти в подраздел "Компилятор ГЕНОЛОГа"; выбрать нужный концевой пункт и нажать на нем "курсор вправо". Как только при компиляции будет достигнут соответствующий данному концевому пункту оператор "прием(N)", будет включен пошаговый режим отладчика ЛОСа.

11.2.3 Изменение приема

Чтобы изменить теорему приема, обычно применяется режим текстового редактора, так как в нем можно скорректировать произвольный подфрагмент теоремы, не переписывая заново всей теоремы. Для входа в такой режим достаточно нажать клавишу Ctr-т (кир.). Разумеется, для работы в текстовом редакторе следует заблаговременно уточнить обозначения теоремных переменных во внутреннем "скобочном" представлении, используя располагаемый под "формульной" записью теоремы выделенный голубым цветом список таких обозначений. По окончании редактирования нажимается Enter, после чего (если только не был специально установлен текстовый режим просмотра) измененная теорема будет перерисована в формульном виде. Как и во всех случаях изменения приема, для сохранения изменений здесь следует нажать F4 либо (для одновременной перекомпиляции) F3.

Для изменения теоремы приема формульным редактором имеется несколько возможностей. Первая из них - войти в набор теоремы "с конца" (нажатие клавиши "Ф" из общего просмотра приема), отбросить с помощью Backspace старую версию концевой части, и затем перенабрать эту часть в требуемом виде. Другая возможность - выделить курсорами либо мышью подтерм теоремы, который нужно изменить, нажать "Ф", ввести формульным редактором новую версию, и нажать "Enter". Если нужно добавить новый антецедент теоремы, то выделяется тот антецедент, перед которым выполняется вставка, нажимается "в", и формульным редактором вводится добавляемый антецедент. Если выделить антецедент и нажать "В", то вставка нового антецедента будет осуществлена непосредственно после выделенного. Если выделить антецедент и нажать "Ctr-Del", то он будет удален. Перечисленные операции можно выполнять не только с антецедентами, но также с операндами любой коммутативно-ассоциативной операции, встречающейся в теореме. Наконец, упомянем совсем радикальный способ изменения теоремы приема - полный перенабор ее формульным редактором. Для этого нажимается клавиша Ctr-ф, вводящая в режим формульного редактора. Для удобства во время набора новой версии старая версия теоремы сохраняется в верхней части экрана - кроме случаев, когда места для обеих версий сразу недостаточно.

Для коррекции чертежа - как и для ввода нового чертежа - нажимается клавиша Ctr-ч, вводящая в геометрический редактор.

Если нужно изменить содержимое окна с номером 2,3 либо 4, то нажимается, соответственно, клавиша Ctr-2, Ctr-3 либо Ctr-4. По завершении редактирования нажимается Enter.

Для изменения нормализаторов, связанных с некоторым подтермом теоремы, фильтра либо указателя приема, следует поступать так же, как для ввода новых нормализаторов - выделить этот подтерм цветовой указкой и нажать Enter. Для удаления ранее введенных нормализаторов следует войти в просмотр всего их списка (Ctr-5, и далее PageDown - PageUp), выделить ненужную группу нормализаторов и нажать Ctr-Del. Из просмотра списка нормализаторов, инициируемого нажатием клавиши Ctr-5, можно также изменять нормализаторы. Для этого на текущей просматриваемой группе нормализаторов нажимается "н". Наконец, для удаления вообще всех нормализаторов данного приема нажимается Ctr-End. Последняя операция бывает необходима, если была существенно изменена теорема или удалены некоторые фильтры либо указатели, с подтермами которых были связаны нормализаторы (в этих случаях попытка просмотра нормализаторов с помощью Ctr-5 может приводить к сообщениям о некорректном обращении к операторам ЛОСа). Еще раз напо-

минаем о необходимости использовать F4 либо F3 для сохранения изменений.

11.2.4 Буфер базы приемов

Чтобы можно было контролировать вводимые в приемы изменения, используется буфер базы приемов. Этот буфер создается автоматически; в нем регистрируются новые, измененные и удаленные приемы. Каждый раз при его создании в корневом меню оглавления базы приемов добавляется последним пунктом подменю "Буфер". В этом подменю вводятся далее подменю "Новые приемы", "Измененные приемы" и "Удаленные приемы" (в зависимости от наличия таковых). При добавлении, изменении либо удалении приема в соответствующем (одном из указанных трех) подменю вводится концевой пункт, подзаголовок которого копируется с подзаголовка соответствующего (содержащего либо содержавшего прием) концевого пункта основной части оглавления. Через этот концевой пункт происходит выход в просмотр старой (в случае нового приема - основной) версии приема.

Для перехода от просмотра в буфере старой версии к просмотру новой версии используется клавиша "о"; для возвращения от просмотра основной версии к просмотру старой версии - клавиша "б".

При просмотре версии приема, находящейся в буфере, не рекомендуется вносить какие-либо изменения в описание приема (!) - это может привести к необходимости "ремонта" оглавления базы приемов. Однако, при просмотре версии в буфере возможна ее компиляция либо уничтожение ее программы - как и обычно, с помощью клавиш F5 и F7. Указанный выше ремонт оглавления состоит в том, чтобы через технический просмотр 8-го информационного блока (см. подраздел "Ресурсы и установки"главного меню системы) выйти в корень оглавления базы приемов (метки "оглавление", "прием") и изменить содержимое логического терминала "позиция" на "число(0 0)". После выхода из технического просмотра - вернуться в просмотр оглавления приемов и сразу же нажать в нем клавишу "О" (кир.) для сброса непригодного далее буфера.

Если необходимо для целей отладки временно откомпилировать старую версию приема, удалив программу новой версии, то из просмотра новой версии приема нажимается клавиша "И". Обратно, для удаления программы старой версии и компиляции программы новой (снова из просмотра новой версии) нажимается клавиша "Т". Наконец, если новая версия не оправдала надежд и нужно вернуться к старой версии, нажимается "Б".

Время от времени буфер следует расчищать: если в нем накопилось слишком много приемов, то будут заблокированы изменение приемов, ввод новых приемов и удаление старых. Расчистка буфера происходит при нажатии в любом месте оглавления базы приемов клавиши "О" (если это происходило в момент просмотра пунктов буфера, то произойдет выход в главное меню, иначе - сохранится просмотр оглавления), Если после расчистки буфера и возвращения из главного меню в оглавление базы приемов экран оказался пустым, то это означает, что пункты оглавления оказались "прокручены вверх". Для их возвращения используется PageUp.

Для удобства работы с буфером предусмотрен автоматический переход в него из любого места оглавления базы приемов - по нажатию клавиши "ф".

11.2.5 Перенесение приема в другой концевой пункт оглавления

Чтобы перенести прием в другой концевой пункт оглавления базы приемов, следует войти в его просмотр, нажать Insert, выйти обратно в оглавление, перейти к нужному концевому пункту и (не входя в него) снова нажать Insert. Перед вторым нажатием Insert следует убедиться, что необходимый концевой пункт действительно выделен в качестве текущего (т.е. имеет голубой цвет).

11.2.6 Удаление приема

Для удаления ненужного приема нажимается клавиша Ctr-Del. Заметим, что в этом случае прием сохраняется в буфере базы приемов и при необходимости может быть оттуда извлечен обычной процедурой перенесения приема из одного пункта оглавления в другой пункт: на переносимом приеме нажимается Insert, далее по оглавлению базы приемов находится нужный концевой пункт и (без входа в этот пункт, но после выделения его голубым цветом) снова нажимается Insert. После расчистки буфера (нажатием клавиши "О") происходит окончательное удаление ранее исключенных приемов.

Особой осторожности требует удаление приемов пакетных нормализаторов. Если удаляется или даже просто перекомпилируется единственный оставшийся прием такого оператора, то происходит нарушение архитектуры его программы, для восстановления которой нужно обязательно (!) удалить клавишей F7 и затем снова откомпилировать прием "окончание" этого нормализатора, играющий роль переключателя уровней срабатывания. Лишь затем можно подключать к программе этого оператора другие приемы - вводя новые или компилируя временно отключенные старые. Если нормализатор имеет более одного приема, то удаление приема и перекомпиляция возможны без ограничений.

11.2.7 Автоматическое пополнение описания приема

Процедуры, созданные для автоматического синтеза приемов на ГЕНОЛОГе, имеют пока предварительный характер, однако даже и в таком виде они могут существенно ускорить процесс ввода элементов описания приема и подсказать полезные добавления к этому описанию.

Если после ввода теоремы нажать "а", то возникает оглавление типов приемов. Фактически это ветвь оглавления программ для подраздела "Справочник ЗАГОЛОВОКПРИЕМА" раздела "Генератор приемов". Переходя в этом оглавлении к концевому пункту (точка после номера пункта) для приемов требуемого типа и заходя в этот пункт, инициируем работу генератора приемов. Если был создан хотя бы один прием, то на экране появится описание первого из таких приемов, ограниченное снизу голубой линией. Это описание, после необходимой коррекции, можно сохранить в базе приемов (F4) или даже сразу откомпилировать (F3). Если было создано несколько приемов, то каждый следующий прием из серии вызывается на экран нажатием клавиши "ш"; его также можно сохранить либо откомпилировать. По исчерпании серии, нажатия клавиши "ш" каких-либо изменений на экране вызывать не будут. Заметим, что данный режим создания приемов пока применяется исключительно для приемов справочников - согласно первом пункту "Справочники и сопровождающие их простейшие приемы" оглавления типов приемов.

Значительно более употребительным (практически постоянно) является автоматическое пополнение списков указателей и нормализаторов приема. Для такого пополнения нажимается клавиша "н". Если процедура предлагает ввести новый элемент либо удалить старый, то он прорисовывается под нижней линией четвертого окна. В случае, если этот элемент является нормализатором приема, одновременно выделяется цветовой указкой нормализуемый им подтерм. Нажатием клавиши Insert можно ввести предлагаемый элемент в описание приема, нажатием клавиши Delete - отказаться от его ввода. Если процедура предлагает добавить новый элемент, то в левой части прорисовываемого вверху экрана меню написано "Новый указатель"; если она предлагает удалить старый элемент, то в левой части прорисовываемого вверху экрана меню написано "Старый указатель". В последнем случае нажатие Delete приведет к удалению элемента, а нажатие Insert - к его сохранению. Предложения выдаются последовательно, по мере нажатия клавиш Insert - Delete. Этот процесс можно в любой момент оборвать, нажав Esc. Введенные до этого изменения сохранятся (но не в файлах, а только в текущем отображении приема).

Заметим, что, как правило, предложения по удалению старых указателей вызваны не какими-то специальными мотивировками, а просто тем, что процедура не генерирует именно эти типы указателей - они находятся вне ее компетенции. Автоматическое создание обращений к нормализаторам общей стандартизации существенно облегчает ручной ввод приемов - ведь такими указателями обычно снабжаются практически все отличные от переменных и констант подвыражения формируемых приемом термов.

11.3 Дополнительные возможности редактора приемов

11.3.1 Разрезание окна на несколько частей либо склейка частей окна

Если третье либо четвертое окно становится слишком большим и затрудняет работу с приемом (например, не оставляет места для просмотра нормализаторов), то его можно разрезать на несколько частей. Для этого нужно войти в просмотр термов окна цветовой указкой, выбрать тот терм, который будет последним в первой половине разрезаемого окна, и нажать клавишу "р" (кир.). Другой способ разрезать окно - войти в режим его редактирования и поместить запятую в точке разрезания.

Чтобы склеить текущую часть окна с непосредственно следующей за ней, нужно войти в просмотр термов окна цветовой указкой и нажать F6. Другой способ - войти в редактирование окна и убрать запятую в его конце.

11.3.2 Копирование приема либо его фрагментов

При программировании на ГЕНОЛОГе иногда возникают серии похожих приемов, для ускоренного ввода которых можно применять копирование первого приема серии с последующими исправлениями.

Для полного копирования приема имеются две возможности - с регистрацией в файлах и компиляцией для вводимой копии, либо без регистрации и компиляции. Во втором случае нажимается клавиша Ctr-д, после чего возникает заготовка для редактирования альтернативной версии приема (старая версия приема сохраняется;

под новой версией проведена голубая линия). Эту версию приема после завершения редактирования можно сохранить и откомпилировать обычным образом (F4 либо F3). В первом случае нажимается клавиша "д", и созданная копия приема сразу же сохраняется и компилируется. Эта возможность, в отличие от предыдущей, практически не применяется.

Иногда бывает нужно многократно использовать в различных приемах одни и те же фильтры. Чтобы ускорить их ввод, можно выделить такой фильтр цветовой указкой и нажать клавишу "у". Если затем войти в просмотр другого приема и, не входя в режим цветовой указки, нажать "у" снова, то данный фильтр (а если указанным образом ранее было отобрано несколько фильтров, то все они сразу) будет занесен в третье окно приема. Далее нужно сохранить введенное таким образом изменение, нажав F4 либо F3. Чтобы сбросить накопитель фильтров, вводимых в приемы по нажатии клавиши "у", используется клавиша "У".

Вообще, для перенесения каких-либо фрагментов из приема в прием можно использовать обычный буфер текстового редактора: сначала нужно войти в режим редактирования текстовым редактором того окна, которое содержит копируемый фрагмент (в том числе первого окна); затем занести этот фрагмент в буфер нажатием PageDown в начале и конце фрагмента; перейти в другой прием либо в другое окно того же самого приема, и извлечь фрагмент из буфера нажатием PageUp.

Можно создавать различные приемы, используя одну и ту же теорему. Для этого сначала нужно войти в прием, имеющий данную теорему, и для занесения ее в специальный буфер нажать Ctr-б. Далее, при создании нового приема можно начинать цикл его ввода с нажатия Ctr-и либо Ctr-к (кир.). В обоих случаях будет прорисована отобранная в буфер теорема. В первом случае прием будет сохранен в файлах на основе уже имеющейся в них записи теоремы - "узла теоремы"; во втором - будет введен новый узел той же теоремы. Вторая возможность используется, если с первым узлом уже был связан прием, имеющий тот же самый заголовок: выход из узла теоремы на структуру данных приема по заголовку этого приема должен происходить однозначным образом.

11.3.3 Изменение логического символа, за которым закреплен прием

Иногда бывает удобно изменить логический символ, за которым первоначально был закреплен прием (например, прием был создан на основе копии аналогичного приема, относившегося к другому понятию). В этом случае нажимается клавиша Ctr-й; вводится текстовым редактором необходимый символ, и нажимается Enter. Произойдет закрепление приема за новым символом, перенесение его описания в файлах на новое место и перекомпиляция.

11.3.4 Поиск приемов заданного вида в базе приемов

Для быстрого просмотра всех приемов базы приемов ГЕНОЛОГа, удовлетворяющих заданному условию, предусмотрен специальный режим, использующий оператор ЛОСа "текприем(...)". Задание на отбор приемов формулируется в виде ЛОСовской программы этого оператора. Все, что нужно для этого сделать - разместить после начального оператора "метка(икс(десять))" этой программы и до последнего ее оператора "ответ(набор(См))" систему условий на отбираемые для просмотра приемы. В описании оператора "текприем(...)" уточняется, какую информацию о текущем

приеме несут различные входные переменные оператора "текприем(...)". Основные из них: x3 - теорема приема; x4 - заголовок приема; x5 - описание приема. Описание приема представляет собой набор указателей приема, к которому добавлен терм "условие(и($A_1 \dots A_n$))", где $A_1, \dots, A_n$ - все фильтры приемы, а также добавлены термы "быстрпреобр($B_1 \dots B_m$)", определяющие нормализаторы приема (B_1 определяет нормализуемый подтерм - прямо либо через указатель вхождения "фикс(...)"; $B_2, \dots, B_m$ суть нормализаторы приема, связанные с этим подтермом).

После того, как установлена необходимая программа оператора "текприем(...)", следует выйти из редактора программ ЛОСа и перейти в произвольный раздел оглавления базы приемов. Далее нажатие клавиши F8 запускает цикл просмотра всех (вне зависимости от раздела, где была нажата F8) приемов системы, удовлетворяющих заданному ограничению. Чтобы перейти от текущего приема к просмотру следующего, нажимается клавиша "курсор влево". Чтобы оборвать просмотр, нажимается Esc, переводящее в главное меню. После этого можно вернуться в оглавление базы приемов - текущими окажутся те конечный пункт оглавления и прием этого пункта, на котором был оборван просмотр. Если в процессе просмотра приемы исчерпаны либо их вообще не было, то система выходит в отладчик ЛОСа на оператор "трассировка(стоп 0)" - далее нажатие Esc возвратит ее в главное меню. Заметим, что оператор "текприем(...)" можно запрограммировать так, чтобы он не только отбирал нужные приемы для просмотра, но и вносил в них те или иные изменения, с перекомпиляцией или без нее. Таким образом, появляется возможность с разумными затратами сил осуществлять при необходимости различные глобальные преобразования всей базы приемов. Отметим также, что изменять просматриваемые в указанном выше цикле приемы можно и обычным образом, вручную. Единственное ограничение здесь - невозможность проконтролировать результаты перекомпиляции путем перехода к редактору программ ЛОСа (клавиша Home). Чтобы изменить найденный прием с возможностью такого контроля, следует оборвать на нем просмотр с помощью Esc, а затем снова зайти в базу приемов и выполнить редактирование приема.

При регулировке решателя те или иные приемы иногда отключаются с помощью клавиши F7. Чтобы выявить все такие отключенные (не откомпилированные) приемы базы приемов, следует нажать клавишу Ctr-F8 и далее действовать, как в указанном выше цикле просмотра (переход к следующему отключенному приему - "курсор влево"; обрыв цикла просмотра - Esc).

11.3.5 Перенесение приемов из версии системы, находящейся в директории ALT

Чтобы переносить приемы ГЕНОЛОГа из одной версии системы в другую, следует разместить ту версию, приемы которой будут копироваться, в поддиректории ALT рабочей директории той версии, в которую нужно скопировать приемы. После запуска последней версии и входа в просмотр какого - либо ее приема следует нажать клавишу Ctr-a (кир.). Произойдет переход в просмотр оглавления приемов версии из поддиректории ALT. В этой версии следует выбрать представляющий интерес прием и, войдя в его просмотр, нажать Ctr-o (кир.). Произойдет возвращение в оглавление приемов основной версии. Далее нужно войти в тот концевой пункт оглавления, куда должен быть перенесен прием (или создать новый такой пункт). Войдя в него, следует нажать Ctr-ъ. Появится описание скопированного приема. Далее его нужно сохранить обычным образом - нажав клавишу F4 либо F3.

Имеется также другая возможность перенесения из одной версии в другую эле-

ментов логической системы - через контейнер, роль которого играет 12-й информационный блок (файлы $Cijkl$). Контейнер заполняется в одной версии системы, затем его файлы переносятся в другую версию, и в ней извлекается их содержимое.

11.4 Анализ применений приема на обучающем материале

Почти любой новый прием, заносимый в решатель, требует уточнений, выявляемых в процессе прогонки решателя по тем или иным разделам задачника. Для такой прогонки после ввода приема предпринимается выбор необходимого раздела в оглавлении задачника и нажатие Ctr-з в корневом меню данного раздела. По окончании цикла решения задач данного раздела нажимается клавиша "з" и анализируется статистика процессов решения - так, как об этом говорилось выше в разделе "Запуск решения задачи и его пошаговый просмотр" 4-й главы. В первую очередь, здесь выявляются задачи, ответ которых изменился, либо решение которых существенно замедлилось, либо которые вообще перестали решаться после ввода приема или группы новых приемов. Каждая такая задача далее анализируется отдельно. Если была введена либо изменена группа приемов, то с помощью буфера базы приемов приемы этой группы последовательно отключаются либо восстанавливаются в своем исходном виде. Это позволяет выявить конкретного виновника замедления, ошибки либо отказа. Далее устанавливается прерывание при входе в найденный таким образом прием, и на уровне отладчика ЛОСа анализируется его работа в процессе решения задачи. Для более надежного выявления моментов срабатывания приема рекомендуется перекомпилировать его, введя в четвертое окно указатель "стоп" - тогда выход в отладчик ЛОСа будет происходить непосредственно перед реализацией преобразований приема.

На основе рассмотрения выявленных указанным способом особых случаев срабатывания приемов в группе задач происходит выработка новых его эвристических решающих правил. Чтобы впоследствии можно было быстро находить все те задачи задачника, в которых прием сработал хотя бы однажды (это может понадобиться при проверке того, что усиление фильтров приема не нарушило процессов решения таких задач), информация о данных задачах автоматически сохраняется при запуске очередной прогонки в специальном информационном блоке - архиве задачника. Впоследствии можно получить список таких задач, расположенных в заданном разделе задачника, используя следующую процедуру:

1. Выделить в некотором меню задачника тот пункт, который вводит в требуемый раздел;
2. Нажать клавишу Shift-2 для перехода в оглавление базы приемов;
3. Войти в просмотр нужного приема и нажать клавишу "А" (кир.);
4. Нажать клавишу Shift-3 для перехода в оглавление задачника;
5. Нажать клавишу "ф" для перехода в буфер задачника.

В буфере задачника будет создано новое подменю, заголовок которого копирует текст концевого пункта базы приемов, к которому относился прием. В этом подменю размещаются дубликаты указанных выше задач (регистрируются не более 50 первых

задач - в случае большего числа задач нужно по отдельности анализировать подразделы рассматриваемого раздела). Из просмотра дубликата задачи можно перейти к просмотру оригинала нажатием клавиши "б". Однако, все тестовые запуски (в том числе групповой запуск для данного подменю буфера) можно выполнять непосредственно на дубликатах. В действительности здесь дублируются не сами задачи, а только ссылки на них из оглавления. Поэтому данные об измененных результатах решения задачи будут просматриваться одновременно из буфера и из того раздела оглавления, в котором расположена основная версия задачи.

При вводе нового приема можно сокращать прогонку по задачнику, отбирая лишь те задачи, при решении которых возникают логические символы, необходимые для срабатывания этого приема. Для организации такой сокращенной прогонки следует применять следующую процедуру:

1. Войти в просмотр приема и нажать клавишу "С";

2. Нажать Shift-F3 для перехода в оглавление задачника (заметим, что в этот момент в буфере задачника будет находиться пустой подраздел, заголовок которого копирует текст концевого пункта базы приемов, содержащего анализируемый прием; в этом подразделе при прогонке будут накапливаться дубликаты задач, в которых прием сработал хотя бы однажды;

3. Выбрать нужный раздел задачника и нажать "З".

Начнется сокращенный цикл решения задач раздела; по окончании его выполняется обычный анализ статистики.

Для оптимизации решателя предусмотрена возможность определения "холостого хода" приема на заданном разделе задачника - средней трудоемкости попыток применения приема на одно его срабатывание. Чтобы получить такую характеристику, следует запустить цикл решения задач из выбранного раздела задачника не клавишей Ctr-з, а клавишей "а" (кир.). Процесс решения чуть-чуть замедлится, так как для приемов, в попытки применения которых система будет входить при решении задач, начнется накопление пар (суммарная трудоемкость попыток применения приема - число применений приема). Моментом входа в попытку применить прием здесь считается прохождение через оператор "контрольприема(...)", размещаемый компилятором в начале программы приема. По окончании прогонки следует войти в просмотр статистики, где будут указаны случаи наибольшей средней трудоемкости; нажатием клавиши "х" (кир.) можно перейти в режим просмотра соответствующих приемов (переход к каждому следующему приему - нажатием "ш"), упорядоченных по убыванию этой трудоемкости. Чтобы на экран была выдана пара (суммарная трудоемкость - число применений), при просмотре приема следует нажать клавишу "Х"(кир.).

Предусмотрена некоторая автоматизация анализа избыточности применений приема. Чтобы выявить те задачи раздела, где отказ от приема приводит к ускорению решения, следует прежде всего создать в буфере задачника список дубликатов задач этого раздела, в которых прием срабатывает (выбрать раздел задачника; нажать Ctr-2; выбрать прием; нажать "А" (кир.); нажать Ctr-3 для возвращения в задачник). После этого нажатие клавиши "й" инициирует цикл попыток решить отобранные в буфере задачи без данного приема. Трудоемкости решения с приемом и без него сравниваются; если задача быстрее решается без приема, то в оглавлении буфера после ее номера ставится знак вопроса.

11.5 Расширение списка символов, прорисовываемых формульным редактором

Чтобы формульный редактор мог прорисовывать тексты задач в произвольных предметных областях, приходится постоянно пополнять его словарь. Для упрощения понимания обозначений, названия новых логических символов переносятся в формульный редактор без каких-либо изменений. Однако, это происходит не автоматически - сначала нужно сопроводить новое понятие A несколькими приемами справочников формульного редактора. Все эти приемы создаются по одной и той же теореме - "формредактор(A B_1 ... B_n)". Здесь $B_1, \ldots, B_n$ - указатели, уточняющие способ прорисовки. Обязательными являются указатели "префикс" и "логсимвол". Они означают, что будет использоваться запись $A(t_1 \ldots t_m)$, причем заголовок представляет собой текст названия логического символа A. Если операция (отношение) A многоместное, причем операнды отделяются запятыми, то вводится указатель ",". Если A - многоместное, но операнды суть переменные, не разделяемые запятыми, то вместо этого берется указатель "кортеж". Если A есть символ константы, то используется указатель "конст". Наконец, если нужно обеспечить возможность набора символа двмуя нажатиями клавиш, то вводится указатель "ключ(K_1 K_2)". Здесь K_1, K_2 - логические символы, кодирующие выбранные клавиши, причем в латинском регистре. Чтобы при наборе теоремы приема текстовым редактором ввести такие символы, нажимается клавиша F12, переводящая в латинский регистр, затем нажимается F8 (режим ввода названий кодов клавиш), и далее нажимается требуемая клавиша. При переходе к очередному K_i повторное нажатие F12 не требуется, но повторное нажатие F8 необходимо. Для возвращения из латинского регистра в кириллицу, по окончании ввода символов K_i нажимается F11.

Если указатель "ключ(...)" отсутствует, то название A придется вводить формульным редактором через вспомогательное обращение к текстовому редактору (Ctr-Enter). Это не так удобно, как ввод названия двумя касаниями, однако приходится учитывать, что пар клавиш не так уж много, и вводить их лишь для сравнительно часто используемых понятий.

Как только теорема приема введена и нажата клавиша Enter, завершающая работу текстового редактора, нажимается клавиша "а" (кир.). Она переводит в оглавление типов приемов. Здесь следует выбрать первый пункт - "Справочники и сопровождающие их простейшие приемы". Далее нажимается клавиша "курсор вправо", после чего на экране появляется автоматически сгенерированный первый из необходимых приемов. Для сохранения и компиляции его нажимаем F3. Для извлечения очередного приема справочника формульного редактора нажимаем "ш". Затем - снова F3, и так до тех пор, пока генератор приемов не исчерпает новые приемы.

Если указатель "ключ(...)" не использовался, то на этом присоединение понятия A к словарю формульного редактора завершается. В противном случае нужно проверить, не была ли уже использована предложенная комбинация K_1, K_2. Для этого в серии новых приемов находим прием с заголовком "префикс(2)" и переходим к просмотру его программы (клавиша "Home"). Здесь нужно рассмотреть ту ветвь программы справочника "префикс", к которой относится текущий фрагмент программы, чтобы убедиться, что в ней уже не было другого фрагмента, содержащего оператор "равно(конец(x1) K_2)" или эквивалентный ему оператор, связывающий x1 с K_2. Иногда новый и старый фрагменты оказываются занесены в общую ветвь, идущую после указанного оператора. Если такой фрагмент был, то сочетание кла-

виш уже использовано. Тогда нужно вернуться в редактирование теоремы приема (только данного), выбрать другое сочетание клавиш, перекомпилировать прием, и снова предпринять указанную проверку.

11.6 Инициализация пакетных операторов

Перед тем, как создавать приемы нового пакетного оператора, необходимо провести определенную подготовительную работу - выбрать название оператора и создать приемы справочников, описывающих формат этого оператора. Предусмотрены два способа такой инициализации оператора - вручную либо с применением специального интерфейса.

11.6.1 Инициализация оператора вручную

Проверочный оператор

Для ввода нового проверочного оператора необходимо прежде всего выбрать (либо ввести новый) логический символ - название этого оператора. Затем вводится прием справочника "легковидеть", определяющий формат оператора (см. выше подраздел "Приемы справочников" раздела "Типы заголовков приемов"). После ввода псевдотеоремы справочника "легковидеть" рекомендуется нажать "а", выбрать первый пункт оглавления типов приемов (курсор вправо), и вводить далее серию приемов, нажимая поочередно F3 и "ш" - пока такие приемы будут появляться. Кроме справочника "легковидеть", будут также введены прием справочника "очевидно" и два простейших приема самого проверочного оператора - прием "быстрпроверка", обеспечивающий непосредственное обнаружение в контексте проверяемого утверждения, а также ряд других простейших способов проверки, и прием "контрольбуфера", предпринимающий попытку извлечь готовый результат из буфера предыдущих обращений.

Синтезатор

Для ввода нового синтезатора определяется его название, и вводятся приемы справочников "синтезатор" и "значения", определяющие формат синтезатора. После ввода псевдотеоремы справочника "синтезатор" рекомендуется нажать "а" и выбрать первый пункт оглавления типов приемов - это приведет к автоматической генерации приема данного справочника, справочника "значение" и первого приема синтезатора, осуществляющего попытку извлечь готовый результат из буфера результатов обращения к данному синтезатору. Как и в случае проверочных операторов, здесь последовательно нажимаются F3 и "ш".

Нормализатор

Для ввода нового нормализатора выбирается его название и вводится прием справочника "быстрпреобр", определяющий формат оператора. В случае нормализатора общей стандартизации термов с заданным заголовком вводятся также приемы справочников "нормализатор" и "ключоператора". Как и в предыдущих случаях, после ввода псевдотеоремы справочника "быстрпреобр" рекомендуется нажать "а" и использовать оглавление типов приемов - тогда, кроме приема данного справочника,

будет также введен первый фиктивный прием нормализатора, обеспечивающий смену текущего уровня его работы (его заголовок - "окончание"). В случае нормализатора общей стандартизации нужно не забыть ввести после этого также приемы справочников "нормализатор", "ключоператора".

Анализатор

Для ввода нового анализатора выбирается его название и вводится прием справочника "анализатор", определяющий формат оператора. Это делается с помощью оглавления типов приемов, причем одновременно создается прием для переключения текущего уровня анализатора (его заголовок - "внешвывод").

Оператор фильтра

Для ввода нового оператора фильтра выбирается его название и вводятся приемы справочников "блок", "блокпроверок", определяющие формат оператора.

11.6.2 Интерфейс ускоренной инициализации оператора

Для упрощения процесса создания нового пакетного оператора создан специальный интерфейс. Чтобы создать с его помощью оператор, выбирается тот раздел оглавления базы приемов, из которого будет сделана ссылка на ветвь данного оператора. Затем нажимается клавиша Ctr-п.

После нажатия на экране возникает диалоговый блок для ввода названия оператора и выбора его типа. Чтобы ввести название, предпринимается нажатие левой клавиши мыши внутри верхнего окна ("Название оператора"). Тогда появляется курсор текстового редактора, с помощью которого и вводится название. Это название может быть либо новым, либо старым, но не использованным ранее в качестве названия оператора ЛОСа. В случае нового названия оно сразу же регистрируется в файлах словаря. Если вводится нормализатор общей стандартизации выражений с заданным заголовком, то его название можно вообще не вводить, а сразу нажать левую кнопку мыши в окне "Нормализатор". Иначе - нажатие левой кнопки мыши в одном из окон "Нормализатор", "Проверочный оператор", "Синтезатор", "Анализатор", "Оператор фильтра" предпринимается после ввода названия. После этого нажатия появляется новый диалоговый блок. В зависимости от типа оператора, далее выполняются следующие действия:

1. Нормализатор. Если вводится нормализатор общей стандартизации выражений с заданным заголовком, то этот заголовок набирается в левом верехнем окне (предварительно нажимается клавиша "с" либо на этом окне нажимается левая кнопка мыши). Если набрано название "...", то оператор будет называться "усм...". Автоматически устанавливается ряд значений в других окнах, так что в данном случае сразу можно нажать Enter. Если же вводимый нормализатор не является нормализатором общей стандартизации выражений с заданным заголовком, то левое верхнее окно не заполняется, но уточняются прочие параметры оператора (число уровней срабатывания, наличие списка посылок, использование буфера и т.д. - все эти пункты соответствуют стандартным элементам из описания формата оператора). Далее также нажимается Enter.

2. Проверочный оператор. Здесь в верхнем окне вводится текстовым редактором в скобочной записи вид проверяемого утверждения. В окне, расположенном ниже, перечисляются формульным редактором входные переменные из шаблона проверяемого утверждения, расположенные в том порядке, в котором они будут располагаться при обращении к оператору. Далее указывается число уровней срабатывания (обычно больше 1), и нажимается Enter.

3. Синтезатор. В верхнем окне набирается одно или несколько утверждений, представляющих собой те условия, которым должны удовлетворять подбираемые синтезатором выходные значения. Затем в окнах, расположенных ниже, перечисляются входные и выходные переменные, а также указывается число уровней срабатывания. Нажатия левой клавиши мыши на окнах, указывающих возможные уточнения типа синтезатора, приводят к появлению либо удалению знаков "плюс" справа от этих окон. По завершении ввода нажимается Enter.

4. Анализатор. В верхней части экрана расположены друг под другом 5 пар окон. В левом окне пары указывается уровень сканирования задачи на исследование, по достижении которого будет происходить обращение к анализатору; в правом окне - максимальный уровень соответствующего обращения к анализатору, по превышении которого работа анализатора будет прервана. Обычно указывается не более одной - двух таких пар (например, одна - для предварительного обращения, имеющего целью быстро усмотреть нужные следствия, и другая - для более длительного анализа). По завершении ввода нажимается Enter.

5. Оператор фильтра. В верхнем окне текстовым редактором набирается краткое описание смысла входных и выходных переменных оператора. Оно будет использоваться в качестве текста того пункта оглавления базы приемов, в котором расположатся задающие формат оператора справочники. Оно же будет выводиться вместо фиктивной теоремы приема "блок" при просмотре приемов оператора. Далее перечисляются (как термы "вхождение(...)", "тип(...)" и т.п.) типы значений входных и выходных переменных, уточняется тип обращения к оператору, и нажимается Enter.

По окончании указанных действий будут созданы все необходимые приемы, инициализирующие работу с оператором, причем эти приемы автоматически регистрируются в новых подпунктах оглавления приемов, созданных внутри ветви оператора.

Глава 12

Примеры записи приемов на ГЕНОЛОГе и упражнения

Не все приведенные в предыдущих разделах элементы языка ГЕНОЛОГ используются одинаково часто. Некоторые из них возникали для весьма специфических случаев, и более подробно о таких элементах будет сказано во второй книге данной монографии, посвященной опыту обучения решателей задач. Фактически, эта книга и будет представлять собой настоящее введение в программирование на ГЕНОЛОГе. Здесь же, чтобы возникло хотя бы предварительное представление о стандартах такого программирования, разберем несколько десятков приемов, записанных на ГЕНОЛОГе, и проиллюстрируем на них ряд наиболее часто встречающихся элементов языка.

12.1 Примеры записи приемов

Приводимые в данном разделе примеры имеются в реальной базе приемов решателя. Рекомендуется при ознакомлении с очередным приводимым примером приема войти в просмотр его оригинала в базе приемов, и находить в этом оригинале те элементы, о которых далее пойдет речь.

12.1.1 Приемы тождественной замены

Начнем с приема, осуществляющего переход от логарифма произведения к сумме логарифмов (путь к нему в оглавлении базы приемов - "Элементарная алгебра", "Логарифмы", "Общая стандартизация выражений", "Логарифм произведения"). Сразу ясно, что здесь возможны различные варианты приема - в зависимости от того, усматриваются ли из контекста знаки сомножителей: случай, когда усматривается неотрицательность одного сомножителя и произведения остальных; случай, когда усматривается неположительность одного сомножителя и произведения остальных; случай, когда при преобразовании вводятся модули, и т.п. Остановимся лишь на первом из них. Теорема приема здесь может быть представлена в следующем виде:

$$\forall_{abc}(0 \le a \ \& \ 0 \le b \to \log_c(ab) = \log_c a + \log_c b).$$

Сразу заметим, что в ней опущены условия на область допустимых значений: $0 < c, \neg(c = 1)$, число(a), число(b), число(c). Это - простейшее проявление принципа

контекстной корректности для теорем приемов: отбрасываются проверки тех условий, которые, при соблюдении некоторых необременительных требований на "правильную" постановку задачи и на фактически допускаемые решающими правилами приемов преобразования, автоматически будут выполняться в контексте применения приема.

Заголовок приема определяет направление замены слева направо, т.е. представляет собой логический символ "второйтерм" (в приемах замены заголовок указывает на заменяющий терм равенства либо эквивалентности).

Первый из фильтров приема - "уровень(1)" - определяет уровень срабатывания приема при сканировании задачи. Ясно, что в большинстве стандартных ситуаций целесообразно бывает почти сразу переходить к логарифмам сомножителей, чтобы после приведения подобных членов добиться упрощения выражения. После такой стандартизации, на этапе редактирования ответа, можно будет предпринять обратную свертку в логарифм произведения. Разумеется, прием можно было бы сделать более избирательным, блокируя данное преобразование, если оно заведомо ничего не дает. Впрочем, такие дополнительные фильтры при обучении вводятся лишь по мере надобности, и на рассматривавшихся задачах для данного приема не возникли.

Следующий фильтр приема имеет вид "или(не(тип(преобразовать)) посылка комент(длина))". Он блокирует применение приема для изменения условия задачи на преобразование, если эта задача уже имеет комментарий "длина". Такой комментарий вводится после завершающей обработки преобразуемого терма процедурой "свертка", обращающейся к пакетным нормализаторам сокращенной записи - очевидно, на этом этапе развертка логарифма произведения в сумму логарифмов нецелесообразна.

Наконец, последний фильтр имеет вид "не(контекст(подтерм(равно(теквхожд x4)) тип(описать) условие) не(известно(теквхожд)) известно(x4))". Он блокирует преобразование логарифма в сумму, если этот логарифм содержит неизвестные и является левой частью уравнения, правая часть которого известна. В такой ситуации целесообразнее сразу провести потенцирование.

Оба антецедента теоремы приема обрабатываются с помощью проверочных операторов (именно, "усмменьшеилиравно"). Для задания такого способа их обработки служит указатель "блокпроверок(1 2)".

Чтобы предусмотреть случай, когда перед произведением стоит минус, в прием введен указатель идентификации "заменазнака(минус x1)". При наличии минуса он будет передан сомножителю x1; проверка неотрицательности выполняется для модифицированного таким образом x1.

Наконец, указатель "заменавхождений" определяет режим одновременной замены всех вхождений в задачу данного логарифма (с контролем возможности усмотрения в контексте замены неотрицательности сомножителей); указатель "нормализатор" - выделяет прием как выполняющий преобразование общей стандартизации. Он активизирует попытку общей стандартизации надтермов преобразуемого выражения после выполнения замены.

Подтермы заменяющего терма приема снабжены указателями нормализации, обеспечивающими их общую стандартизацию. Сам заменяющий терм $\log_c a + \log_c b$ снабжен нормализатором "нормплюс"; выражения $\log_c a$,$\log_c b$ - нормализаторами "нормлогарифм". Например, если основание логарифма c совпадало с множителем b, то нормализатор "нормлогарифм" заменит выражение $\log_c c$ на 1, и в качестве заменяющего терма будет использовано выражение $\log_c a + 1$.

Разобранный только что пример весьма прост; для сравнения приведем другой, более сложный (хотя далеко не самый сложный из имеющихся в решателе) прием тождественной замены. Он осуществляет стандартизацию тригонометрических выражений с помощью тождества $(\sin x)^2 + (\cos x)^2 = 1$. Мы специально используем здесь обозначение $(\sin x)^2$ вместо $\sin^2 x$, так как в формульном редакторе решателя принята именно эта запись.

Прежде всего, предпримем некоторое обобщение данного тождества, чтобы им можно было пользоваться даже в неявных ситуациях. Первый шаг обобщения - введение коэффициента a перед слагаемыми левой части: $a(\sin x)^2 + a(\cos x)^2 = a$. Следующий шаг - учет возможных множителей в a, представляющих степени синуса либо косинуса x. Если такие множители есть, то фактически в задаче будут усматриваться не вторые, а какие-то другие степени синуса и косинуса x; именно, здесь понадобится теорема вида $m-p = 2 \,\&\, q-n = 2 \to a(\cos x)^m(\sin x)^n + a(\cos x)^p(\sin x)^q = a(\cos x)^p(\sin x)^n$. Она получена из предыдущей выделением в a сомножителей $(\cos x)^p$, $(sinx)^n$; для остатка множителей снова использована буква a. Чтобы такая обобщенная теорема давала необходимую идентификацию и в простейшем случае, когда никаких дополнительных множителей указанного вида в общем коэффициенте не было, придется использовать указатели приема, объясняющие компилятору, что значения p и n могут равняться 0, а соответствующие множители $(\cos x)^p$, $(\sin x)^n$ - отсутствовать. Следующий шаг обобщения теоремы - учет случая, когда коэффициенты при слагаемых различны и применяется частичное исключение квадратов синуса и косинуса - например, при переходе от $5(\sin x)^2 + 3(\cos x)^2$ к $2(\sin x)^2 + 3$. Если выделять константу на основе того численного коэффициента, который меньше по модулю, причем делать это только в случае совпадения знаков слагаемых, то окончательно возникает теорема вида:

$m - p = 2 \,\&\, q - n = 2 \,\&\, (0 < r \,\&\, 0 < s \,\&\, t = \min(r, s) \vee r < 0 \,\&\, s < 0 \,\&\, t = \max(r, s)) \,\&\, g = r - t \,\&\, h = s - t \to ra(\cos x)^m(\sin x)^n + sa(\cos x)^p(\sin x)^q = ga(\cos x)^m(\sin x)^n + ha(\cos x)^p(\sin x)^q + ta(\cos x)^p(\sin x)^n$.

Здесь r, s - числовые коэффициенты; a - произведение прочих общих у рассматриваемых слагаемых множителей (если они есть); t - определяемый в третьем антецеденте числовой коэффициент с наименьшим модулем; g, h - остатки числовых коэффициентов слагаемых после выделения из них t. Заметим, что хотя бы один из этих остатков всегда равен 0, но для упрощения записи теоремы оба они явно фигурируют в заменяющем выражении. Применение к данному выражению нормализаторов общей стандартизации сразу же отбросит нулевые слагаемые, и в итоге заменяющий терм будет выглядеть вполне неизбыточным образом. Замедление, проистекающее из такого, на первый взгляд не очень экономного способа вычислений, в действительности пренебрежимо мало по сравнению с паузами сканирования между применениями приемов. Поэтому исключение разбора случаев за счет применения заменяющих термов избыточно общего вида, в сочетании с обработкой их нормализаторами, устраняющими в конкретных случаях всю возможную избыточность - это стандарт программирования на ГЕНОЛОГе.

Заметим, что в приведенной выше записи для краткости опущен квантор общности по всем переменным, встречающимся в тождестве. При фактическом вводе теоремы приема наличие такого квантора является обязательным. В большинстве случаев пропуск в кванторной приставке нескольких переменных особой роли играть не будет, однако иногда это может привести к неправильной компиляции приема (в особенности там, где перед компиляцией выполняется преобразование теоремы приема - например, в геометрии).

Прием, имеющий теорему указанного выше вида, можно найти в разделе ("Элементарная алгебра","Тригонометрия", "Синус", "Общая стандартизация выражений", "Стандартизация тригонометрических многочленов с применением тождества ..."). В концевом пункте этого раздела расположено несколько приемов (кроме рассматриваемого здесь применения тождества для суммы квадратов синуса и косинуса, обобщаются также замены вида $1-(\cos x)^2=(\sin x)^2$ и $1-(\sin x)^2=(\cos x)^2$, а кроме того, рассматривается стандартизация с участием четвертых степеней синуса и косинуса). К нужному нам приему можно перейти, нажимая клавишу "курсор вниз" до последней точки цепи приемов. Переменные в его теореме обозначены иначе, чем в приведенной выше, однако в остальном эти теоремы идентичны.

Заголовок приема - "второйтерм"; фильтры - "уровень(1)" (тождество применяется почти сразу же, как только предоставляется такая возможность); "натуральное(m)", "натуральное(n)", "натуральное(p)", "натуральное(q)"; "десчисло(r)", "десчисло(s)". Последние фильтры ограничивают применение приема лишь теми случаями, когда показатели степеней при синусе и косинусе суть десятичные записи натуральных чисел, а коэффициенты r,s также суть десятичные записи числовых констант.

Указатель "программа(1 2 3 4 5)" означает, что все антецеденты являются программно реализуемыми; это возможно из-за того, что встречающиеся в них переменные идентифицированы, согласно фильтрам, с числовыми константами, и соответствующие операции вычитания, определения минимума и максимума, а также проверка неравенств реализуются непосредственно.

Указатели "перечень(r десчисло(r))" и "перечень(s десчисло(s))" заставляют компилятор сразу идентифицировать r,s с произведениями всех числовых множителей рассматриваемых слагаемых. В действительности, благодаря предшествующей стандартизации, заключающейся в перемножении числовых констант, таких множителей может быть не более одного; при их отсутствии переменная идентифицируется с единицей рассматриваемой ассоциативно-коммутативной операции, т.е. в данном случае с обычной единицей. В принципе, данные указатели здесь не обязательны - они лишь препятствуют отнесению числовых коэффициентов к произведению a остальных общих множителей.

Указатели "единица(1 m n p q r s a)" и "заменазнака(минус r s)" определяют возможность вырожденных, единичных значений степеней и коэффициентов, а также возможность наличия знаков "минус" перед слагаемыми, которые будут отнесены к числовым коэффициентам.

Указатели "подстановка(фикс(0 1 1 4) n 0)" и "подстановка(фикс(0 1 2 3) p 0)" определяют возможность отсутствия множителей $(\sin x)^n$ и $(\cos x)^p$, причем в этом случае показатели степени n,p идентифицируются с нулем.

Указатель "пустоеслово(a)" означает, что список оставшихся общих множителей может быть пустым - он нужен, чтобы компилятор не вставил проверку непустоты такого списка. Указатель "пересечениесписков(фикс(0 1 1)фикс(0 1 2))" означает, что заголовки слагаемых (даже после отбрасывания возможных знаков "минус") не обязательно должны быть символами умножения. При отсутствии данного указателя компилятор вставил бы проверку наличия заголовка "умножение" несмотря на наличие указателя "единица(...)", так как произведения слишком длинны и автоматическое распознавание возможности вырожденного случая здесь пока не предусмотрено.

Указатель "титр(набор(1 терм(фикс(0 1 1)фикс(0 1 2))))" определяет поясняющий срабатывание приема текст. Шаблон этого текста можно просмотреть и при необходимости отредактировать, нажав из просмотра приема клавишу "6". В данном

случае появится следующий текст: "Преобразуем слагаемые () и (), используя формулу для суммы квадратов синуса и косинуса". Вместо скобок в него будут вставлены выражения, определяемые указателем, т.е. в точности рассматриваемые слагаемые - их указатели вхождения в теорему суть "фикс(0 1 1)" и "фикс(0 1 2)".

Нормализаторы приема просматриваем, нажав сначала клавишу "5" и используя для смены нормализатора клавиши PageDown, PageUp (выход из просмотра - по End). Все подвыражения заменяющей части с заголовками "степень", "умножение", "плюс", снабжены нормализаторами общей стандартизации "нормстепень", "нормумножение", "нормплюс". Кроме того, сами преобразуемые слагаемые также снабжены нормализаторами "нормумножение". Это сделано для того, чтобы в сопровождающий текст подставлялись выражения, у которых устранены умножения на коэффициенты 1, возникшие при идентификации.

12.1.2 Приемы эквивалентной замены

Приемы эквивалентной замены обычно используются либо для общей стандартизации утверждений, либо для разрешения относительно неизвестных условий задачи на описание (уравнения, неравенства и т.п.).

Рассмотрим прием, предназначенный для решения квадратных уравнений. В оглавлении базы приемов к нему ведет путь "Элементарная алгебра", "Степени", "Решение уравнений", "Решение квадратного уравнения"; далее нажатиями клавиши "курсор вниз" переходим к последнему приему группы приемов данного концевого пункта.

При формулировке теоремы приема следует учитывать, что приемы предварительной стандартизации уравнений переводят все известные слагаемые в правую часть уравнения, оставляя в левой части только неизвестные слагаемые. Соответственно, квадратное уравнение будет иметь вид $ax^2 + bx = c$, а не традиционный вид $ax^2 + bx + c = 0$. Теорема приема записывается в следующем виде (для удобства ссылок в ней взяты обозначения переменных из базы приемов - неизвестная вместо x обозначена d):

$$\forall_{abcde}(e = b^2 + 4ac \to ad^2 + bd = c \leftrightarrow (\neg(a = 0)\ \&\ 0 \le e\ \&\ (d = \tfrac{\sqrt{e}-b}{2a} \vee d = -\tfrac{\sqrt{e}+b}{2a})$$
$$\vee bd = c\ \&\ a = 0)).$$

Единственный антецедент теоремы служит для того, чтобы можно было ввести вспомогательное обозначение e для дискриминанта трехчлена. Фактически, как будет видно из дальнейшего, он не только вводит обозначение для дискриминанта, но и вычисляет этот дискриминант, обращаясь для преобразований его к вспомогательным пакетным операторам.

Заголовок приема - "второйтерм". Фильтры "уровень(1 2)", "условие", "тип(описать)", "известно(a)", "известно(b)", "известно(c)", "не(известно(d))", "корень" указывают, что прием применяется при сканировании задачи на описание для известных a, b, c и не известном d, причем уровень сканирования должен быть равен 1 либо 2, а преобразуемое равенство должно иметь корневое вхождение, т.е. собственно являться уравнением.

Фильтр "альтернатива(неизвестные(2)уровень(1)уровень(2))" уточняет, что в задачах, имеющих более одной неизвестной, уровень срабатывания приема равен 1, а в задачах с одной неизвестной он равен 2. Такого рода фильтры возникают, если в тех или иных примерах желательно обеспечить опережающее срабатывание одних приемов и отложить срабатывание других. Они имеют сугубо эвристический характер и

могут при последующем обучении многократно изменяться или отбрасываться - если локальные "конфликты" между приемами, из-за которых эти фильтры возникли, были устранены другими методами.

Последние два фильтра - "или(равно(числонеизвестных(корень)1)не(контекст(условие(x6)заголовок(x6 целое натуральное)первыйсимвол(x6 x7)неизвестная(x7)входит(x7 d))))" и "не(контекст(ключ(цели связка x6)пересекаются(x6 d)))" - отсекают случаи уравнений, решаемых средствами других разделов. В первом из них блокируется применение формулы корней квадратного уравнения для уравнения в целых числах, имеющего более одной неизвестной; во втором - применение этой формулы при рассмотрении дифференциального уравнения (для решения дифференциальных уравнений, не разрешенных явно относительно производной, используется специальный прием, который обозначает производную новой переменной - неизвестной, относительно которой и выполняется разрешение в рамках некоторой вспомогательной задачи). Такие фильтры возникают по мере того, как обучение решателя распространяется на новые разделы. Часто при этом бывает нужна некоторая расчистка для блокировки срабатываний ненужных приемов из ранее рассматривавшихся разделов; обычно для нее бывает достаточно рассмотрения нескольких первых примеров.

Указатель "идентификатор(1)" определяет, что первый антецедент теоремы используется для ввода вспомогательного обозначения e. Указатели "единица(1 a b)" и "заменазнака(минус a b)" определяют, что коэффициенты a, b могут принимать вырожденное значение 1 и что минус перед соответствующими членами передается данным коэффициентам.

Наконец, указатель "примечание(условие(меньше(1 число(условие(x6)заголовок(x6 равно)))разборслучаев))" вводит комментарий "разборслучаев" к преобразованному уравнению, учитывая свозможность, что оно после преобразования может оказаться дизъюнкцией, если число уравнений в задаче больше одного. Такой комментарий форсирует разбор случаев по данной дизъюнкции - для предотвращения возникновения новых дизъюнкций, которые могли бы появиться при независимом преобразовании остальных уравнений.

Перечислим нормализаторы приема. Для вычисления дискриминанта используется цепочка обращений к нормализаторам "нормплюс", "стандплюс", "видумножение". Первый предпринимает общую стандартизацию сумм; второй выполняет раскрывание скобок; последний - пытается разложить дискриминант на множители, чтобы впоследствии упростить извлечение из него квадратного корня. Утверждение $a = 0$ из заменяющей части обрабатываемого нормализатором "нормусм" (этот нормализатор предпринимает попытку усмотрения истинности либо ложности обрабатываемого им утверждения путем применения проверочного оператора, и в случае удачи заменяет данное утверждение на логическую константу "истина" либо "ложь"), а также нормализатором "нормчисло" (он служит для простейшей стандартизации числовых равенств). Если удается усмотреть истинность либо ложность утверждения $a = 0$, то в заменяющей части вместо него сразу же вводится соответствующая логическая константа.

Аналогично, утверждение $0 \leq e$ из заменяющей части преобразуется нормализаторами "нормусм" и "нормменьшеилиравно" (простейшая стандартизация нестрогих неравенств). Если его истинность либо ложность очевидна, то вместо него сразу же будет подставлена логическая константа. При обращении к указанным нормализаторам вводится дополнительная посылка $\neg(a = 0)$.

При преобразовании квадратного корня из дискриминанта используются нормализаторы "нормстепень" и "сбросмодуля". Последний нормализатор отбрасывает мо-

дули у сомножителей преобразуемого им выражения - в данной ситуации модуль не нужен, так как все равно при определении корней берутся два значения корня из дискриминанта, отличающиеся знаком. При обращении к нормализаторам вводится дополнительная посылка $0 \le e$.

Выделим, далее, цепочки "нормплюс", "видумножение", "посылки($0 \le e$)", используемые для разложения на множители числителей $\sqrt{e} - b$, $\sqrt{e} + b$, а также нормализатор "видумножение", обеспечивающий разложение на множители коэффициента a. Наконец, заметим, что ко всему заменяющему терму применяется нормализатор "нормлог", обеспечивающий общелогическую чистку утверждения, в частности, устраняющий содержащиеся в нем логические константы.

Следующий пример приема эквивалентной замены - интегрирование дифференциального уравнения с разделяющимися переменными. Этот прием может быть найден в разделе ("Дифференциальные уравнения", "Уравнения первого порядка", "Уравнение с разделяющимися переменными", "Интегрирование уравнения с разделяющимися переменными"). Берется последний прием из группы приемов данного концевого пункта - переход к нему осуществляется путем нажатий клавиши "курсор вниз". Напомним, что дифференциальное уравнение с одной неизвестной записывается как равенство с производными и формально представляет собой условие не на саму неизвестную функцию y, а лишь на значения ее и производных в некоторой "текущей" точке x. Более аккуратная в логическом отношении запись потребовала бы ввода явного обозначения для функций, определяемых левой и правой частями равенства, и оказалась бы более громоздкой и непривычной. Поэтому в логическом языке решателя (как и в обычной практике) здесь применяется контекстная семантика: равенство для значений в точке x в действительности понимается по умолчанию, как равенство для функций, причем такое понимание определяется специальной целевой установкой задачи - наличием цели (связка $x_1 \dots x_n$), перечисляющей все аргументы неизвестных функций, встречающихся в условиях задачи. Эта цель является как бы вынесенной за скобки логической конструкцией, связывающей встречающиеся в задаче переменные $x_1, \dots x_n$. Учитывая такое соглашение о записи дифференциальных уравнений в условиях задачи, для решения дифференциального уравнения с разделяющимися переменными, приведенного предварительной стандартизацией к виду

$$a(x)b(y(x))\frac{dy(x)}{dx} + c(x)e(y(x)) = 0,$$

будем иметь следующую теорему приема:

$$\forall_{abcexyhv}(\int \tfrac{b(z)}{e(z)}dz = \lambda_z(h(z), \text{число}(z)) \ \&\ \int \tfrac{c(x)}{a(x)}dx = \lambda_x(v(x), \text{число}(x)) \rightarrow$$
$$a(x)b(y(x))\tfrac{dy(x)}{dx} + c(x)e(y(x)) = 0 \leftrightarrow \exists_p(\text{число}(p) \ \&\ h(y(x)) = -v(x) + p)$$
$$\&\ \neg(e(y(x)) = 0) \vee e(y(x)) = 0 \ \&\ a(x)b(y(x))\tfrac{dy(x)}{dx} = 0 \ \&\ \text{число}(\tfrac{dy(x)}{dx})).$$

Первый и второй антецеденты теоремы будут определять вычисление соответствующих интегралов; возникающие после интегрирования функции $h(z)$ и $v(x)$ (где вспомогательная переменная z введена на период интегрирования вместо $y(x)$) используются для описания параметрического семейства решений с произвольной постоянной p. Особо учтен подслучай обращения в 0 второго слагаемого, причем утверждение "число$(dy(x)/dx)$"из этого подслучая введено как эквивалент условия дифференцируемости функции $y(x)$ в тех точках, где она рассматривается.

Заголовок приема - "второйтерм". Фильтры "условие(2)", "условие", "тип(описать)", "корень", "неизвестная(y)" означают, что прием срабатывает на втором уровне сканирования (сразу после простейшей стандартизации уравнения), причем преобразуется условие задачи на описание, у которого y - неизвестная функция. Фильтры "известно(фикс(0 1 1 1 1))", "известно(фикс(0 1 1 2 1))" означают, что выражения $a(x)$, $c(x)$ не содержат неизвестных.

Указатель "идентификатор(1 2)" определяет рассмотрение первого и второго антецедентов как равенств, определяющих значения вспомогательных выражений. В каждом из них изначально идентифицированы все переменные левой части; эта часть будет сформирована и обработана сопровождающими ее нормализаторами (они и вычислят интегралы). Затем полученное выражение будет идентифицироваться с правой частью равенства, что и позволит получить $h(z)$,$v(x)$ для дальнейшего их использования.

Указатель "отображение(a b c e h v)" означает, что для перечисленных в нем переменных Y выражения $Y(X)$ - в скобочной записи они выглядят как "значение(Y X)" - будут идентифицироваться с произвольными выражениями. При отсутствии данного указателя $Y(X)$ идентифицировалось бы только с выражением вида "значение(...)".

Указатель "примечание(серия)" сопровождает преобразованное уравнение комментарием "серия", означающим, что возникший в нем квантор существования дает параметрическое описание ответа.

Указатели "перечень(b не(известно(b)))", "перечень(e не(известно(e)))" определяют идентификацию $b(y(x))$, $e(y(x))$ со всеми содержащими неизвестные множителями соответствующих слагаемых. Указатели, определяющие выделение указанных в теореме двух слагаемых путем группировки всех членов суммы, имеющих производную, и всех членов, имеющих множителем неизвестную функцию, отсутствуют, так как эти группировки заблаговременно осуществляются другими приемами.

Указатели "единица(1 a b c)" и "заменазнака(минус a c)" означают возможность вырожденных значений 1 для $a(x)$, $b(y(x))$, $c(x)$, а также передачу знаков "минус" перед слагаемыми выражениям $a(x)$,$c(x)$.

Указатели "новаргумент(b x фикс)", "новаргумент(e x фикс)" определяют способ идентификации $e(y(x))$, $b(y(x))$. Перед идентификацией этих выражений переменные y, x должны быть уже идентифицированы. Компилятор создает обращение к процедуре "новаргумент", проверяющей, что вхождения переменной x внутри идентифицируемых с $e(y(x))$, $b(y(x))$ выражений расположены только внутри $y(x)$. Все такие вхождения выделяются, и при формировании $b(z), e(z)$ вместо них будет подставлена новая переменная z. То, что эта переменная - новая, следует пояснить компилятору отдельно, что и делает указатель "новаяпеременная(z)". Логический символ "фикс" в приведенных выше указателях введен, чтобы показать, что идентифицируемые выражения не подвергаются предварительной обработке нормализаторами, ориентированными на приведение всех подвыражений с x к виду $y(x)$ - очевидно, здесь это излишне.

Указатель "конец(1 2)" отодвигает на конец трудоемкую обработку первого и второго антецедентов (т.е. фактически интегрирование уравнения) до того, как будут проверены все остальные условия срабатывания приема.

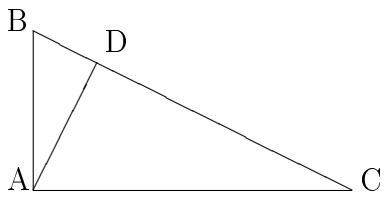
Наконец, указатель "удалениеусловия(число(производная(отображение(x4 число(x4)значение(y x4))x)))" удаляет из списка условий утверждение о существовании производной $dy(x)/dx$, которое после интегрирования уравнения не нужно.

Оба интеграла из антецедентов снабжены нормализатором "нормИнтеграл". Это

- чрезвычайно большой пакетный нормализатор, осуществляющий в решателе формальное интегрирование. Перед обращением к нему подынтегральные выражения обрабатываются нормализатором "упрощинтеграл", выполняющим некоторые несложные предварительные преобразования. Этому нормализатору передается комментарий (переменная X), указывающий переменную интегрирования X. Для предварительного упрощения параметрического описания, возникающего после интегрирования уравнения, применен нормализатор "уравндифф". Впрочем, и нормализатор "упрощинтеграл", и нормализатор "уравндифф" - очень маленькие, почти вырожденные пакеты, которые при последующем развитии системы, возможно, будут отброшены. Прочие используемые в приеме нормализаторы осуществляют общую стандартизацию подтермов заменяющего терма.

12.1.3 Приемы вывода в посылках задачи

Рассмотрим один из простейших приемов вывода, связанный с прямоугольным треугольником, у которого проведена высота к гипотенузе (см. в оглавлении базы приемов путь "Элементарная геометрия", "Фигуры", "Треугольник", "Прямоугольный треугольник", "Высота прямоугольного треугольника, проведенная из прямого угла", "Соотношение для длин катета, гипотенузы и отрезка гипотенузы между основанием высоты и вершиной треугольника"). В этом случае прием уже снабжается чертежом:



Теорема приема имеет вид:

$\forall_{ABCD}$(прямая(AB)$\perp$прямая(AC) & $D \in$ прямая(BC) & прямая(AD)$\perp$прямая(BC) $\rightarrow l(BD) \cdot l(BC) = (l(AB))^2$).

Прием имеет заголовок "вывод"; он позволяет в тех случаях, когда длины отрезков BD,BC,AB уже введены в рассмотрение либо их ввод в рассмотрение представляется целесообразным, присоединить к посылкам задачи связывающее эти длины соотношение, указанное в теореме.

Фильтр "уровень(5 7)" определяет два возможных уровня срабатывания приема - пятый и седьмой. Хотя один из последующих фильтров уточняет, в каких случаях срабатывание на пятом уровне нецелесообразно, но он сравнительно слабый, и остается достаточно много ситуаций, при которых срабатывание возможно как на пятом, так и на седьмом уровнях. Такая избыточность иногда создается в приемах вывода, чтобы срабатывание было возможным, даже если необходимые для него посылки появляются в задаче не сразу, а лишь на достаточно высоких уровнях сканирования. Напомним, что другая возможность обеспечить срабатывание приема после появления недостававших ранее посылок - уменьшить вес его "опорной" посылки тем приемом, который ввел представляющую интерес для ее пересмотра информацию.

Фильтры "посылка" и "или(тип(доказать)тип(исследовать))" встречаются практически во всех геометрических приемах, так как обычно геометрические задачи на вычисление решаются путем вывода следствий в посылках их блока анализа. Иногда здесь добавляется возможность срабатывания в посылках задач на преобразование,

что бывает нужно для задач на нахождение геометрических мест точек.

Фильтры "не(равно(D B))" и "не(равно(D C))" представляют собой ускорители отсечения: они позволяют в процессе идентификации сразу отбрасывать положения точки D, совпадающие с B и C, не выполняя в этих случаях дальнейших проверок. Опыт показывает, что отсутствие в геометрическом приеме фильтров, отсекающих вырожденные положения точек, может существенно замедлить работу решателя, так как прием начнет при сканировании задачи предпринимать повсюду множество бессмысленных проверок. Поэтому они обычно вводятся еще до оптимизации приема на примерах.

Фильтры "усм(актив(расстояние(A B)))", "усм(актив(расстояние(B C)))", "усм(актив(расстояние(D x5)))" означают, что в посылках задачи должны встречаться расстояния $l(BC)$,$l(AB)$, а также расстояние от точки D до какой-либо другой точки. Эти фильтры обрабатываются идентифицирующими операторами - ввод фильтра "усм(F)" эквивалентен присоединению F к списку антецедентов теоремы приема и указанию на обработку этого антецедента идентифицирующим оператором. Фильтры вида "усм(...)", кроме ввода определенных ограничений на срабатывание приема, играют роль ускорителей идентификации, так как позволяют переходить от одного объекта к другому с использованием буферов идентифицирующих операторов - фактически это эквивалентно применению сетевого представления. Так, фильтр "усм(актив(расстояние(A B)))", если бы точка A была уже идентифицирована, а B - еще нет, позволил бы искать последнюю лишь среди тех точек, для которых явно введено в рассмотрение их расстояние до точки A.

Ввод в рассмотрение расстояний $l(AB)$,$l(BC)$ должен быть обеспечен до попытки применения данного приема; впрочем, можно создавать несколько различных версий одного и того же приема, имеющих различную фильтрацию и срабатывающих на различных уровнях. Одна из такиз версий данного приема (с уровнем срабатывания 13) приведена в том же концевом пункте оглавления базы приемов, что и описываемый прием.

Фильтры "конец(или(известно(терм(расстояние(AB))) Неизв(терм(расстояние(A B))))))" и "конец(или(известно(терм(расстояние(BC))) Неизв(терм(расстояние(BC)))))" означают, что каждое из расстояний $l(AB)$,$l(BC)$, после обработки нормализатором "нормрасстояние", должно быть либо выражено термом без неизвестных, либо представлять интерес для определения неизвестных внешней задачи на описание. Последнее условие выражается при помощи фильтра "Неизв(...)" (как в данном случае), либо при помощи фильтра "неизв(...)", накладывающего несколько более сильные требования и применяемого чаще, чем "Неизв". Каждый из этих фильтров прослеживает цепочку равенств в посылках задачи, начинающуюся равенством, содержащим заданное выражение, и заканчивающуюся равенством, содержащим неизвестные внешней задачи на описание; связи между равенствами в такой цепочке - по вхождению общего выражения вида "расстояние(...)" либо "угол(...)". Подробнее - см. описания операторов "Неизв(...)", "неизв(...)". Указатели "конец(...)" в приведенных выше фильтрах отодвигают их проверку на конец программы приема - чтобы сравнительно трудоемкие процедуры определения нормализованных расстояний применялись лишь после установления необходимой конфигурации чертежа.

Фильтр "конец(не(равно(терм(расстояние(AB)) терм(расстояние(AC)))))" указывает, что треугольник ABC не должен быть равнобедренным - иначе вводимое приемом соотношение дублирует соотношения, вводимые другими, более простыми приемами.

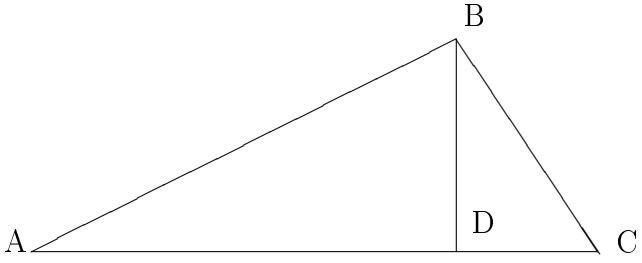
Следующие два фильтра - "конец(или(известно(терм(расстояние(AB))) извест-

но(терм(расстояние(CD))) Неизв(терм(расстояние(CD))) комментпосылок(треугольник степень треугольник(ABC))))" и "или(уровень(7) комментпосылок(треугольник степень треугольник(ABC)) внешнеизв(терм(расстояние(AB))) известно(терм(расстояние(AB)))меньше(число(разряд(результат расстояние x6))2))" - возникли при попытке несколько ограничить применение данного приема, если для рассматриваемого прямоугольного треугольника уже выписывалось соотношение теоремы Пифагора. Ввод этого соотношения сопровождается добавлением комментария "треугольник степень треугольник(ABC)" к посылкам задачи. Оба фильтра чисто эвристические; формулировки их извлечены из сравнения двух классов задач, в одном из которых срабатывание приема было необходимым, а в другом - избыточным.

Наконец, последний фильтр "не(известно(результат))" - блокирует выписывание соотношения, если все его переменные суть известные параметры задачи.

Указатели приема суть "усм(1 2 3)" и "теквхожд(1)"; они определяют применение идентифицирующих операторов для обработки антецедентов теоремы. Активизация приема начинается с усмотрения в посылках задачи некоторого утверждения о перпендикулярности, отправляясь от которого программа приема и идентифицирует первый антецедент.

В качестве следующего примера вывода в посылках рассмотрим геометрический прием, выполняющий простейшее дополнительное построение - проведение высоты в треугольнике. Собственно говоря, имеется много различных приемов, выполняющих именно это действие - каждый применяется в некоторой своей специфической ситуации. Мы выберем одну из наименее громоздких версий - лежащую в разделе ("Элементарная геометрия", "Фигуры", "Треугольник", "Отрезок, соединяющий вершину треугольника с точкой на противоположной стороне или на ее продолжении", "Высоты треугольника", "Проведение высоты в треугольнике"). В данном разделе находится два приема; берем последний, переходя к нему при необходимости нажатием клавиши "курсор вниз". Чертеж приема несложен:



Теорема этого приема - один из возможных примеров используемых в ГЕНОЛОГе псевдотеорем:

$\forall_{ABCD}$(актив($l(AB)$) & актив($l(BC)$) & актив($l(AC)$) $\rightarrow$ точка(D) & $D \in$ прямая(AC) & прямая(BD)$\perp$прямая(AC) & актив($l(BD)$)).

Она не является формулировкой какого - либо свойства геометрических конфигураций, а лишь определяет ту ситуацию, в которой нужно выполнить дополнительное построение (это делают антецеденты теоремы приема) и перечисляет те утверждения (в консеквенте), занесение которых в список посылок означает выполнение построения. Как и обычно, фильтры приема накладывают ряд дополнительных условий на ситуацию, в которой это построение будет делаться.

Антецеденты теоремы означают, что к моменту применения приема расстояния $l(AB)$, $l(BC)$, $l(AC)$ уже должны в явном виде упоминаться в посылках задачи. В геометрии такое упоминание расстояний, углов и площадей приводит к немедленному

срабатыванию приемов, регистрирующих их в посылках вида "актив(X)", где X - данное расстояние, угол, площадь, и т.п.

Консеквент теоремы представляет собой конъюнкцию утверждений о новой точке D - основании высоты, проводимой приемом. Здесь указывается, что новый объект D представляет собой точку, лежащую на прямой AC, что прямые BD и AC перпендикулярны, и что расстояние $l(BD)$ - длина высоты - далее рассматривается как представляющее интерес.

Заголовок приема - "вывод". Фильтры "уровень(8)", "посылка", "или(тип(доказать)тип(исследовать))" - стандартные для планиметрии.

Следующий фильтр - "контекст(усм(принадлежит(х5 отрезок(AB)) принадлежит (х6 отрезок(AC)) перпендикулярно(прямая(х5 х6)прямая(AC)) актив(расстояние(х5 х6))) не(равно(х5 A)) не(равно(х5 х6)) не(равно(х5 C)))" - раскрывает подлинную мотивировку срабатывания приема: на одной из сторон AB,BC уже имеется точка, из которой опущен перпендикуляр на сторону AC, причем длина этого перпендикуляра явно упоминается в посылках. Разумеется, это обстоятельство делает данное дополнительное построение гораздо более объяснимым.

Фильтр "не(контекст(усм(перпендикулярно(прямая(х7 B)прямая(AC)))))" означает отсутствие ранее проведенного перпендикуляра к прямой AC, проходящего через точку B - он необходим, чтобы прием не начал бесконечный процесс ввода все новых и новых обозначений для основания такого перпендикуляра.

Фильтр "конец(или(контекст(усм(равно(расстояние(AB) расстояние(BC)))) не(контекст(список(х5 терм(расстояние(AB)) терм(расстояние(BC))) не(известно(х5))))))" требует, чтобы либо треугольник ABC был равнобедренным, с вершиной B, либо чтобы расстояния $l(AB), l(BC)$ уже были вычислены. В этих случаях попытки связать с помощью теоремы Пифагора длины сторон треугольника и длину высоты $l(BD)$, ради которых и введен прием, становятся достаточно осмысленными.

Указатель "усм(1 2)" определяет обработку первого и второго антецедентов с помощью идентифицирующих операторов; точка инициализации приема при сканировании задачи - третий антецедент. Он дает точки A и C, а точка B далее извлекается из пересечения множеств точек, для которых выделены расстояния их до A и C.

Указатель "новыйсимвол(D фикс(0 1)фикс(0 2)фикс(0 3))" означает, что для D выбирается новая переменная, а в посылки внешней задачи на описание передаются утверждения "точка(D)", "$D \in$ прямая(AC)", "прямая(BD)$\perp$прямая(AC)".

Указатели "новаяпосылка(х5 контекст(вид(х5 принадлежит(х6 отрезок(х7 B)))) 2)", "новаяпосылка(х5 контекст(вид(х5 актив(расстояние(х6 х7)))усм(принадлежит (х6 прямая(A C))))3)" определяют переключение внимания после проведения высоты: веса посылок, указывающих принадлежность точки отрезку с концом в B, уменьшаются до 2, а веса посылок, выделяющих расстояния до точек прямой AC, уменьшаются до 3. Такие указатели вводятся по мере надобности в процессе прогонки решателя по задачнику - там, где указанное переключение внимания было совершенно необходимо для решения задачи либо позволяло существенно его ускорить. По существу, пока они представляют собой лишь обучающий материал для анализа подходов к автоматическому созданию в приемах средств переключения внимания. Однако, даже при накоплении их по указанному выше принципу (в общем, укладывающемуся в рамки концепции обучения методом "поощрений и наказаний"), решатель достаточно часто начинает замечать новые возможности, возникающие после срабатываний приемов.

Указатели "перпендикулярно(D B A C)" и "отрезок(B D)" определяют пополнение чертежа; первый из них вводит на чертеже точку D как основание перпендику-

ляра, опущенного из B на AC, второй - соединяет B с D отрезком.

Следующий пример - тоже геометрический. Он связан с ситуациями, в которых вводится численный параметр, обозначающий некоторую характеристику чертежа - расстояние, угол, и т.п. Такой параметр нужен лишь для промежуточных вычислений и в ответ включаться не будет. Однако, если временно рассматривать его как известную величину, то будет инициировано срабатывание многих полезных приемов, и шансы найти ответ задачи, в котором вспомогательные параметры, "сократившись" так или иначе, не встретятся, существенно возрастут.

Рассматриваемый прием вводит обозначение для длины катета прямоугольного треугольника, если нужно определить угол такого треугольника. Он расположен в оглавлении базы приемов по адресу ("Элементарная геометрия", "Перпендикулярно", "Ввод вспомогательного параметра - длины перпендикуляра, опущенного из точки на прямую"); как и ранее, берется последний прием из концевого списка, к которому переходим, нажимая клавишу "Курсор вниз" до прекращения изменений на экране.

Чертежа этот прием не имеет - ввиду простоты ситуации его применения. Теорема приема имеет вид:

$\forall_{ABCab}(\angle(ABC) = a$ & прямая$(AB) \perp$прямая$(AC) \rightarrow b = l(AB))$.

Заголовок приема - "вывод". Фильтры "уровень(6)", "тип(исследовать)", "цель(известно)" определяют срабатывание его на сравнительно большом (шестом) уровне в задаче на исследование.

Фильтр "внешнеизв(a)" означает, что выражение a не известно, и все входящие в него неизвестные являются теми неизвестными внешней задачи на описание, которые следует вычислить. В частности, в a не могут входить обозначения точек чертежа - обычно все эти обозначения суть неизвестные задачи на исследование и одновременно рассматриваются как известные во внешней задаче на описание, ибо входят в ее посылки. Данный фильтр указывает, что вычисление угла ABC в текущем контексте представляет интерес. Ясно, что для определения угла достаточно знать лишь отношение длин катетов, и поэтому временное рассмотрение длины $l(AB)$ одного из них, как известной, в рассматриваемой задаче перспективно.

Фильтр "контекст(посылка(х3)заголовок(х3 равно)вхождениетерма(х3 терм(расстояние(AB))х4)не(контекст(операнд(х5 х4)символ(х5 равно))))" означает, что расстояние $l(AB)$, для которого вводится вспомогательное обозначение, до этого уже было введено в рассмотрение в некотором равенстве, связывающем между собой параметры чертежа, причем это равенство не было явно разрешено относительно $l(AB)$.

Фильтр "не(контекст(равно(х3 терм(расстояние(AB))) или(известно(х3)внешнеизв(х3))))" отбрасывает действие приема, как заведомо бессмысленное, если расстояние $l(AB)$ либо уже вычислено, либо выражено только через те неизвестные, которые суть вычисляемые во внешней задаче параметры. В этих случаях необходимая инициализация приемов будет выполняться и без ввода специального обозначения для данного расстояния.

Фильтр "не(контекст(посылка(х3) вид(х3 равно(х4 расстояние(AB))) переменная(х4)))" блокирует ввод обозначения, если рассматриваемое расстояние уже равно некоторой переменной. Фильтр "не(Входит(вспомпараметр комментариипосылок))" блокирует применение приема, если до этого уже был введен какой-либо вспомогательный числовой параметр. Возможно, это и чересчур жесткое ограничение - вполне можно представить себе задачу, в которой нужно ввести несколько вспомогательных

числовых параметров. В общем-то, некоторые из приемов решателя, вводящих такие параметры, лишены этого ограничителя, и при дальнейшем обучении может понадобиться несколько ослабить его и в данном приеме, однако пока такой надобности не возникало.

Фильтры "усм(актив(прямая(AB)))" и "усм(актив(прямая(AC)))" указывают, что прямые AB,AC уже были явно выделены в задаче.

Указатель "усм(2)" определяет применение идентифицирующего оператора при проверке условия перпендикулярности. Указатель "вспомпараметр(b фикс(0))" определяет выбор новой переменной b в качестве обозначения для $l(AB)$ и регистрацию ее в качестве временно известного параметра. Указатель вхождения "фикс(0)", ссылающийся на заносимое в посылки задачи на исследование утверждение $b = l(AB)$, также определяет перенесение его в список посылок внешней задачи на описание.

Чтобы в указанных фильтрах выражение "терм(расстояние(AB))" для уже вычисленного расстояния заменялось на его величину, в приеме введен нормализатор "нормрасстояние" для $l(AB)$. Одновременно он может затронуть и результирующее утверждение теоремы приема, однако в тех случаях, когда фильтры истинны, это несущественно.

Приведем теперь примеры приемов вывода в посылках, не относящиеся к планиметрии.

В режиме логического вывода решаются задачи на качественное исследование графика функции вещественной переменной. Исследуемая функция и связываемые с ней вспомогательные функции обозначаются переменными, и в посылках задачи на исследование происходит накопление различной информации об этих функциях. Например, расмотрим прием вывода, вычисляющий множество корней производной исследуемой функции (см. раздел "Математический анализ", "Общие свойства числовых функций", "Исследование функций", "Качественное описание графика функции", "Определение корней производной"). Его теорема имеет вид:

$$\forall_{afgb}(a = \lambda_x(g(x), f(x)) \rightarrow \mathrm{roots}(a, \mathrm{Dom}(a)) = set_x(f(x)\ \&\ g(x) = 0)).$$

Здесь a - обозначение для производной исследуемой функции (что будет видно из фильтров приема). Правая часть равенства в консеквенте обрабатывается вспомогательной задачей, в которой решается уравнение $g(x) = 0$; на основе результата решения создается новая посылка, дающая явное выражение для множествак корней производной на ее области определения. Конечно, данная теорема приема - это еще один пример псевдотеоремы, или почти - псевдотеоремы.

Фильтры "уровень(2)", "посылка", "тип(исследовать)", "цель(исследовать)" стандартные; новым является лишь последний из них, указывающий на то, что решается задача на качественное описание графика функции. Фильтр "контекст(посылка(x3) вид(x3 Производная(x4 a)))" означает наличие посылки, характеризующей функцию a как производную некоторой функции x4. Фильтр "комментпосылки(1 корни)" выражает отсутствие комментария к задающему функцию a равенству, означающего, что корни этой функции уже ранее вычислялись. Указатель "примечпосылки(1 корни)" создает такой комментарий в случае применения данного приема.

Указатель "примечание(ориентацияравенства)" снабжает вводимое равенство для множества корней производной пометкой, блокирующей перестановку его левой и правой частей. Уточнение ориентации равенства позволяет управлять последующими заменами, которые оно порождает. В данном случае выражение "roots(a,Dom(a))" будет заменяться на явное описание множества корней, а не наоборот. Указатель

"примечание(равно)" снабжает новую посылку пометкой, блокирующей замену его подтермов с помощью других равенств. Указатель "примечание(упростить)" вводит пометку, блокирующую последующие попытки упрощать новое равенство.

Указатель "отображение(fg)" определяет идентификацию $f(x), g(x)$ с произвольными выражениями; Указатель "точкапривязки(отображение)" уточняет, что инициализация применения приема при сканировании будет начинаться с усмотрения символа "отображение" в равенстве для a. Наконец, указатель "лимит(20000000)" ограничивает трудоемкость попыток нахождения корней производной.

Нормализаторов приема всего два. Прежде всего, это обработка пакетным нормализатором "нормкласс" всей правой части нового равенства. Такой нормализатор предпринимает попытку избавиться от описателя "класс", перейдя к более простым средствам задания множеств. Однако, еще до обращения к нему, условия принадлежности классу $f(x)$ & $g(x) = 0$ преобразуются вспомогательной задачей на описание, в которой, собственно, и осуществляется основная работа данного приема. Нормализатор приема, организующий обращение к этой задаче, имеет вид "задача(6 тип(описать) цель(неизвестная(x)) полный явное прямойответ корни одз упростить)". Он сопровождается дополнением "удалениепосылок(x3 входит(x x3))", отбрасывающим из контекста применения приема все утверждения с x. Применение его происходит путем решения вспомогательной задачи на описание, условиями которой служат условия принадлежности классу, посылками - урезанный указанным образом контекст, а неизвестной - x, причем эта задача решается до максимального уровня 6. Ответ задачи и берется в качестве результата применения нормализатора приема.

В качестве последнего примера приемов вывода рассмотрим прием из аналитической геометрии, извлекающий из условия параллельности прямой и плоскости соотношение для коэффициентов уравнений, задающих эти прямую и плоскость. Прием размещен в разделе ("Аналитическая геометрия", "Уравнение прямой в пространстве", "Условие параллельности прямой и плоскости").

Теорема приема имеет вид:

$\forall_{ABCDEKabcdefpqrs}$(прямая($AB$)$\|$плоскость($CDE$) & коорд(прямая($AB$), K) = set_{xyz}(пропорцнаборы($(x{+}a, y{+}b, z{+}c), (d, e, f)$) & число($x$) & число($y$) & число($z$)) & коорд(плоскость($CDE$), K) = set_{uvw}($pu + qv + rw + s = 0$ & число(u) & число(v) & число(w)) $\rightarrow pd + qe + rf = 0$).

Используемый в теореме предикат "пропорцнаборы($(x{+}a, y{+}b, z{+}c), (d, e, f)$)"означает линейную зависимость указанных в нем двух трехмерных векторов и определяет, таким образом, прямую, проходящую через точку $(-a, -b, -c)$ и имеющую направляющий вектор (d, e, f). Значением переменной K служит та аффинная система координат, в которой выписываются уравнения.

Заголовок и фильтры приема - обычные; указатель "усм(1)" определяет проверку параллельности прямой и плоскости с помощью идентифицирующих операторов - так, как они использовались в элементарной геометрии. Указатели "единица(0 a b c s)" и "единица(1 p q r)" допускают вырожденные нулевые и единичнеы значения слагаемых и коэффициентов. Указатель "заменазнака(минус p q r)" учитывет возможность знака "минус" перед коэффициентами. Указатели "подстановка(фикс(3 2 4 1 1 1)p 0)", "подстановка(фикс(3 2 4 1 1 2)q 0)" и "подстановка(фикс(3 2 4 1 1 3)r 0)" определяют возможности пропуска в уравнении плоскости слагаемых с каждой из переменных u, v, w.

Уравнение плоскости идентифицируется непосредственно с некоторым равенством в посылках, в то время как для второго антецедента, определяющего уравнение прямой , введен указатель "идентификатор(2)". Это означает, что сначала выражение коорд(прямая$(AB), K$) обрабатывается нормализатором, определяющим некоторое уравнение прямой, а затем уже найденное уравнение идентифицируется с правой частью второго антецедента. Указатель "точкапривязки(коорд)" выбирает для инициализации срабатывания приема при сканировании символ "коорд" из уравнения плоскости.

Кроме уже упомянутого выше нормализатора "нормкоорд", в приеме используются обычные нормализаторы общей стандартизации, а также нормализатор "стандравно", упрощающий новое равенство для блокировки вывода дублирующих соотношений - в таких случаях равенство заменяется на константу "истина" и вывод не происходит.

12.1.4 Приемы вывода в условиях задачи на описание

Приемы вывода следствий в условиях задачи на описание применяются очень редко, так как они лишь увеличивают объем записей в списке условий, а этот список должен быть постепенно преобразован к виду ответа. При получении в списке условий явных выражений для неизвестных будет выполнена подстановка этих выражений в оставшиеся условия, и далее начнутся преобразования по проверке их истинности. Всякое добавление новых утверждений к списку условий будет, как правило, лишь усложнять данную проверку. Поэтому основными средствами решения задач на описание являются, во-первых, эквивалентные преобразования условий, постепенно приводящие их к виду ответа, и, во-вторых, использование логического вывода в блоке анализа задачи на описание, с сохранением указаний на взаимную выводимость получаемых утверждений. Эти указания, в случае нахождения в блоке анализа явных выражений для неизвестных, позволяют хотя бы частично исключить из списка условий задачи на описание те условия, которые являются следствиями найденных выражений для неизвестных.

В ряде случаев, однако, прием вывода настолько явно приближает задачу к ответу, что его целесообразно применить непосредственно для пополнения списка условий, не откладывая до рассмотрения посылок блока анализа. Мы приведем здесь два примера такого рода. Первый из них связан с возведением в куб уравнения с тремя кубическими радикалами. Он может быть найден в разделе ("Элементарная алгебра", "Степени", "Решение уравнений", "Уравнение с радикалами", "Возведение в куб уравнения с тремя кубическими радикалами").

Теорема приема имеет вид:

$$\forall_{abcdef}(a\sqrt[3]{d} + b\sqrt[3]{e} + c\sqrt[3]{f} = 0 \rightarrow a^3d + b^3e + c^3f - 3abc\sqrt[3]{d}\sqrt[3]{e}\sqrt[3]{f} = 0).$$

При ее получении последнее слагаемое левой части переносится в правую часть; далее обе части возводятся в куб; после группировки в левой части выделяется выражение $3ab\sqrt[3]{d}\sqrt[3]{e}(a\sqrt[3]{d} + b\sqrt[3]{e})$, в котором сумма в скобках заменяется, с учетом уравнения, на $-c\sqrt[3]{f}$.

Формально такой переход не является эквивалентным (результирующее уравнение будет, в частности, выполнено при равенстве трех слагаемых исходного уравнения, а исходное при этом может быть не выполнено). Поэтому он реализован как прием вывода следствия в условиях задачи. Заголовок приема - "выводусловия". Прием уменьшает число слагаемых с кубическими радикалами с трех до одного,

что создает благоприятные предпосылки для быстрого решения нового уравнения. После его решения остается старое уравнение, в которое подставляется найденный корень, и далее полученное равенство "проверяется" - упрощается эквивалентными преобразованиями до получения логической константы "истина" либо "ложь", либо сводится к некоторому переносимому в ответ условию на параметры задачи.

Фильтры приема - простейшие: "уровень(5)" определяет срабатывание на пятом уровне; "условие", "тип(описать)" и "корень" - указывают, что рассматривается уравнение в условиях задачи на описание; "не(известно(d))", "не(известно(e))" и "не(известно(f))" - указывают, что выражения d, e, f под радикалами содержат неизвестные.

Так как прием не исключает "старого" уравнения, то для блокирования его многократных срабатываний нужно принимать специальные меры. Это делается с помощью указателя "первыйключ(3 фикс($abcdef$))". После идентификации переменных a, b, c, d, e, f этот указатель инициирует проверку наличия комментария "3" к возводимому в куб уравнению (т.е. к условию текущей задачи на описание). Если такой комментарий имеется, то применение приема прерывается, иначе данный комментарий вводится, и последующие попытки применить прием оказываются заблокированы.

Указатели идентификации "единица(1 a b c)" и "заменазнака(минус a b c)" разрешают коэффициентам a, b, c принимать вырожденное единичное значение, а также передают этим коэффициентам знак минус перед соответствующим слагаемым, если он имеется.

Для обработки подтермов нового уравнения используются нормализаторы общей стандартизации. Кроме того, для обработки левой части нового уравнения применяется нормализатор "стандплюс", обеспечивающий раскрывание скобок и приведение подобных членов после возведения в куб.

Прием имеет текстформульный шаблон, используемый для пояснения преобразований: "Переносим слагаемое () уравнения () в правую часть и возводим обе части уравнения в куб. После этого в сумме утроенных произведений из левой части выделяем сомножитель - левую часть возводимого в куб уравнения, и заменяем его на правую часть данного уравнения. Далее все ненулевые слагаемые группируем в левой части". Указатель "титр(набор(1 терм(фикс(фикс(1 1 3))фикс(1))))" определяет подстановку в данный шаблон выражения $c\sqrt[3]{f}$ и исходного уравнения вместо скобок.

Многие приемы вывода в условиях задачи на описание служат для инициализации разбора случаев, вводя некоторую дизъюнкцию, определяющую такой разбор случаев. В качестве примера рассмотрим прием решения целочисленных уравнений либо неравенств, который при появлении неравенства $x \leq n$ для натурального выражения x с неизвестными выводит дизъюнкцию $x = 1 \vee \ldots \vee x = n$, перечисляющую все его возможные значения. Этот прием находится в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Неравенства с целочисленной неизвестной", "Разбор случаев для выражения, принимающего натуральные значения"). Теорема приема имеет вид:

$\forall_{xn}(x \leq n$ & натуральное$(x) \to \exists_m(m \in \{1, \ldots, n\}$ & $x = m))$.

Фильтры "уровень(2)", "условие", "тип(описать)", "неизвестная(x)", "натуральное(n)" определяют общий контекст срабатывания приема. Заметим, что последний из фильтров не просто требует, чтобы выражение n имело натуральное значение, но чтобы n было десятичной записью конкретного натурального числа. Разумеется, в

этом случае можно выписать конечную дизъюнкцию указанного выше вида. Однако, чтобы эта дизъюнкция не оказалась слошком большой, введен дополнительный фильтр "меньше(числзначение(n)6)", ограничивающий ее не более чем 6 членами. Этот фильтр - не более чем временная заглушка; при дальнейшем развитии приема можно будет, вероятно, выделить случаи, когда нужно выписывать и значительно большее число членов, либо создать дубликат данного приема, выписывающий длинные дизъюнкции на более высоких уровнях, когда прочих средств решения задачи не хватило.

Фильтр "контекст(условие(x1)заголовок(x1 натуральное)первыйсимвол(x1 x2)неизвестная(x2)входит(x2 x))" обеспечивает наличие принимающей натуральные значения неизвестной, встречающейся в выражении x.

Указатель "или(фикс(0)фикс(0 2 1))" нужен для того, чтобы квантор существования из теоремы приема определял при заменах дизъюнкции. Ссылка "фикс(0)" - на вхождение этого квантора общности; "фикс(0 2 1)" - на подквантворное утверждение $m \in \{1, \ldots, n\}$, определяющее перечисление значений параметра m при выписке дизъюнктивных членов. Это утверждение обрабатывается компилятором в точности так же, как программно реализуемые антецеденты.

Указатель "блокпроверок(2)" означает, что проверка "натуральности" значений выражения x выполняется проверочным оператором. Указатель "примечание(разборслучаев)" снабжает новую дизъюнкцию комментарием "разборслучаев", инициирующим немедленный разбор случаев. При его отсутствии перед разбором случаев возникала бы пауза, которая здесь могла бы привести к повторным срабатываниям того же самого приема.

12.1.5 Приемы нормализаторов

Существует множество различных типов нормализаторов, значительно отличающихся друг от друга своим назначением и архитектурой. Приведем здесь примеры приемов для основных их типов.

Нормализаторы общей стандартизации

Нормализаторы общей стандартизации вводятся практически для всех функциональных логических символов; иногда - также для предикатных символов. Они обеспечивают простейшую стандартизацию выражений, заголовком которых служат соответствующие символы. Все такие нормализаторы - корневые, то есть их приемы могут заменить только весь преобразуемый терм целиком. За редкими исключениями, нормализаторы общей стандартизации не должны вводить новых выражений, требущих специального сопровождения по области допустимых значений. При создании нового приема любого типа все новые выражения (быть может, кроме каких-то особых случаев) подвергаются обработке нормализаторами общей стандартизации, что позволяет устранять цепочки срабатываний простейших стандартизирующих приемов вслед за срабатыванием какого-либо более существенного приема. Так достигается весьма значительное ускорение преобразований.

Заголовок нормализатора общей стандартизации для логического символа F получается добавлением приставки "норм" к названию этого символа. Это соглашение упрощает ручную вставку обращений к нормализаторам общей стандартизации (хотя обычно обращения к ним можно добавлять к набранному вручную приему автоматически).

В качестве примера рассмотрим прием нормализатора общей стандартизации "нормумножение", осуществляющий перемножение степеней с одинаковыми основаниями. Путь к приему в оглавлении базы приемов - ("Элементарная алгебра", "Умножение", "Нормализатор общей стандартизации НОРМУМНОЖЕНИЕ", "Умножение степеней с одинаковыми основаниями"). Берется второй из двух представленных в этом разделе приемов. Теорема его имеет вид

$$\forall_{abc}(0 \le a \to a^b a^c = a^{b+c}).$$

Прием имеет заголовок "замена(второйтерм нормумножение)", что соответствует замене слева направо.

Указанная замена, хотя и применяется практически всегда, когда это только возможно, все же имеет некоторые ограничения. Прежде всего, здесь следует отметить использование записей вида $2\sqrt{2}$, которые обычно не преобразуются к виду $2^{3/2}$. Для блокировки такого рода замен в приеме служат фильтры "или(не(символ(b 1))не(десчисло(a))не(констдробь(c)))" и "или(не(символ(c 1))не(десчисло(a))не(констдробь(b)))". Они не разрешают проведение замены, если a есть числовая константа, один из показателей степени равен 1, а другой имеет вид простой числовой дроби.

Другое ограничение заключается в том, что при решении уравнений специально могло быть предпринято выделение в отдельный множитель степени с суммой всех неизвестных слагаемых исходного показателя (при известном основании степени). Например, это могло бы быть полезно для усмотрения квадратного уравнения относительно такого множителя. В этой ситуации необходимо заблокировать обратное преобразование, что и достигается с помощью фильтра "или(коммент(нормуравн) и(известно(b)известно(c)))". Здесь комментарий "нормуравн" указывает на наличие ранее предпринятого специального выделения множителя.

Фильтр "постпозиция(фикс(0 1 2)фикс(0 1 1))" используется для оптимизации; он отбрасывает попытки рассмотрения случаев, когда вторая степень расположена в произведении до первой, так как эти случаи избыточны.

Указатель "блокпроверок(1)"определяет обработку антецедента теоремы проверочным оператором; указатель "единица(1 b c)" разрешает вырожденные случаи, когда вместо соответствующей степени a имеется само a.

Наконец, указатель "комментарий(1 множители)" вводит комментарий "множители" при обращении к проверочному оператору, блокирующий попытку разложения a на множители - для нормализатора общей стандартизации такая попытка уже является слишком дорогостоящей.

Нормализаторы вычисления

Ряд нормализаторов общей стандартизации оказались чрезвычайно большими и превратились в нормализаторы, осуществляющие уже не предварительную стандартизацию, а символьное вычисление. Таковы, например, нормализаторы "нормпроизводная", "нормпредел" и "нормИнтграл", используемые, соответственно, для вычисления производных, пределов и неопределенных интегралов. Хотя они и имеют такие же названия, как прочие нормализаторы общей стандартизации, их формат пополнен рядом существенных элементов. В первую очередь, это указатели "контрольнормализации" и "коррекцияпосылок". Первый из них означает, что результаты обращений к нормализатору будут сохраняться в специальном буфере, и при каждом новом обращении сначала будет предприниматься попытка извлечь готовый результат из буфера. Преобразования нормализаторов, формат которых имеет такой указатель,

отображаются на экране при пошаговом просмотре решения. Второй указатель означает, что нормализатор может пополнять свой список посылок, и новые посылки будут переданы внешней задаче.

Будем рассматривать прием нормализатора "нормпроизводная", выполняющий дифференцирование выражения степенного вида, у которого и основание, и показатель зависят от переменной дифференцирования. Этот прием находится в разделе ("Математический анализ", "Производная", "Нормализатор НОРМПРОИЗВОДНАЯ", "Производная степени"). Он является последним в группе приемов данного раздела.

Теорема приема имеет вид:

$$\forall_{abfgx}(a = \frac{df(x)}{dx} \ \& \ \text{число}(a) \ \& \ b = \frac{dg(x)}{dx} \ \& \ \text{число}(b) \rightarrow \frac{d((f(x))^{g(x)})}{dx} = (f(x))^{g(x)} \cdot (b \ln f(x) + \frac{g(x)a}{f(x)})).$$

В ее первом и третьем антецедентах вычисляются производные основания и показателя; второй и четвертый антецеденты проверяют, что производные удалось вычислить - лишь после этого можно усмотреть, что полученные выражения имеют числовые значения.

Заголовок приема - "замена(второйтерм нормпроизводная)". Фильтров прием не имеет, если не считать формально отнесенного к указателям условия "уровень(4)" на уровень его срабатывания. Кроме данного приема, имеется еще несколько приемов для дифференцирования степени в частных случаях (константное основание либо константный показатель), уровни срабатывания которых меньше 4.

Указатели "идентификатор(1 3)" и "блокпроверок(2 4)" определяют уже объясненное выше использование антецедентов. Указатель "отображение(f g)" означает, что функциональные переменные $f(x), g(x)$ идентифицируются с произвольными выражениями.

Указатель "вывод(условие(коммент(нормпредел))меньше(0 значение(f x)))" определяет занесение новой посылки о положительности основания степени, необходимой для сопровождения по о.д.з. из-за появившегося логарифма. Эта посылка впоследствии будет передана во внешнюю задачу.

Нормализаторы "нормпроизводная", которыми сопровождаются правые части равенств в антецедентах, суть рекурсивные обращения к тому же самому пакету для вычисления соответствующих производных. Подтермы заменяющего терма снабжены обращениями к обычным нормализаторам общей стандартизации. Кроме того, новая посылка обрабатывается нормализатором общей стандартизации строгих неравенств и нормализатором, обеспечивающим стандартное упорядочение операндов коммутативных операций.

Следующий пример - прием нормализатора "нормпредел", вычисляющий предел дроби по правилу Лопиталя. Нормализатор "номпредел" представляет собой пакетный оператор, значительно более крупный, чем оператор "нормпроизводная". Он насчитывает более 200 приемов и имеет 9 уровней срабатывания.

Среди указателей формата этого оператора имеется элемент "разборслучаев", который позволяет при вычислении предела инициировать разбор случаев для параметров рассматриваемого выражения, если в разных ситуациях предел вычисляется по-разному. Результаты разбора отдельных подслучаев потом объединяются в одно условное выражение. При принятии решения о разборе случаев происходит откат к рассмотрению исходного выражения (быть может, с подстановкой в него значений тех

параметров, которые фиксированы в текущем подслучае). Новые случаи относятся к последнему выделенному для рассмотрения надслучаю. При откате сохраняются только результаты рассмотрения ранее разбиравшихся подслучаев и схема этих подслучаев, а весь промежуток вычислений от начала рассмотрения надслучая до инициализации разбора случаев пропадает. На первый взгляд, это может показаться не очень экономным способом вычислений. Однако, так как все результаты последних обращений к пакетным операторам сохраняются в буферах, то при повторе они просто берутся оттуда в готовом виде, и существенных потерь не происходит.

Прием для правила Лопиталя расположен в разделе ("Математический анализ", "Предел", "Нормализатор НОРМПРЕДЕЛ", "Предел дроби", "Правило Лопиталя"). Теорема этого приема имеет следующий вид:

$$\forall_{fgabcdehui}(d = \lim_{y\to a\backslash b} f(y)\ \&\ e = \lim_{y\to a\backslash b} g(y)\ \&\ (d = 0\ \&\ e = 0 \vee (d = \infty \vee d = -\infty)\ \&\ (e = \infty \vee e = -\infty))\ \&\ h = \lambda_y(\tfrac{df(y)}{dy}, \text{число}(y))\ \&\ u = \lambda_y(\tfrac{dg(y)}{dy}, \text{число}(y))\ \&\ c = \lim_{y\to a\backslash b} \tfrac{h(y)}{u(y)}\ \&\ (\text{число}(c) \vee c = \infty \vee c = -\infty)\ \&\ i = c \to \lim_{y\to a\backslash b} \tfrac{f(y)}{g(y)} = i)$$

Первые два антецедента вычисляют пределы числителя и знаменателя дроби; третий проверяет, что либо оба эти пределы равны 0, либо бесконечны. Четвертый и пятый антецеденты вычисляют производные; шестой - находит предел отношения производных; седьмой - проверяет, что предел действительно найден - либо конечен, либо бесконечен. Наконец, восьмой антецедент нужен для обращения к вспомогательной задаче на упрощение найденного для предела выражения.

Запись $y \to a\backslash b$ означает, что y стремится к a, причем b - тип стремления (двусторонний, слева либо справа). Это - общая форма указания типа предела для формульной записи. В конкретных случаях вместо нее используются обычные $y \to a$, $y \to a + 0$, $y \to a - 0$. Указатель b при двустороннем пределе равен 0, при пределе слева - 1, и при пределе справа - 2. Однако, такое кодирование используется лишь на уровне внутреннего, скобочного представления термов.

Заголовок приема - "замена(второйтерм нормпредел)". Фильтры его, в основном, эвристические, и предназначены для блокировки применения в особо сомнительных ситуациях. Первым идет фильтр "не(контекст(операнд(фикс(0 1 1 3) x11) не(длинаменее(x11 конъюнктчлен))))", который отсекает случаи чрезмерно длинных (более 80 символов) числителя либо знаменателя. Некоторым оправданием этому служит то, что часто дифференцирование приводит к дальнейшему усложнению выражений. Заметим, что число 80 представлено в фильтре как символьная константа "конъюнктчлен". Использование символьных констант упрощает компиляцию, но несколько усложняет ручную работу с приемом; по мере усовершенствования интерфейса редактора приемов предполагается его устранить. Пока этого не сделано, для того, чтобы найти нужную символьную константу для большого числа либо определить число, которое она представляет, приходится пользоваться интерфейсом "Ресурсы и установки" из главного меню. Если есть символьная константа, то находится номер ее логического символа и вычитается 260; если есть число, то к нему прибавляется 260, и находится логический символ с данным номером.

Следующий фильтр - "не(контекст(операнд(фикс(0 1 1 3)x11)вид(x11 умножение(
x12 степень(x10 x13)))входит(y x10)не(константа(x13))единица(1 x12)контекст(равно(x14 терм(предел(отображение(y число(y)x10)x2 x1)))заголовок(x14 0 плюсбеск минусбеск))))". Он проверяет наличие сомножителя числителя либо знаменателя, имеющего вид степени с неконстантным показателем x13 и содержащим переменную y

основанием x10. Фильтр предпринимает попытку вычислить предел x14 этого основания (используя нормализатор "нормпредел", применяемый к соответствующему выражению внутри данного фильтра). Если предел оказался равен 0 либо бесконечности, то применение приема блокируется. Эвристической мотивировкой фильтра является то, что дифференцирование не устраняет наличия множителя указанного вида, причем этот множитель влияет на стремление числителя либо знаменателя к 0 или к бесконечности.

Дальше идет фильтр "не(контекст(позиция(x10 корень)символ(x10 факториал) входит(y x10)))". Он отменяет заведомо бессмысленную попытку дифференцирования, если выражение содержит целочисленную функцию "факториал", зависящую от y.

Следующий фильтр служит для предотвращения многократных последовательных дифференцирований. Он имеет вид "контекст(равно(x13 число(комментарий(x11)заголовок(x11 частнпроизв)))или(меньше(x13 2)контекст(операнд(фикс(0 1 1 3) x11)вид(x11 плюс(x12 умножение(x10 y)))единица(0 x12)единица(1 x10)заменазнака(минус x10)не(входит(y x10)))и(меньше(x13 3)не(контекст(операнд(фикс(0 1 1 3)x11) не(длинаменее(x11 соединение))))))не(контекст(тригаргумент(корень x15) подчинено(x15 x16)контекст(вид(x16 степень(x17 x18)) альтернатива(меньше(x13 2) не(константа(x18)) не(целое(x18))))входит(y x15))))". При каждом обращении к оператору "нормпредел" с целью вычислить предел отношения производных вводится новый экземпляр комментария "частнпроизв", которым снабжается это обращение. Таким образом, число этих комментариев равно числу внешних применений правила Лопиталя. Фильтр находит данное число x13 и проверяет выполнение хотя бы одного из следующих условий:

а) x13 менее 2;

б) Числитель либо знаменатель имеет слагаемое, линейное по y, и тогда x13 игнорируется;

в) x13 менее 3, причем числитель и знаменатель имеют каждый не более 40 символов.

Если одно из этих условий выполнено, то далее фильтр проверяет отсутствие тригонометрической функции от выражения с вхождением y, расположенной внутри степенного выражения, у которого при x13 большем 1 показатель степени не является десятичной записью целого числа, а при x13 меньшем или равном 1 - не является константой.

Разумеется, перечисленные условия не являются неоспоримыми и не претендуют на полноту - просто они возникли на тех примерах, которые до сих пор были рассмотрены. При продолжении обучения решателя эвристический образ "перспективной для применения правила Лопиталя ситуации", вероятно, будут развиваться дальше.

Фильтр "коммент(производная)" проверяет отсутствие комментария "производная", который предназначен специально для отмены применения правила Лопиталя и вводится там, где нужна ускоренная попытка вычисления предела. Фильтр "или(длинаменее(значение(h y)конецприставки) длинаменее(значение(uy)конецприставки))" проверяет, что хотя бы одна из производных числителя и знаменателя имеет не более 70 символов в своей записи. Наконец, фильтр "или(коммент(целое)не(контекст(позиция(x10 корень)вид(x10 степень(x11 x12))входит(y x12))))" проверяет, что при наличии комментария "целое" выражение под пределом не имеет степенных подвыражений с показателем, зависящим от y. Комментарий "целое" вводится приемом, который предпринимает попытку сведения вычисления предела последовательности к вычислению предела числовой функции; вероятность того, что правило

Лопиталя здесь окажется полезным в ситуациях с комбинаторного типа "показательными" подвыражениями, невелика.

Указатель "уровень(5)" определяет пятый уровень срабатывания приема (из возможных девяти). Указатель "идентификатор(1 2 4 5 6 8)" выделяет те равенства в антецедентах, которые обрабатываются компилятором как присвоения значений соответствующим переменным. Указатель "блокпроверок(3 7)" определяет использование проверочных операторов для антецедентов, анализирующих промежуточные результаты c, d, e вычисления пределов. Заметим, что эти антецеденты построены из элементарных утверждений с помощью конъюнкций и дизъюнкций. Такие ситуации поддерживаются компилятором, обеспечивающим необходимую логику обращений к операторам, проверяющим отдельные элементарные утверждения. В тех случаях, когда утверждение имеет вид равенства переменной X константе A, компилятор вместо обращения к проверочному оператору просто вставляет оператор ЛОСа, распознающий наличие заголовка A у терма, идентифицированного с X.

Указатель "отображение(f g h u)" определяет возможность идентификации функциональных переменных $f(y)$, $g(y)$, $h(y)$, $u(y)$ с произвольными выражениями.

Указатель "лимит(вариант(контекст(список(x10 значение(f y) значение(g y)) вид(x10 степень(y x11)) единица(1 x11)натуральное(x11)) 7000000 1500000)))" определяет органичение числа шагов интерпретатора, отводимых на попытку применения приема. Если оно оказывается превышено, попытка обрывается. Лимит шагов определяется условным выражением, учитывающим контекст: если числитель либо знаменатель представляет собой степень переменной y, имеющую константный натуральный показатель, то он берется значительно большим.

Указатель "титр(вариант(заголовок(d 0)набор(1)набор(2)))" нужен для выбора текстового пояснения, соответствующего конкретной ситуации. Текстформульный шаблон для создания сопровождающих пояснений имеет у данного приема следующий вид:

"¬1 Так как пределы числителя и знаменателя равны 0, применяем правило Лопиталя ¬2 Так как пределы числителя и знаменателя бесконечны, применяем правило Лопиталя ¬3 Производная числителя ¬4 Производная знаменателя ¬5 Предел отношения производных".

В этом шаблоне первые два фрагмента (начинающиеся с ¬1, ¬2) дают общий поясняющий текст. Указатель "титр(...)" определяет выбор первого из них при $d = 0$ и второго - при бесконечном d. Остальные фрагменты шаблона нужны для пояснения обращений к нормализаторам (см. ниже).

Среди нормализаторов приема выделим, прежде всего, рекурсивные обращения к вычислению пределов числителя и знаменателя. Они имеют вид "нормпредел(замечание(производная) замечание(титр(стоп)))". Запись "титр(стоп)" определяет комментарий к нормализатору, означающий, что обращение к нему вообще не будет выводиться на экран при просмотре срабатывания приема.

Обе производные снабжены нормализаторами "номпроизводная(замечание(нормпредел) замечание(титр(набор(i))))", где i есть 3 для первой производной и 4 для второй. Запись "титр(набор(i))" определяет комментарий к нормализатору, означающий, что при обращении к нему на экран будет выводиться поясняющий фрагмент текстформульного шаблона, имеющий номер i.

Отношение производных обрабатывается нормализатором общей стандартизации "нормдробь", к посылкам которого добавляется утверждение $y \to a \backslash b$. Оно может оказаться полезным при сокращении дроби - для усмотрения отличия множителя от нуля. Предел отношения производных обрабатывается нормализатором "нормпре-

дел(примечание(частнпроизв)замечание(титр(набор(5))))", где комментарий "частнпроизв", как уже говорилось выше, играет роль счетчика числа вложенных применений правила Лопиталя, а запись "титр(набор(5))" извлекает из текстформульного шаблона пятый фрагмент для сопровождения обращения к нормализатору.

Наконец, переменная i в последнем антецеденте обрабатывается нормализатором "задача(5 упростить одз цель(условие(длинатекста(списокпеременных(теквхожд)1) ответ)))". Это - обращение к вспомогательной задаче на преобразование, решаемой до 5-го уровня. Если выражение под пределом не имело параметров, то упрощение в рамках приема считается излишним - тогда задача сопровождается целью "ответ", инициирующей немедленную выдачу ответа без каких-либо преобразований.

Для нормализатора "нормпредел" рассмотрим также пример приема, инициирующего разбор случаев. Такой разбор случаев может понадобиться, если дробное выражение зависит от параметров и априори не усматривается отличие предела его знаменателя от 0. Соответствующий прием находится в подразделе "Разбор случаев по равенству нулю предела знаменателя" того же раздела, где был расположен подраздел "Правило Лопиталя". Теорема приема имеет вид:

$\forall_{axfbcd}(a = \lim_{x\to c\setminus b} f(x)$ & число(a) & $d = \lim_{x\to c\setminus b} g(x)$ & число$(d) \to \lim_{x\to c\setminus b} \frac{f(x)}{g(x)} =$ разборслучаев$(d = 0 \vee \neg(d = 0)))$.

Разумеется, это еще один пример псевдотеоремы; равенство предела условной записи "разборслучаев(...)" является лишь удобным для компиляции техническим трюком. Впрочем, этот трюк выходит за рамки одного лишь пакета "нормпредел" и допустим в любом пакетном нормализаторе, описание формата которого снабжено элементом "разборслучаев".

Прием имеет заголовок "замена(второйтерм нормпредел)" - так, как будто он является обычным приемом замены. Указателем на то, что вместо замены происходит инициализация разбора случаев, служит правая часть "разборслучаев(...)" равенства в теореме приема - никаких других указателей для этого не требуется.

Фильтры "не(заголовок(d 0))", "не(константа(d))", "конец(не(легковидеть(не(равно(d 0)))))" отсекают ситуации, когда предел знаменателя не содержит параметров либо очевидно ненулевой. Заметим, что проверка последнего условия, сравнительно трудоемкая, проводится после идентификации остальных элементов ситуации - для этого соответствующий фильтр снабжен пометкой "конец(...)".

Указатель "уровень(3)" фиксирует уровень срабатывания приема; "идентификатор(1 3)" - определяет использование равенств в антецедентах для вычисления пределов числителя и знаменателя. Указатель "блокпроверок(2 4)" определяет использование проверочного оператора для усмотрения того, что пределы удалось вычислить. Наконец, указатель "отображение(f g)" разрешает идентификацию функциональных переменных $f(x), g(x)$ с произвольными выражениями.

Правые части равенств в антецедентах обрабатываются нормализаторами "нормпредел". Условие равенства предела знаменателя нулю обрабатывается простейшим нормализаторами числовых равенств "нормчисло". Заметим, что нецелесообразно применять нормализатор общей стандартизации логических связок "нормлог" к вводимой для разбора случаев дизъюнкции, так как она, очевидно, тавтологична и будет им заменена на константу "истина".

Следующий пример относится к нормализатору "нормИнтеграл", обеспечивающему вычисление неопределенных интегралов. Он имеет несколько более простой

формат, чем нормализатор "нормпредел", так как не предполагает режима разбора случаев. Число уровней срабатывания этого нормализатора равно 8, а число приемов - около полутора сотен, т.е. меньше, чем у нормализатора вычисления пределов. Однако, сами программы приемов здесь более громоздки, так как используется множество стандартных формул и подстановок. На этом материале преимущество программирования на ГЕНОЛОГе особенно наглядно - вместо набора вручную программы на ЛОСе, занимающей несколько полных "экранов", с помощью формульного редактора рисуется средней сложности формула с интегралом, сопровождаемая, как правило, весьма небольшим списком фильтров и указателей.

Рассмотрим сначала пример приема, основанного на табличном интеграле для дроби, числитель и знаменатель которой суть линейные комбинации синуса и косинуса. Этот прием находится в разделе ("Математический анализ", "Интегралы", "Нормализатор нахождения первообразной нормИнтеграл", "Тригонометрические функции", "Дроби с линейными комбинациями синуса и косинуса"). Прием является третьим от начала в списке приемов данного раздела. Теорема его имеет вид:

$$\forall_{abcde}(e = d^2+c^2 \to \int \frac{a\sin x + b\cos x}{c\sin x + d\cos x}dx = \lambda_x(\frac{(bd+ac)x + (bc-ad)\ln|c\sin x + d\cos x|}{e}$$

,число(x)))

Фильтров у приема нет; указатель "идентификатор(1)" определяет вычисление в первом антецеденте вспомогательного выражения e - суммы квадратов коэффициентов знаменателя. Указатели "единица(1 a b c d)" и "заменазнака(минус a b c d)" определяют возможность вырожденных коэффициентов и наличия минуса перед коэффициентом. Указатели "подстановка(фикс(0 1 1 3 1 1)a 0)" и "подстановка(фикс (0 1 1 3 1 2)b 0)" определяют возможность отсутствия в числителе слагаемого с синусом либо косинусом. Наконец, указатель "пересечениесписков(фикс(0 1 1 3 1))" нужен для того, чтобы обратить внимание компилятора на возможное отличие заголовка числителя подынтегрального выражения от символа "плюс" - иначе компилятор потребует, чтобы числитель обязательно был суммой, и этим отсечет указанные выше вырожденные случаи отсутствия одного из слагаемых.

Кроме нормализаторов общей стандартизации, расставляемых, по существу, автоматически, можно выделить нормализатор "стандинтеграл(замечание(переменная x))", которым обрабатывается результирующее выражение для первообразной. Этот нормализатор возник для предварительного упрощения результатов интегрирования - чтобы учесть специфику возникающих здесь типичных подвыражений, а также чтобы добиться ускорения по сравнению с решением вспомогательной задачи на преобразование. Постепенно развиваясь, он достиг сейчас достаточно больших размеров - свыше 100 приемов, некоторые из которых попросту дублируют стандартные общие приемы упрощения, а некоторые ориентированы на частные случаи, появление которых при интегрировании представляется сравнительно вероятным. Обращение к нормализатору должно сопровождаться вводом комментария (переменная x), указывающим на переменную интегрирования.

Если сравнить объем программы приема на ЛОСе (перейти к ней можно нажатием Home, а вернуться - нажатием End) и на ГЕНОЛОГе, то видно, что первый в 4-5 раз больше второго. При этом запись на ГЕНОЛОГе существенно нагляднее.

Следующий пример приема нормализатора "нормИнтеграл" - простейшая логарифмическая замена переменной интегрирования. Прием расположен в подразделе

("Замена переменной интегрирования", "Логарифмическая функция", "Простейшая замена") корневого раздела нормализатора; берется последний прием списка приемов данного подраздела. Усматривая в знаменателе подынтегрального выражения некоторый множитель $f(x)$, прием пытается перейти к новой переменной $y = \ln f(x)$. Теорема его имеет вид:

$$\forall_{fghuv}(\lambda_x(\frac{g(x)}{h(x)\frac{df(x)}{dx}}, \text{число}(x)) = \lambda_x(u(\ln f(x)), \text{число}(x)) \ \& \ \int u(x)dx = \lambda_x(v(x),$$
$$\text{число}(x)) \to \int \frac{g(x)}{f(x)h(x)}dx = \lambda_x(v(\ln f(x)), \text{число}(x))).$$

Смысл первого равенства в антецедентах - убедиться в том, что выражение, остающееся после сокращения подынтегрального выражения на производную заменяющей функции $\ln f(x)$, можно выразить только через эту функцию, как некоторое $u(\ln f(x))$. Далее в теореме приема предпринимается экономия на вводе новой переменной y, и интеграл для $u(y)$ записывается как интеграл для $u(x)$. В возникающую после интегрирования функцию $v(x)$ подставляется обратно $\ln f(x)$.

Фильтр "контекст(подчинено(x1 фикс(0 1 1 3)) символ(x1 логарифм) контекст(подчинено(x2 x1) равно(x2 значение(f x))))" определяет проверку того, что подынтегральное выражение содержит логарифм, внутри которого имеется вхождение выражения, идентифицированного с $f(x)$. Эта проверка предшествует обработке первого антецедента и служит для отсечения случаев, в которых он почти наверняка не будет реализован.

Первый и второй антецеденты выделены указателями "идентификатор(1 2)" - у каждого из них сначала формируется левая часть, а затем происходит идентификация ее с правой частью. Левая часть первого антецедента теоремы имеет вид $\lambda_x(A(x),$ число$(x))$, где $A(x)$ - результат обработки указанного в теореме выражения нормализаторами "задача(5 упростить одз)", "посылки(число(x))". Указатель "новаргумент(u x извлечение)" определяет применение к выражению $A(x)$, перед идентификацией его с $u(\ln f(x))$, нормализатора "извлечение". Этот нормализатор пытается преобразовать $A(x)$ к виду $B(x)$, в котором переменная x встречается только внутри подвыражений $\ln f(x)$. Если это удается, то далее $u(t)$ понимается компилятором как результат замены всех таких подвыражений в $B(x)$ на терм t.

Левая часть второго антецедента обрабатывается нормализатором "нормИнтеграл" - предпринимается попытка вычисления интеграла после замены переменной. Результирующее выражение $v(\ln f(x))$ упрощается с помощью нормализатора "стандинтеграл".

Нормализаторы усмотрения зависимости от заданного выражения

Приемы замены переменной при интегрировании либо при решении дифференциальных уравнений часто связаны с усмотрением возможности представить некоторое выражение A в таком виде, чтобы все вхождения заданной переменной x в A были расположены только внутри вхождений заданного подвыражения t. Не всегда изначально A представлено в таком виде явно, и тогда предварительно приходится выполнять преобразования, приводящие его к данному виду. Здесь используются специальные нормализаторы, наиболее развитым из которых является нормализатор "извлечение". Кроме преобразуемого терма, нормализатору должны быть переданы также x и t. Это делается с помощью вводимого при обращении комментария (новаргумент t x), у которого x представлено в формате терма.

Рассмотрим простой прием нормализатора "извлечение", выражающий четную степень косинуса через тангенс. Этот прием расположен в разделе ("Математический анализ", "Общие свойства числовых функций", "Нормализатор ИЗВЛЕЧЕНИЕ", "Выражение через тангенс"); он является четвертым от конца списка приемов этого раздела. Теорема приема имеет вид:

$$\forall_{ab}((\cos a)^{2b} = \frac{1}{((\mathrm{tg}\, a)^2 + 1)^b}).$$

Заголовок приема - "замена(второйтерм извлечение)"; фильтров прием не имеет. Указатель "единица(1 b)" разрешает принимать параметру b вырожденное единичное значение. Указатель "вход(новаргумент c d)" определяет дополнительную идентификацию, состоящую в том, что находится комментарий (новаргумент A B), и переменные c, d идентифицируются, соответственно, с A и B. Указатель "контекст(вид(c тангенс(a)))" также определяет дополнительную идентификацию; если заголовком выражения, идентифицированного с c, является "тангенс", то переменная a идентифицируется с аргументом тангенса. После того, как прием убедился, что требуется переходить к тангенсам заданного аргумента, и идентифицировал вхождение четной степени косинуса этого аргумента, он применяет указанное выше тождество. В его заменяющей части лишь знаменатель обрабатывается нормализатором общей стандартизации "нормстепень". Вообще, обработка нормализаторами заменяющих термов в случаях перехода к специальным подвыражениям должна предприниматься осторожно, чтобы эти подвыражения после нее не пропали.

Нормализаторы приведения к заданным заголовкам

В некоторых случаях нормализатор бывает нужен для преобразования терма к виду, у которого заголовок или примыкающая к нему корневая часть терма удовлетворяют специальному условию. Типичный пример - разложение выражения на множители, когда требуется получить один из заголовков "умножение", "степень", "дробь", быть может, после отбрасывания корневого минуса. Формально, такое преобразование всегда можно ввести фиктивными средствами (например, умножив выражение на единицу). Однако, имеются определенные эвристические ограничения на разумные средства, применяемые при переходе к заданным заголовкам. По-видимому, их можно выделить из анализа примеров и объяснить генератору приемов, который самостоятельно отбирал бы тождества для создания новых приемов таких нормализаторов. Пока же, при обучении решателя вручную, например, тому же разложению на множители, подобные вопросы не возникают, так как каждому "очевидно", какого рода переходы к произведениям, дробям и степеням здесь полезны, а какие - бессмысленны.

Приведем несколько примеров приемов нормализатора разложения на множители. Этот нормализатор получил название "видумножение", оставшееся за ним после серий экспериментов по автоматическому комплектованию нормализаторов преобразования к заданным заголовкам. Список заголовков в них определялся по имеющимся теоремам для декомпозиции уравнений с заданным заголовкам одной из своих частей (для равенств нулю произведения, дроби либо степени такие теоремы как раз имеются). Нормализатор имеет 6 уровней срабатывания; в описание его формата входят упоминавшиеся выше указатели "контрольнормализации" и "коррекция-посылок". В частности, они предопределяют сохранение результатов обращений к нормализатору (как удачных, так и неудачных) в специальном буфере, и повторные

попытки разложения на множители одного и того же выражения в идущих подряд попытках применения различных приемов оказываются почти бесплатными. Это - очень большой нормализатор; число его приемов более 320. Почти все тригонометрические тождества при обучении оказались отнесенными именно к нему.

В качестве первого примера рассмотрим разложение на множители по формуле суммы кубов. Прием расположен в разделе ("Элементарная алгебра", "Умножение", "Нормализатор разложения на множители ВИДУМНОЖЕНИЕ", "Тождества для непосредственного разложения на множители", "Сумма кубов"). Берется второй из двух приемов данного раздела. Теорема приема имеет вид:

$$\forall_{ab}(a^3 + b^3 = (a + b)(a^2 - ab + b^2)).$$

Прием имеет заголовок "замена(второйтерм видумножение)", указывающий на замену слева направо.

Так как заменяемая часть симметрична, используется фильтр, отсекающий половину попыток применения приема: рассматриваются только случаи, когда выражение a лексикографически предшествует выражению b. Этот фильтр имеет вид "лексикопредшествует(фикс(0 1 1)фикс(0 1 2))".

Фильтр "коммент(числоценка)" блокирует применение приема в ситуациях, когда разложение на множители предпринято для упрощения неравенства с нулем в одной из частей: здесь целесообразнее перенести один из кубов в противоположную часть неравенства и извлечь кубический корень. Комментарий "числоценка" создается тем приемом, который предпринял указанную попытку упрощения неравенства.

Фильтр "или(входит(нормИнтеграл комментарии) не(контекст(комментарий(нормИнтеграл x3) тригаргумент(корень x4)входит(x3 x4))))" блокирует применение приема при предварительном преобразовании подынтегрального выражения, если сумма кубов содержит тригонометрическую операцию от аргумента, содержащего переменную интегрирования. Это - эвристическое соображение, основанное на том, что обычно при интегрировании степени тригонометрических функций преобразуются к функциям кратного аргумента. Другие приемы разложения на множители в таких ситуациях оказались сравнительно безвредны, и лишь приемы для суммы и разности кубов начали приводить к нежелательным последствиям. Далее расположено еще несколько аналогичных эвристических фильтров, блокирующих применение данного приема в тех специальных случаях, когда водимое им довольно ощутимое усложение выражения нежелательно. Все эти фильтры были подсказаны совсем небольшим числом примеров, имеют достаточно грубый характер, и при продолжении обучения возможен цикл их уточнений.

Надо сказать, что любой сколь-нибудь сомнительный или непонятный фильтр легко можно проанализировать заново, определив все те задачи, в которых он блокирует срабатывание приема, или в которых прием срабатывает несмотря на этот фильтр. Для этого применяется прокрутка по разделам задачника, в которых возможно применение приема, после чего возникает исчерпывающая информация, на основании которой можно отбросить фильтр, скорректировать его либо вообще предложить другую схему решения рассматриваемых задач. Таким образом, несмотря на огромное число приемов в решателе, он вполне допускает локальную оптимизацию, и изменения любого элемента любого приема обычно легко контролируются, не приводя к разрушению регулировки базы приемов в целом. В приводимых ниже упражнениях мы проиллюстрируем технику анализа ранее созданных эвристических фильтров.

Указатель "уровень(4)" определяет размещение программы приема в группе приемов данного нормализатора, срабатывающих на четвертом уровне. Указатель "знак-суммы(минус фикс(0 1))" вводит попытку одновременного изменения знака всех слагаемых при применении данного приема. Указатель "модификатор" вводит проверку того, что преобразуемая сумма не имеет других слагаемых - иначе вместо разложения на множители произошла бы попытка группировки, для которой предусмотрен другой прием.

Указатель "замечание(6 фикс(0 2 2))" вводит комментарий к данному нормализатору, означающий, что второй множитель заменяющего терма не должен подвергаться попыткам разложения на множители по формуле квадратного трехчлена (дискриминант неположителен). Прием нормализатора, осуществляющий разложение по формуле квадратного трехчлена, имеет фильтр, учитывающий этот комментарий.

Указатель "замечание(условие(и(входит(нормчислитель комментарии) контекст (тригаргумент(корень x4)))) фильтрудвоения)" вводит комментарий "фильтрудвоения", если преобразуемое выражение содержит тригонометрическую операцию, причем попытка разложения на множители относится к числителю дроби, возникшей после сложения двух дробей. Этот комментарий блокирует попытки перехода к двойному тригонометрическому аргументу, так как в данной ситуации они могут привести к нежелательным последствиям, например, изменить множитель числителя, имеющийся в знаменателе, и сделать невозможным сокращение дроби. При необходимости переход к двойному тригонометрическому аргументу будет выполняться после выхода из рассматриваемого нормализатора.

Расстановка нормализаторов в приеме стандартная; в ряде случаев применение нормализатора общей стандартизации сопровождается применением нормализатора, упрощающего выражение относительно неизвестных: могут быть приведены подобные члены с известными коэффициентами либо выполнена группировка сомножителей под общий неизвестный показатель степени - действия, ускоряющие последующие рассмотрения уравнений. Если обращение к разложению на множители произошло не из задачи с неизвестными, то такой нормализатор ничего не изменяет.

Следующий пример - разложение на множители путем попыток группировки пар слагаемых. Имеется множество приемов такого типа, двойственных приемам непосредственного разложения на множители. Рассмотрим один из них - группировку при усмотрении разности квадратов. Прием расположен в подразделе ("Частичные группировки", "Разность квадратов") корневого раздела нормализатора "видумножение". Теорема его имеет вид:

$$\forall_{abcde}(e = a(b-c)(b+c) + d \to ab^2 - ac^2 + d = e).$$

Выглядит она несколько искусственно, так как сочетает в себе два преобразования - переход от $ab^2 - ac^2$ к $a(b-c)(b+c)$ и разложение на множители суммы, полученной после данной группировки. Это разложение (с учетом указателей нормализации) выполняется первым антецедентом; если результат e действительно имеет вид произведения, дроби либо степени, то прием заменяет на него преобразуемое выражение. Таким образом, прием прибегает к рекурсии, и по завершении ее сразу выдает результат разложения на множители.

Заголовок приема - "замена(второйтерм видумножение)". Фильтр "или(заголовок(e умножение степень дробь) и(заголовок(e минус) первыйсимвол(e умножение степень дробь)))" проверяет, что выражение e - результат разложения на множители

после группировки - действительно имеет требуемый вид "обобщенного произведения".

Фильтр "или(коммент(группировка) контекст(вид(d плюс(х6 умножение(х7 степень(плюс(b минус(c))х8)))) единица(1 х7 х8) заменазнака(минус х7) единица(0 х6)) контекст(вид(d плюс(х6 умножение(х7 степень(плюс(b c) х8)))) единица(1 х7 х8) заменазнака(минус х7) единица(0 х6)))" регулирует применение приема после того, как уже были предприняты какие-то другие группировки. Заметим, что при обращении к разложению на множители после группировки, выполняемое в первом антецеденте, сопровождается комментарием "группировка". При наличии такого комментария рассматриваемый фильтр проверяет, что некоторое слагаемое преобразуемого выражения уже имеет своим множителем степень суммы либо разности b и c - чтобы данная группировка продолжала начатую попытку выделения сквозного общего множителя.

Фильтр "или(не(коммент(группировка)) не(меньше(количествооперандов(теквхожд) 4)))" проверяет, в случае первой группировки пары слагаемых, что общее число слагаемых не менее 4 - для малого числа слагаемых разложение на множители обеспечивается другими приемами, без предварительных группировок.

Фильтр "меньше(количествооперандов(теквхожд)8)" введен для блокировки замедляющих действия решателя попыток разложения на множители слишком длинных сумм путем группировок. Фильтр "не(константа(корень))" блокирует применение группировок при разложении на множители константных выражений. Фильтр "коммент(нормуравнение)" проверяет отсутствие комментария "нормуравнение" - этот комментарий используется для указания на ослабленный режим разложения.

Фильтр "не(контекст(вид(d плюс(дробь(х6 х7)х8))единица(0 х8)заменазнака(минус х6)))" проверяет, что сумма не имеет дробных слагаемых - в этом случае речь идет о сложении дробей, а не о разложении на множители, и применяется другой прием нормализатора.

Фильтр "или(не(константа(b))не(константа(c)))" блокирует группировку для подобных членов ab^2, ac^2, если b, c оказались константами.

Фильтры "не(контекст(вид(b степень(х6 х7)) не(известно(х7))))" и "не(контекст(вид(c степень(х6 х7)) не(известно(х7))))" блокируют группировки при разложении на множители разности частей показательного уравнения, так как обычно в этих случаях показательное выражение заменяется на новую неизвестную.

Наконец, фильтр "коммент(тригонометрия)" проверяет отсутствие комментария "тригонометрия", вводимого нормализатором после того, как было принято решение о попытке разложения на множители путем перехода от степеней тригонометрических функций к функциям кратного аргумента.

Указатель "уровень(4)" откладывает применение приема до четвертого уровня - на случай, если имеется более простой способ разложения, чем группировки. Указатель "идентификатор(1)" объясняет, что первый антецедент нужен для ввода вспомогательного выражения e; указатель "единица(1 a)" - что возможно вырожденное единичное значение коэффициента a.

Кроме обычных нормализаторов общей стандартизации, в приеме применяется обработка правой части первого антецедента нормализатором "видумножение(замечание(группировка) замечание(титр(набор(2))))". Организация текстформульного сопровождения срабатывания приема с помощью указателя "титр(...)" здесь аналогична ранее разобранной.

Для разложения на множители многочленов с целыми коэффициентами, степень

которых не превосходит 5, нормализатор "видумножение" применяет (если неприменимы более простые средства) метод неопределенных коэффициентов. В качестве примера рассмотрим прием для разложения многочлена степени 3 на линейный и квадратичный множители. Этот прием размещен в подразделе ("Разложение многочленов путем подбора целочисленных коэффициентов", "Представление многочлена третьей степени в виде произведения линейного множителя и квадратного трехчлена") корневого раздела нормализатора. Берется второй из двух имеющихся в разделе приемов. Теорема приема имеет вид:

$\forall_{abcdefghijk}(fh = d \ \& \ gj = a \ \& \ 0 < g \ \& \ gi = b - fj \ \& \ gh + fi = c \rightarrow ae^3 + be^2k + cek^2 + dk^3 = (fk + ge)(hk^2 + iek + je^2))$.

Она возникла из следующих соображений. Для разложения на множители однородного многочлена $ae^3 + be^2k + cek^2 + dk^3$ степени 3 относительно переменных e, k с целыми коэффициентами a, b, c, d рассматривается произведение линейного и квадратичного множителей $fk + ge$, $hk^2 + iek + je^2$ с неопределенными целыми коэффициентами f, g, h, i, j. После раскрывания скобок, для неопределенных коэффициентов возникают соотношения $fh = d, gj = a, gi + fj = b, gh + fi = c$. Первое из них позволяет организовать перечисление всевозможных f как делителей известного a, при этом определяется и h. Второе, аналогичным образом, позволяет перечислять g, j. После того, как определены f, j, третье соотношение, переписанное в виде $gi = b - fj$, позволяет перечислять всевозможные g, j. Для текущих найденных значений следует проверять выполнение последнего, четвертого соотношения. Таким образом получаем первый, второй, четвертый и пятый антецеденты теоремы. Третий антецедент нужен для отсечения избыточных случаев - всегда можно поменять знаки всех неопределенных коэффициентов так, чтобы g оказалось положительным (при нулевом g многочлен можно разложить и без данного приема).

Прием имеет заголовок "замена(второйтерм видумножение)". Фильтры "целое(a)", "целое(b)", "целое(c)", "целое(d)" требуют, чтобы коэффициенты были идентифицированы с десятичными записями целых чисел.

Ускоряющий фильтр "постпозиция(фикс(0 1 4)фикс(0 1 1))" отбрасывает симметричные случаи при идентификации - всегда член с третьей степенью переменной k должен идти после члена с третьей степенью переменной e. Фильтр "меньше(количествооперандов(теквхожд) 5)" также ускоряющий - он блокирует попытки идентификации, если число слагаемых больше 4, и таким образом многочлен заведомо не имеет рассматриваемого вида.

Фильтр "коммент(группировка)" блокирует попытки воспользоваться приемом, если реализуется попытка разложения на множители после предпринятой ранее группировки слагаемых.

Фильтр "контекст(разряд(теквхожд степень x12) не(второйсимвол(x12 2)))" также ускоряющий - он отменяет попытку применения приема, если выражение не имеет степеней, отличных от второй. Фильтр "коммент(6 теквхожд)" блокирует бесперспективную попытку применения приема, если имеется комментарий, указывающий, что многочлен был получен как "длинный" множитель в разложении суммы либо разности кубов.

Фильтр "не(константа(корень))" блокирует попытку применения приема при обработке константных выражений. Этот фильтр, конечно, может отсечь какие-то случаи, в которых прием мог бы оказаться полезен. Однако, потребность в разложении константных выражений на множители возникает сравнительно редко, а замедление попытки такого разложения вносят ощутимое. Поэтому данный фильтр просто

означает, что для тех особых случаев, когда разложение константной суммы на множители все-таки необходимо, нужно создать специальный прием, переобозначающий какие-то константные подвыражения на вспомогательные переменные, и затем обращающийся к разложению на множители уже неконстантной суммы.

Фильтр "коммент(нормуравнение)" блокирует применение приема, если есть указание о применении ослабленных средств разложения. Фильтр "меньше(2 количествооперандов(теквхожд))" отсекает вырожденные случаи, когда число слагаемых менее трех.

Указатель "уровень(3)" определяет третий уровень применения приема; "модификатор" - требует, чтобы многочлен не имел слагаемых, кроме приведенных в теореме. Указатель "программа(1 2 3 4 5)" означает, что все антецеденты теоремы являются программно реализуемыми. Первый, второй и четвертый из них используют программу, перечисляющую все множители целого числа. Интересен тот факт, что трудоемкость даже такого неэкономичного перечисления, на фоне трудоемкости общего процесса сканирования задач, невелика, и срабатывание приема выглядит обычно почти мгновенным. Это является следствием того, что архитектура современных процессоров хорошо приспособлена к численным процедурам и никак не приспособлена к логическим процессам.

Указатели "подстановка(фикс(0 1 2)b 0)" и "постановка(фикс(0 1 3)c 0)" определяют возможность отсутствия второго и третьего слагаемых многочлена, причем в этих случаях коэффициенты b, c берутся равными 0.

Указатели "перечень(a десчисло(a))", "перечень(d десчисло(d))" упрощают работу компилятора, фиксируя идентификацию коэффициентов a, d как числовых множителей соответствующих слагаемых.

Указатель "пересечениесписков(фикс(0 1 2)фикс(0 1 3))" объясняет компилятору, что второе и третье слагаемые не обязательно имеют своим заголовком (после отбрасывания возможного минуса) символ умножения. Такое пояснение нужно здесь из-за того, что эти слагаемые имеют по три множителя, и компилятор пока не способен анализировать эту ситуацию самостоятельно.

Указатели "единица(1 a b c d)" и "заменазнака(минус a b c d)" определяют возможность вырожденных единичных и отрицательных значений коэффициентов.

Прием использует только нормализаторы общей стандартизации; его указатель "титр(...)" обычным образом вводит текстформульное сопровождение.

Нормализаторы утверждений с неизвестными

В некоторых случаях для ускорения разрешения условия задачи на описание относительно неизвестных можно вводить специальные нормализаторы. Обычно это имеет смысл делать тогда, когда условия задачи слабо взаимодействуют между собой, и сранительно длительное изолированное преобразование одного из них не нарушает общего хода решения. Например, так бывает при решении неравенств - обычно каждое из них преобразуется к разрешенному относительно неизвестной виду без использования других. В качестве примера рассмотрим прием нормализатора решения строгих неравенств. Этот нормализатор называется "уравнменьше". Он имеет три уровня срабатывания и снабжен буфером сохранения результатов последних обращений. Шаги его применения в обычной трассировке отображаются на экране. Число приемов этого нормализатора равно 74 - не очень много, так как в него включены лишь простейшие переходы, позволяющие ускорять работу решателя при разборе случаев, часто воникающем во время решения неравенств.

Выберем прием решения простейших неравенств вида $ax < b$. Он расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньше", "Нормализатор решения строгих неравенств УСММЕНЬШЕ", "Решение неравенств $XA < B$,$B < XA$"). Берется последний из приемов раздела.

Теорема приема имеет вид:

$$\forall_{abc}(ac < b \leftrightarrow c < 0 \ \& \ \frac{b}{c} < a \ \vee \ 0 < c \ \& \ a < \frac{b}{c} \ \vee \ c = 0 \ \& \ 0 < b).$$

Заголовок приема - "замена(второйтерм уравнменьше)". Фильтры "не(известно(a))", "известно(b)", "известно(c)" указывают, что выражение a содержит неизвестные, а выражения b, c - не содержат. Заметим, что список переменных, которые нормализатор будет считать неизвестными, определяется его комментарием (неизвестные $x_1 \ldots x_n$), автоматически вводимым при обращении к нормализатору из задачи, имеющей неизвестные, либо из внешнего пакетного оператора, обладающего таким комментарием. Такая передача нормализатору списка неизвестных обуславливается наличием в описании его формата указателя "неизвестные".

Указатель "перечень(a не(известно(a)))" фиксирует способ идентификации выражения a путем выделения всех неизвестных множителей левой части неравенства. Указатель "заменазнака(минус a)" разрешает относить минус в левой части неравенства к выражению a.

Выражения b,c при формировании заменяющего утверждения обрабатываются нормализатором "видумножение" - это делается, чтобы чаще происходило сокращение нормализатором "нормдробь" дроби b/c. Последнему нормализатору, применяемому в первом и втором подслучаях заменяющей дизъюнкции, придается дополнительная посылка $c \neq 0$, истинная в обоих этих подслучаях. Неравенства $c < 0$, $0 < c$ и $0 < b$, не содержащие неизвестных, обрабатываются нормализатором "нормменьше" общей стандартизации строгих неравенств. Аналогичным образом, равенство $c = 0$ обрабатывается нормализаторами "нормусм" и "нормчисло", первый из которых предпринимает попытку усмотрения истинности либо ложности утверждений с помощью проверочных операторов, заменяя их после этого на логическую константу, а второй - служит нормализатором общей стандартизации для числовых равенств. Наконец, неравенства с неизвестными $b/c < a$, $a < b/c$ обрабатываются рекурсивным образом - к ним применяется нормализатор "уравнменьше", которому передается дополнительная посылка - неравенство для c. Чтобы исключить логические константы, которые могут возникнуть в заменяющем утверждении после работы перечисленных нормализаторов, ко всему этому утверждению применяется нормализатор "нормлог".

Текстформульный шаблон, сопровождающий применение приема, имеет вид:

"¬1 Так как множитель () положителен, делим на него обе части неравенства, и далее преобразуем полученное неравенство ¬2 Так как множитель () отрицателен, делим на него обе части неравенства и меняем знак неравенства. Далее преобразуем полученное неравенство ¬3 Рассматриваем случаи в зависимости от знака множителя () ¬4 Дальнейшее преобразование неравенства в случае отрицательного множителя ¬5 Дальнейшее преобразование неравенства в случае положительного множителя".

Кроме общей ситуации, в нем предусмотрены случаи однозначно определяемого в контексте знака множителя c - тогда нормализаторы исключат в заменяющей дизъюнкции все случаи, кроме одного, вводя для невыполнимых членов дизъюнк-

ции логическюу константу "ложь", которая затем отбрасывается нормализатором "нормлог". Первый фрагмент текстформульного шаблона ориентирован на случай положительного c, второй - на случай отрицательного, третий - соответствует общему случаю неустановленного знака c. Четвертый и пятый фрагменты используются при рекурсивных обращениях к нормализатору "уравнменьше" - эти обращения сопровождаются элементами "замечание(титр(набор(4)))", "замечание(титр(набор(5)))".

Указатель "титр(вариант(заголовок(фикс(0 2 2 1)истина)набор(1 терм(c))вариант(заголовок(фикс(0 2 1 1)истина)набор(2 терм(c)))набор(3 терм(c))))" является диспетчером, выбирающим из текстформульного шаблона нужный фрагмент и передающим ему для подстановки вместо скобок выражение c.

Нормализатор выделения повторяющихся подвыражений с неизвестными

При решении систем уравнений с несколькими неизвестными часто помогает прием перехода к новым неизвестным. Он основан на обнаружении некоторой группы подвыражений с неизвестными, через которые можно выразить все рассматриваемые уравнения. Чтобы такую группу подвыражений можно было усмотреть, обычно приходится прибегать к предварительным преобразованиям уравнений, явным образом выделяя в них повторяющиеся вхождения одного и того же подвыражения. Для этих преобразований используется специальный нормализатор "повторчисло". Этот нормализатор имеет 6 уровней срабатывания. Буфера для сохранения результатов обращений к нему не предусмотрено; при пошаговом показе решения его действия не отображаются. Тем не менее, он относится к числу сравнительно больших нормализаторов, насчитывая более сотни приемов.

Нормализатор "повторчисло" используется в итеративном цикле обработки выделенной группы утверждений с неизвестными A, для которых нужно усмотреть список подвыражений, выбираемых в качестве новых неизвестных. Здесь возможны два случая - несколько уравнений либо одно неравенство. Нормализатору передается текущее утверждение, которое он и будет изменять, но при этом сообщается и весь список утверждений - через комментарий (внешвхождение A), в котором позиция набора A, соответствующая обрабатываемому утверждению, заменяется на 0. Кроме того, нормализатору передается комментарий (новыенеизвестные B), у которого B - изначально пустой накопитель для заполнения его ссылками на те утверждения из A, к рассмотрению которых следует вернуться после обработки текущего утверждения.

В качестве примера рассмотрим прием нормализатора, выражающий сумму квадратов двух выражений через сумму и произведение этих выражений. Такого рода переход может быть полезен, если в обрабатываемых утверждениях уже встречались (быть может, в слегка замаскированном виде) данные сумма и произведение. Прием расположен в разделе ("Элементарная алгебра", "Число", "Нормализатор выделения повторяющихся вхождений числовых выражений ПОВТОРЧИСЛО", "Плюс", "Сумма и разность квадратов", "Выражение суммы квадратов через сумму и произведение"). Берется последний прием раздела. Теорема приема имеет вид:

$$\forall_{bc}(b^2 + c^2 = (b + c)^2 - 2(bc)).$$

Заголовок приема - "замена(второйтерм повторчисло)". Фильтры "не(известно(b))", "не(известно(c))" указывают на то, что выражения b, c содержат неизвестные; фильтр "неизвестные(2)" - что число неизвестных задачи не меньше 2. Последнее

требование является естественным, так как прием предполагает выделить для последующего обозначения новыми неизвестными сразу два выражения - сумму и произведение b и c.

Фильтр "лексикопредшествует(b c)" отбрасывает избыточный, в силу симметрии, случай обратного порядка выделения слагаемых b^2, c^2.

Фильтр "контекст(внешвхождение(x4) вид(x4 умножение(x5 b c)) единица(1 x5) буфер(новыенеизвестные базавхождения(x4)))" требует, чтобы хотя бы одно из рассматриваемых для перехода к новым неизвестным утверждений имело вхождение x4 произведения, содержащего множители b,c. Идентифицирующий терм "внешвхождение(x4)"перечисляет вхождения x4, встречающиеся вне текущего вхождения в обрабатываемом нормализатором терме, а также вхождения x4 в другие термы, перечисленные в комментарии (внешвхождение ...). Указатель "буфер(новыенеизвестные базавхождения(x4))", расположенный в данном фильтре, обеспечивает передачу в накопитель комментария (новыенеизвестные ...) всех утверждений, в которых выделено требуемое вхождение x4.

Таким образом, при применении приема будет сохранено указание на повторный просмотр всех утверждений списка A, имеющих вхождение произведения b на c. Этот просмотр будет необходим, если данный прием выделит произведение bc в явном виде - тогда в других утверждениях его тоже придется выделять в явном виде из произведений, содержащих, кроме b и c, дополнительные множители.

Фильтр "контекст(внешвхождение(x4) вид(x4 плюс(x5 умножение(x6 b) умножение(x6 c))) единица(0 x5) единица(1 x6) буфер(новыенеизвестные базавхождения(x4)))" аналогичен предыдущему - он проверяет наличие в рассматриваемых утверждениях суммы вида $kb + kc$.

Фильтр "не(контекст(внешоперанд(x4) вид(x4 степень(x5 2)) контекст(список(x6 b c) контекст(внешвхождение(x7) вид(x7 умножение(x8 x5 x6)) единица(1 x8)) контекст(внешвхождение(x7) вид(x7 плюс(x8 умножение(x9 x5) умножение(x9 x6))) единица(0 x8) единица(1 x9)))))" проверяет, нельзя ли подобрать для b либо c другую пару - некоторое новое слагаемое e^2 той же суммы, в которую входят b^2, c^2 , причем так, чтобы снова были выполнены оба предыдущих фильтра. Если это сделать можно, то ситуация с преобразованием неоднозначная, и применение данного приема тогда просто отменяется.

Указатель "уровень(3)" определяет третий уровень для попыток применения приема. Нормализаторами общей стандартизации обрабатываются только явно выделенные в правой части равенства сумма и произведение выражений b, c. Заметим, что в этой части возникает вложенное произведение - двойка умножается на произведение bc. Это необходимо для последующего переобозначения выделенных вхождений на новые неизвестные, и применение каких-то других нормализаторов общей стандартизации к правой части может ее испортить.

Нормализатор группировок

В процессе развития решателя в ГЕНОЛОГ был введен еще один тип нормализаторов, который впоследствии так и не был востребован. Однако, в определенных ситуациях он может оказаться полезным, и здесь мы вкратце опишем его работу. При выполнении цепочки преобразований выражения, направленных на уменьшение некоторого функционала, например, на уменьшение длины этого выражения, можно перед каждым шагом изменения выражения сначала накапливать множество различных альтернативных преобразований, и лишь по исчерпании новых вариантов

принимать окончательное решение. Это решение заключается в определении наиболее ценных, с точки зрения принятого функционала, преобразований обрабатываемого терма, причем отбирается максимальная группа попарно не пересекающихся наиболее ценных локальных изменений, которые и выполняются одновременно. Такой режим работы нормализатора задается указателем "группировка" в описании его формата. Следует заметить, что нормализаторы данного типа обладают ощутимо меньшим быстродействием по сравнению с обычными нормализаторами, выполняющими свои преобразования сразу по обнаружению их возможности. Эффект отбора лучшего варианта обеспечивается у последних распределением приемов по различным уровням и уточнением логики принятия решения. Единственное преимущество нормализаторов с указателем "группировка" (будем называть их нормализаторами группировки) - возможность почти не заботиться о фильтрах их приемов. По существу, такой нормализатор представляет собой просто список тождеств, крайне слабо обработанных с точки зрения управления. Разумеется, это удешевляет нормализатор, но сказывается на качестве его работы.

По этой причине, единственным сохранившимся нормализатором группировки в решателе является нормализатор "группмножество", используемый для сокращенной переформулировки выражений из алгебры множеств. Его можно найти в разделе ("Алгебра множеств", "Множество", "Нормализатор группировки ГРУППМНОЖЕСТВО"). Приемы его практически ничего, кроме теоремы, заголовка и указателя уровня срабатывания, не содержат, и здесь мы их не приводим.

12.1.6 Приемы проверочных операторов

Приемы проверочных операторов обычно просты; рассмотрим сначала несколько приемов оператора "усмменьшеилиравно", предназначенного для проверки нестрогих неравенств. Это - большой оператор, число приемов которого приближается к трем с половиной сотням. Хотя формально он обслуживает произвольные нестрогие неравенства $a \leq b$, однако, если a и b ненулевые, один из его приемов сразу сводит проверку данного неравенства к проверке неравенства $a - b \leq 0$. Поэтому практически все остальные приемы оператора имеют дело только с неравенствами, одна из частей которых - нулевая. Выберем прием, используемый для усмотрения неположительности произведения. Он расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Проверочный оператор УСММЕНЬШЕИЛИРАВНО", "Алгебраические операции", "Усмотрение знака произведения"); берется последний прием раздела. Теорема приема имеет вид:

$$\forall_{ab}(a \leq 0 \ \& \ 0 \leq b \rightarrow ab \leq 0).$$

Заголовок приема - "спуск(усмменьшеилиравно)". Единственный фильтр "уровень(2)" определяет второй уровень срабатывания приема. Указатель "блокпроверок(1 2)" определяет рекурсивное применение того же самого оператора для проверки первого и второго антецедентов. Указатель "титр(набор(1 терм(b a)))" определяет подстановку выражений b и a вместо скобок в шаблон текстформульного сопровождения, имеющий вид "Неположительность произведения неотрицательного множителя () на неположительный множитель ()".

Очевидно, что кроме данного приема, для проверки нестрогого знака произведения необходимы еще прием, устанавливающий неотрицательность произведения неположительных множителей, а также прием, устанавливающий неотрицательность произведения неотрицательных множителей. Оба эти приема расположены в том же

разделе. Последний из них интересен тем, что в нем все сомножители рассматриваются одновременно, вместо выделения их по одному, как в первых двух приемах. Так как это может несколько ограничить способность усмотрения неотрицательности (если по отдельности знаки множителей какого-либо фрагмента не устанавливаются, а известен лишь знак всего фрагмента целиком), то дополнительно введен прием усмотрения неотрицательности произведения неотрицательных множителей, аналогичный первым двум - с увеличенным на единицу уровнем срабатывания.

Прием для усмотрения неотрицательности произведения группы неотрицательных сомножителей имеет такого же вида теорему, как и первые два приема:

$$\forall_{ab}(0 \le a \ \& \ 0 \le b \to 0 \le ab).$$

Хотя в ней всего два множителя, однако указатель "дистрибразвертка(фикс(0 2))" обобщает ее на случай произвольного числа множителей. В остальном этот прием ничем не отличается от рассмотренного выше.

Иногда неотрицательность выражения нужно установить лишь в произвольно малой окрестности заданной точки. Тогда к списку посылок оператора "усмменьшеилиравно" добавляется утверждение "стремится(x a b)", или, в формульной записи, $x \to a\backslash b$; здесь b - указатель типа окрестности (двусторонняя, левая, правая). Это утверждение - лишь техническая условность, позволяющая избежать навешиваний кванторов на весь текущий контекст. В качестве примера приема, учитывающего наличие данного утверждения, рассмотрим прием, который для установления неотрицательности выражения в окрестности точки вычисляет предел его в данной точке. Теорема приема имеет вид:

$$\forall_{abcx}(x \to a\backslash b \ \& \ c = \lim_{x \to a\backslash b} f(x) \ \& \ (c = \infty \vee 0 < c) \to 0 \le f(x)).$$

Заголовок приема - "спуск(усмменьшеилиравно)"; фильтр "уровень(3)" определяет его применение на третьем уровне.

Фильтр "не(входит(предел c))" проверяет, что предел удалось вычислить. Фильтр "входит(x фикс(0 2))" проверяет, что переменная x, для которой задано стремление к a, входит в рассматриваемое выражение - иначе вычисление предела ничего не изменит. Фильтр "коммент(быстрпроверка)" блокирует применение приема в ситуациях, когда есть указание на проведение ускоренной проверки неотрицательности, так как вычисление предела может оказаться достаточно трудоемкой процедурой.

Комментарий "перестановка" вводится приемами, которые для проверки неравенства предпринимают попытку перестановки его частей одновременно с изменением знаков всех слагаемых этих частей. В данном случае имеются два независимых аналогичных приема - один для проверки неотрицательности, другой для проверки неположительности. Поэтому применение приема при наличии комментария "перестановка" излишне, и отмена его обеспечивается фильтром "коммент(перестановка)".

Фильтр "не(контекст(комментарий(слагаемое меньше d)))" аналогичен предыдущему. Комментарий (слагаемое меньше ...) вводится, например, если проверка неравенства $0 \le a$ сводится, с помощью неравенства $0 \le a+b$ из посылок, к проверке $0 \le b$, а также в ряде аналогичных случаев. Таких переходов может оказаться достаточно много, и для ускорения проверок в этих случаях вычисление предела блокируется.

Указатели "идентификатор(2)", "блокпроверок(3)", "отображение(f)" - стандартные. Указатель "потвор(1)" отменяет ограничения на повторное использование утверждения "стремится(...)". Такие ограничения вводятся автоматически для всех

утверждений списка посылок, уже идентифицированных некоторым внешним (по цепочке обращений) пакетным оператором. Это делается лишь для предотвращения зацикливаний в рекурсивных обращениях, и в особых случаях может отменяться.

Для вычисления предела правая часть второго антецедента обрабатывается нормализатором "нормпредел".

Кроме проверочных операторов "усмменьшеилиравно" и "усмменьше", обеспечивающих сравнительно быструю проверку нестрогих либо строгих неравенств, введены еще два проверочных оператора - "провменьшеилиравно" и "провменьше", предназначенные для усиленной проверки, с привлечением ряда специальных неравенств типа неравенства для среднего геометрического и среднего арифметического. Хотя число приемов в этих операторах и невелико, но работают они в режиме перебора группировок, и обращаться к ним слишком часто нецелесообразно. В качестве примера рассмотрим прием оператора "провменьшеилиравно", позволяющий группировать в правой части неравенства неотрицательную сумму. Эта сумма составляется из частично используемых двух положительных членов, коэффициенты которых выбираются так, чтобы получилась сумма квадратов некоторых выражений, а модуль третьего члена суммы оказался равен удвоенному произведению этих выражений. Прием расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Проверочный оператор ПРОВМЕНЬШЕИЛИРАВНО", "Поглощение одного слагаемого двумя с применением квадратичной оценки"). Берется второй из двух приемов раздела. Теорема приема имеет вид:

$\forall_{abcdefghij}(0 \le a \ \& \ 0 \le b \ \& \ c^2 = ab \ \& \ d = 2e - |f| \ \& \ g = 2h - |f| \ \& \ 0 \le d \ \& \ 0 \le g \ \& \ 2j \le da + gb + 2i \to j \le ea + hb + fc + i)$

Эта теорема возникла следующим образом. Для проверки неравенства $j \le ea + hb + fc + i$, у которого e, h, f - численные коэффициенты; выражения a, b неотрицательны, причем c равно произведению квадратных корней из a и b (что формулируется в теореме приема как $c^2 = ab$), предпринимается представление правой части в виде

$$(e - \frac{|f|}{2})a + (h - \frac{|f|}{2})b + (\frac{|f|}{2}a + \frac{|f|}{2}b + f\sqrt{a}\sqrt{b}) + i.$$

Третье слагаемое, очевидно, представляет собой полный квадрат и поэтому неотрицательно. Обозначая коэффициенты первых двух слагаемых через $d/2, g/2$ и отбрасывая неотрицательное третье слагаемое, получаем, что нужно установить неравенство $j \le (d/2)a + (g/2)b + i$, или, после устранения знаменателей, $2j \le da + gb + 2i$.

Заголовок приема - "спуск(провменьшеилиравно)". Фильтры "десчисло(e)", "десчисло(h)", "десчисло(f)" указывают компилятору, что e, h, f должны идентифицироваться с десятичными числами. Фильтр "не(легковидеть(меньшеилиравно(0 умножение(f c))))" блокирует применение приема в случае, когда поглощаемое слагаемое fc само является неотрицательным. Фильтр "уровень(1)" определяет применение приема на первом уровне сканирования. Обычно такой фильтр размещался вначале, однако действительный порядок размещения фильтров в программе приема не очень сильно зависит от порядка размещения их в описании приема; в данном случае расположение фильтра совершенно несущественно.

Фильтр "или(меньше(количествооперандов(фикс(0 2)) 6)меньше(1 число(вид(фикс(0 2) плюс(x11 x12)) единица(0 x11) не(легковидеть(меньшеилиравно(0 x12))))))" блокирует применение приема, если число слагаемых правой части слишком велико (больше пяти), а число тех из них, для которых не усматривается неотрицательность, не более одного. Фильтр "контекст(входит(x11 параметры(фикс(0 2)))

меньше(1 число(разряд(фикс(0 2)x11 x12))))" требует, чтобы правая часть неравенства имела хотя бы одну переменную, встречающуюся неоднократно.

Указатель "блокпроверок(1 2)" определяет использование проверочного оператора для усмотрения неотрицательности выражений a, b. Заметим, что здесь будет использован проверочный оператор "усмменьшеилиравно". Оператор "провменьшеилиравно", как оператор усиленной проверки, требует указателя "проверка(...)". Такой указатель "проверка(8)" определяет рекурсивное обращение для проверки результирующего неравенства $2j \le da + gb + 2i$. Напомним, что различие между обычным и усиленным проверочными операторами заключается в описании их формата. Первый задается термом вида "легковидеть(...)", по которому создается справочник "легковидеть"; второй - термом вида "проверка(...)", по которому создается справочник "проверка".

Указатель "идентификатор(3)" определяет использование третьего антецедента для идентификации переменной b по уже идентифицированным a, c. Указатель "программа(4 5 6 7)" определяет использование антецедентов, начиная с четвертого по седьмой, для непосредственных вычислений с десятичными числами. Указатели "единица(1 b e h f)", "заменазнака(минус e h g)", "единица(0 i)" уточняют вырожденные случаи при идентификации переменных.

Указатели "определено(фикс(0 2 2)b)" и "определено(фикс(0 2 1)c)" блокируют идентификацию слагаемого hb до тех пор, пока переменная b не будет идентифицирована из третьего антецедента, а слагаемого ea - пока не будет идентифицировано c. Это уменьшает перебор в попытках применения приема - сначала будет идентифицироваться слагаемое fc, затем - слагаемое ea, далее будет рассматриваться третий антецедент, который, во-первых, отсечет те слагаемые ea, у которых c^2 не делится на a, а, во-вторых, определит b для однозначной идентификации слагаемого hb.

Указатель "пересечениесписков(фикс(3 2))" явно указывает компилятору, что правая часть третьего антецедента может идентифицироваться с выражением, заголовок которого отличен от символа "умножение".

Указатели "перечень(f десчисло(f))", "перечень(e десчисло(e))" относят к e, f все числовые множители слагаемых, идентифицируемых с ea,fc, таким образом позволяя однозначно выделить e и c. Оставшиеся указатели используются для формирования текстформульных пояснений.

В заключение рассмотрим проверочный оператор из геометрии - оператор "однасторона" для усмотрения того, что две точки плоскости лежат по одну сторону от заданной прямой. В случае размещения точек по разные стороны имеется двойственный оператор "разныестороны". Оба оператора - сравнительно большие, имеют около сотни приемов каждый, и весьма часто используются в геометрических приемах. Выберем прием оператора "однасторона", усматривающий размещение двух точек по одну сторону от прямой из соображений двукратного перехода через эту прямую при движении по ломаной. Он расположен в разделе ("Элементарная геометрия", "Однасторона", "Проверочный оператор ОДНАСТОРОНА", "Двойной переход через прямую"). Опуская несложный чертеж, приведем сразу теорему приема:

$\forall_{ABCDEFG}(D \in$ отрезок(CE) & $F \in$ отрезок(CG) & $D \in$ прямая(AB) & $F \in$ прямая(AB) & разныеточки$(C, D) \to$ однасторона$(E, G,$ прямая$(AB)))$.

Здесь точка E является концом отрезка CE, на котором лежит точка D рассматриваемой прямой AB. Последний антецедент требует, чтобы точки C, D различались, так что либо точка E сама лежит на прямой, и тогда любая точка оказывается по

отношению к ней лежащей по ту же сторону от прямой (отношение "однасторона" нестрогое), либо точка E и точка C не лежат на прямой и находятся от нее по разные стороны. В любом случае конец G отрезка CG, пересекающегося с прямой в некоторой точке F, будет лежать с точкой E по одну сторону от прямой AB.

Прием имеет заголовок "спуск(однасторона)". Фильтр "уровень(3)" определяет уровень его срабатывания; фильтры "не(равно(C D))" и "не(равно(C F))" - ускоряющие. Первый из них отбрасывает случаи совпадения точек C, D еще до обращения к проверочному оператору "разныеточки". Второй - отбрасывает вырожденный случай совпадения точек C, F, в котором нет надобности применять данный прием, так как здесь точка E должна будет лежать на прямой AB, и сработает другой прием.

Указатель "усм(1 2 3 4)" определяет использование идентифицирующих операторов при обработке первых четырех антецедентов, указатель "блокпроверок(5)" - обращение к проверочному оператору для обработки последнего антецедента.

12.1.7 Приемы синтезаторов

Синтезатор представляет собой пакетный оператор для подбора значений одного или нескольких операндов заданного отношения, при которых это отношение становится истинным. Здесь возможна либо однократная выдача результата, либо перечисление серии результатов, отбираемых на основе каких-либо эвристических соображений приемами синтезатора. Как уже говорилось ранее, синтезатор является пакетным аналогом задачи на описание, причем для случая, когда эта задача имеет единственное условие - реализуемое синтезатором отношение. Свое название синтезатор получил из-за того, что обычно его прием сводит рассматриваемое отношение к одному или нескольким более простым отношениям, и после реализации их составляет ("синтезирует") свой ответ из полученных элементов.

В качестве первого примера синтезатора рассмотрим оператор "верхняяоценка", используемый для нахождения константных верхних оценок численных выражений. Реализуемое синтезатором утверждение имеет вид $a \le b$, причем переменная a рассматривается как входная, а переменная b - как выходная. Указатель "перечисление" в описании формата оператора определяет использование его в режиме перечисления верхних оценок. В действительности, такое перечисление обычно не очень велико и зависит от того, сколько различных способов оценивания выражения a оказалось предусмотрено в приемах синтезатора. Какая именно из версий найденных верхних оценок окажется достаточной - определяется внешним приемом, обратившимся к синтезатору.

Оператор "верхняяоценка" имеет четыре уровня срабатывания; он не очень велик - насчитывает около 50 приемов. Рассмотрим прием этого оператора, получающий верхнюю оценку для линейной комбинации синуса и косинуса. Он расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Синтезатор ВЕРХНЯЯОЦЕНКА", "Линейная комбинация синуса и косинуса"). Теорема приема имеет вид:

$$\forall_{abc}(a \sin b + c \cos b \le \sqrt{a^2 + c^2}).$$

Заголовок приема - "значение(верхняяоценка)". Прием имеет два фильтра - "константа(a)" и "константа(c)", так что применим только к линейным комбинациям синуса и косинуса, имеющим постоянные коэффициенты. При необходимости его легко обобщить на случай неконстантных коэффициентов, обратившись рекурсивным об-

разом к получению верхней оценки для радикала в правой части неравенства. Указатели и нормализаторы приема - стандартные.

Другой пример использования синтезатора - определение периода числовой функции. Заголовок синтезатора - "период"; реализуемое отношение - "периодична(a b)", истинное, если b есть какой-либо из периодов функции a. Приведем три простых приема этого синтезатора. Первый из них определяет период синуса; второй - отбрасывает при определении периода внешнюю одноместную операцию; третий - находит период результата двуместной операции. Эти приемы расположены в разделе ("Математический анализ", "Общий свойства числовых функций", "Периодичность", "Синтезатор определения периода ПЕРИОД") - соответственно, в подразделах "Синус и косинус"; "Одноместная операция - общий случай"; "Двуместная операция". Теорема первого из них имеет вид:

$$\forall_{abc}(\text{периодична}(\lambda_x(\sin(\frac{ax}{b}+c),\text{число}(x)),\frac{2\pi b}{|a|})).$$

Заголовок приема - "значение(период)"; единственный фильтр - "уровень(1)". Указатели и нормализаторы - стандартные. Этот прием относится к категории "завершающих" приемов, предназначенных для немедленной выдачи результата. Второй и третий приемы обеспечивают рекурсию, подготавливающую возможность применения завершающих приемов. Теорема второго приема имеет вид:

$$\forall_{fga}(\text{периодична}(\lambda_x(f(x),\text{число}),a)\rightarrow\text{периодична}(\lambda_x(g(f(x)),\text{число}),a)).$$

Указатель "символ(g)" определяет идентификацию переменной g с символом одноместной операции. Эта операция отбрасывается, и указатель "значения(1)" определяет рекурсивное обращение для определения периода "упрощенной" таким образом функции.

Теорема третьего приема имеет вид:

$\forall_{fgabcdmn}(\text{периодична}(\lambda_x(f(x),\text{число}(x)),a)$ & периодична$(\lambda_x(g(x),\text{число}(x)),b)$ & $a=nc/pd$ & $b=mc/qd$ & целое(n) & целое(m) & целое(p) & целое$(q)\rightarrow$ периодична $(\lambda_x(f(x)+g(x),\text{число}(x)),\text{нок}(mp,nq)c/pqd))$.

Для определения периода суммы двух функций определяются периоды a,b слагаемых, и если эти периоды соизмеримы, то находится их наименьшее общее кратное, которое и выдается в качестве результата. Указатель "вариант(фикс(0 1 3)умножение дробь степень)" обобщает этот прием на случай других двуместных арифметических операций. Фильтры "входит(x фикс(1 1 3))" и "входит(x фикс(2 1 3))" ограничивают применение приема лишь случаями, когда оба слагаемых содержат переменную x. В случае константного слагаемого применяется другой прием, просто отбрасывающий это слагаемое. Указатели "значения(1)" и "значения(2)" определяют рекурсивные обращения к нахождению периодов слагаемых. Прочие указатели приема и его нормализаторы - стандартные.

В заключение рассмотрим геометрический синтезатор "вычислениедлины", который используется для вычисления длины некоторого расстояния. Он несколько отличается от разобранных выше, так как результатом его применения служит не выражение для искомого расстояния, а лишь конъюнкция соотношений, из которых такое расстояние может быть найдено. Синтезаторы этого типа обычно используются в контексте вывода следствий при анализе некоторой системы утверждений. Фактически, синтезатор здесь лишь составляет план вычислений, а сами вычисления,

после перенесения найденной конъюнкции соотношений в список посылок текущей задачи на исследование, реализуются обычными приемами сканирования задачи. Это упрощает приемы синтезатора и делает планирование вычислений более гибким.

В качестве примера рассмотрим прием синтезатора, использующий теорему синусов для вычисления стороны треугольника по известным стороне и двум углам. Этот прием расположен в разделе ("Элементарная геометрия", "Расстояние", "Синтезатор ВЫЧИСЛЕНИЕДЛИНЫ", "Теорема синусов"). Берется первый прием раздела.

Опуская чертеж (изображение треугольника ABC), приведем сразу теорему приема:

$\forall_{ABC}$(актив($\angle(ABC)$) & актив($\angle(ACB)$) & актив($l(AB)$) $\rightarrow$ вычислениедлины($l(AC), \sin(\angle(ABC))l(AB) = \sin(\angle(ACB))l(AC)$)).

Заголовок приема - "значение(вычислениедлины)". Фильтр "уровень(3)" определяет уровень срабатывания. Фильтры "конец(известно(терм(расстояние(AB))))", "конец(известно(терм(угол($A\ B\ C$))))", "конец(известно(терм(угол($A\ C\ B$))))" указывают, что длина стороны AB и углы ABC, ACB известны. Размещение выражений "расстояние(...)", "угол(...)" под символом "терм" означает, что рассматриваться будут не сами эти выражения, а результаты обработки их нормализаторами приема. В данном случае это нормализаторы "нормрасстояние", "нормугол", которые не делают практически ничего, кроме усмотрения в контексте равенства, дающего своей правой частью выражение R для рассматриваемого расстояния либо угла, и замены преобразуемого терма на R. Обработка фильтров отнесена на конец программы приема - сначала выполняется идентификация антецедентов теоремы. На это указывает размещение фильтров под символом "конец".

Фильтры "усм(актив(прямая(AB)))", "усм(актив(прямая(AC)))", "усм(актив(прямая(BC)))" означают, что прямые для сторон треугольника должны быть введены в рассмотрение.

Указатель "усм(1 2 3)" определяет обработку первых трех антецедентов теоремы идентифицирующими операторами. Нормализаторы приема - стандартные.

12.1.8 Приемы анализаторов

Анализаторы представляют собой пакетные операторы для быстрого вывода следствий с определенной целью. По достижении данной цели вывод обрывается, и отобранные утверждения переносятся из накопителя анализатора в список посылок внешней задачи. Экономия от использования анализатора состоит в том, что каждое новое утверждение здесь рассматривается существенно меньшим числом приемов, чем это было бы при занесении его в список посылок задачи. В результате появляется возможность просматривать весьма значительные списки следствий за умеренное время; перенесение всех этих следствий в основную задачу привело бы к непомерному замедлению ее решения - эффекту "увязания" решателя в массиве малополезных длинных утверждений. Роль накопителя играет в анализаторе вспомогательная задача на исследование, копирующая список посылок текущей задачи.

В качестве примера рассмотрим, прежде всего, анализатор "сложениеуравнений", используемый для вывода следствий при решении систем уравнений в элементарной алгебре. Он имеет совсем немного приемов (около десятка). Включение анализатора предпринимается дважды - сначала на третьем уровне текущей задачи на исследование он активизируется для применения приемов нулевого и первого уровней;

затем, по достижении седьмого уровня сканирования задачи, активизируется уже до наибольшего своего уровня, равного двум.

Выберем прием этого анализатора, предпринимающий создание линейных комбинаций двух уравнений, имеющих "подобные" неизвестные члены. Этот прием расположен в разделе ("Элементарная алгебра", "Плюс", "Анализатор СЛОЖЕНИЕ-УРАВНЕНИЙ", "Линейная комбинация уравнений, имеющих неизвестные члены, отличающиеся лишь известными коэффициентами"). Теорема приема имеет вид:

$$\forall_{abcdefgh}(ag + b = c \ \& \ ah + d = e \ \& \ f = bh - dg - ch + eg \rightarrow f = 0).$$

Здесь первый и второй антецеденты суть исходные уравнения; коэффициенты g, h предполагаются в приеме не содержащими неизвестных, а выражение a - содержащим. Из второго уравнения, домноженного на g, вычитается первое, домноженное на h; все члены переносятся в левую часть, и определяется результат f обработки этой части различными нормализаторами (раскрывание скобок, и т.п.).

Заголовок приема - "внутрвывод(сложениеуравнений)". Фильтр "не(известно(a))" указывает, что общая часть двух подобных членов ag,ah должна содержать неизвестные. Фильтр "не(контекст(список(x9 b d)вид(x9 плюс(x10 дробь(x11 x12)))заменазнака(минус x11)единица(0 x10)))" означает, что уравнения не должны иметь дробных слагаемых. Обычно для уравнения с дробными слагаемыми в задаче на исследование выводится следствие - результат устранения у него знаменателей. Естественно при получении линейных комбинаций уравнений источники таких следствий отбрасывать.

Фильтр "не(заголовок(f 0))" отсекает случаи, когда рассматриваемые уравнения были пропорциональны. Фильтр "известно(h)" требует, чтобы коэффициент h не содержал неизвестных. Аналогичное требование для коэффициента g излишне, так как он изначально формируется в виде произведения всех известных множителей - согласно указателю "перечень(g известно(g))".

Последним размещен фильтр, сравнивающий новое уравнение с исходными. Он имеет вид "или($A_1A_2A_3$)", где A_1 - подфильтр "упрощение(x9 и(f фикс(1 1) фикс(2 1)) число(вид(x9 плюс(x10 x11)) единица(0 x11) не(известно(x10))))"; A_2 - подфильтр "упрощение(x9 и(f фикс(1 1) фикс(2 1)) числонеизвестных(x9))"; A_3 - подфильтр "и(входит(полный комментарии) контекст(неизвестная(x9) линейно(f x9)))". В этих подфильтрах указатели вхождения "фикс(1 1)" и "фикс(2 1)" ссылаются на левые части первого и второго исходных уравнений; f есть левая часть нового уравнения. Первый подфильтр означает, что левая часть каждого из исходных уравнений имеет строго больше неизвестных слагаемых, чем f. Второй означает, что левая часть каждого из исходных уравнений имеет большее число неизвестных, чем f. Наконец, третий означает, что при обращении к анализатору был введен комментарий "полный", причем результат f линеен хотя бы по одной из неизвестных задачи, входящих в f. Заметим, что комментарий "полный" вводится автоматически в тех случаях, когда максимальный уровень текущей задачи на исследование не менее 7. При решении систем алгебраических уравнений это означает вторую попытку вывода следствий из объединенного списка условий и посылок.

Указатели "единица(1 g h)" и "единица(0 d)" разрешают вырожденные единичные значения коэффициентов g, h и обращение в 0 суммы остальных членов левой части второго уравнения. Левая часть хотя бы одного (пусть оно и будет первым) уравнения должна иметь хотя бы два неизвестных слагаемых - иначе результирующее равенство не будет содержать неизвестных. Указатель "идентификатор(3)"

определяет использование третьего антецедента для вычисления f. Указатель "заменазнака(минус g h)" разрешает отнесение минуса, если он есть, к соответствующему коэффициенту g, h.

Указатель "перечень(g известно(g))" определяет идентификацию коэффициента g из совокупности всех сомножителей, не содержащих неизвестных. Указатель "набороперандов(фикс(1 1 1)фикс(2 1 1))" введен для ускорения идентификации. Он отменяет нахождение общей неизвестной части a двух слагаемых ag, ah с использованием трудоемкого оператора "алгебрпересечение", находящего наибольший общий делитель двух одночленов, и a идентифицируется как пересечение двух групп сомножителей этих слагаемых. Так как g, h не должны содержать неизвестных, то при отсутствии неизвестных показателей степеней это не ограничивает общности идентификации.

Указатели вида "обрыв(...)" используются для формулировки условий обрыва работы анализатора. При этом только что выведенное утверждение снабжается комментарием "внешвывод", и происходит передача всех выделенных таким комментарием посылок анализатора во внешнюю задачу на исследование. В данном приеме имеем следующие три условия обрыва вывода. Во-первых, это "обрыв(равно(числонеизвестных(f)1))" - получение уравнения с единственной неизвестной. Во вторых, "обрыв(контекст(вид(f плюс(умножение(x9 степень(x10 x13)) умножение(x11 степень(x12 x13)))) единица(1 x9 x11 x13) заменазнака(минус x9 x11) неизвестная(x10) неизвестная(x12) известно(x9) известно(x11) известно(x13)))" - получение уравнения, выражающего пропорциональность двух неизвестных X, Y: $pX^n + qY^n = 0$; p, q, n - известны. В-третьих, "обрыв(заголовок(результат или))" - при обработке нормализаторами левой части нового уравнения ее удалось разложить на несколько неизвестных множителей, что привело к преобразованию этого уравнения к виду дизъюнкции более простых уравнений.

Указатель "прообраз(1 посылки(не(равно(h 0))))" нужен для регистрации в комментариях того факта, что первое уравнение, в случае ненулевого коэффицента h, является следствием второго исходного уравнения и нового уравнения. Эта информация может оказаться полезной, если при выводе следствий в блоке анализа будут найдены значения неизвестных. Тогда будет предпринята попытка установить с ее помощью, что текущие уравнения задачи на описание являются следствиями равенств R, определяющих значения неизвестных, а также, быть может, ряда других простых сопровождающих утверждений U блока анализа. В этом случае проверка текущих уравнений задачи на описания будет излишней, и они просто будут заменены на утверждения R, U. Указатель "прообраз(2 посылки(не(равно(g 0))))" используется аналогичным образом для второго уравнения.

Указатель "примечание(условие(и(не(контекст(неизвестная(x9) список(x10 фикс(1 1)фикс(2 1)) линейно(x10 x9))) контекст(неизвестная(x9) линейно(f x9))) внешвывод))" помечает новое уравнение комментарием "внешвывод", если оно оказывается линейным по некоторой неизвестной, по которой оба исходных уравнения были нелинейны. Такой комментарий предопределяет перенесение данного уравнения в список посылок внешней задачи на исследование по окончании работы анализатора, без немедленного ее обрыва.

Следующий указатель "примечание(условие(и(контекст(неизвестная(x9) список(x10 фикс(1 1) фикс(2 1)) не(линейно(x10 x9))) не(контекст(неизвестная(x9) не(линейно(f x9))))) внешвывод))" - аналогичен. Он помечает новое уравнение комментарием "внешвывод", если оно линейно по всем своим неизвестным, а хотя бы одно из исходных уравнений - не линейно по какой-либо неизвестной.

Правая часть третьего антецедента теоремы обрабатывается нормализаторами "нормплюс", "стандплюс", "уравнплюс". Первые два из них обеспечивают раскрывание скобок в линейной комбинации; третий - приведение подобных членов с известными коэффициентами. После определения f оно обрабатывается нормализатором "факторизация" упрощенного разложения на множители. Наконец, само новое уравнение $f = 0$ обрабатывается нормализатором "нормчисло", который, в частности, может преобразовать его к виду дизъюнкции, если левая часть имеет вид произведения.

Следующий пример анализатора - оператор "смплощадь", используемый в геометрии для вывода соотношений, связывающих между собой площади выделяемых на чертеже фигур. В основном, здесь применяются приемы, составляющие площадь фигуры из площадей подфигур, а также приемы, усматривающие пропорциональность площадей. Анализатор имеет 5 уровней срабатывания и около сорока приемов. Рассмотрим один из простейших его приемов, разбивающий треугольник ABC на два подтреугольника ABD,BDC, если вершина B соединена отрезком с точкой D на стороне AC, причем известно, в каком отношении эта точка делит сторону AC. Прием расположен в разделе ("Элементарная геометрия", "Фигуры", "Площадь", "Анализатор СМПЛОЩАДЬ", "Разбиение треугольника на два треугольника линией, проведенной из вершины к противоположной стороне"). Берется первый из приемов раздела. Опуская несложный чертеж, приводим сразу теорему приема:

$\forall_{ABCDab}$(актив(S(фигура(ABC))) & $D \in$ отрезок(AC) & $al(AD) = bl(CD)$ & $\neg(a + b = 0) \rightarrow$ актив(S(фигура(ABD))) & актив(S(фигура(BCD))) & S(фигура(ABD)) = bS(фигура(ABC))$/(a + b)$ & S(фигура(ABC)) = S(фигура(ABD)) + S(фигура(BCD))).

Заметим, что соотношения пропорциональности в теоремах геометрических приемов обычно записываются так, как это сделано в третьем антецеденте данной теоремы, т.е. $al(AD) = bl(CD)$ вместо $l(AD)/l(CD) = b/a$. Это объясняется, во-первых, тем, что общая стандартизация посылок приводит к исключению дробей в равенствах, определяющих пропорциональность величин. Во-вторых, для усмотрения пропорциональности используется синтезатор "пропорциональны", определяющий коэффициенты пропорциональности a, b по заданным величинам A, B, и его входной шаблон имеет вид с исключенными дробями - $aB = bA$. В качестве допустимого результата этот синтезатор может выдавать только выражения a, b, не имеющие нечисловых переменных, причем каждая переменная этих выражений должна быть либо известным параметром задачи на исследование, либо неизвестной внешней задачи на описание.

Первый антецедент означает, что площадь треугольника ABC уже входила в какую-либо из посылок. В этой ситуации срабатывает прием, вводящий вспомогательную посылку "актив(площадь(...))", чтобы упростить последующие ссылки на явно рассматриваемые в задаче площади. Теорема приема вводит в рассмотрение площади треугольников ABD,BCD и регистрирует в посылках анализатора соотношения, связывающие их с площадью треугольника ABC.

Прием имеет заголовок "внутрвывод(смплощадь)". Его фильтр "комментпосылки(1 смплощадь D)" нужен для предотвращения повторного срабатывания с тем же самым разбиением треугольника ABC. После применения данного приема вводится комментарий (смплощадь D) к посылке "актив(S(фигура(ABC))", - это делает указатель "примечпосылки(1 смплощадь D)", - и повторные срабатывания оказываются заблокированы.

Фильтры "не(равно(D A))", "не(равно(D C))" отсекают неинтересные случаи совпадения точки D с концами отрезка AC. Наконец, фильтр "усм(актив(прямая(B D)))" требует, чтобы прямая, соединяющая точки B, D, уже была введена в рассмотрение. Заметим, что в том же самом разделе имеется версия данного приема, у которой такая прямая еще не ведена в рассмотрение; тогда вводится ряд дополнительных фильтров, и уровень срабатывания приема повышается.

Указатель "уровень(1)" определяет первый уровень срабатывания анализатора. Указатель "значения(3)" определяет использование синтезатора "пропорциональны" для определения коэффициентов a, b в третьем антецеденте. Перед обращением к нему выражения $l(AD), l(CD)$ обрабатываются нормализатором "нормрасстояние". Указатель "блокпроверок(4)" определяет использование проверочного оператора для проверки отличия $a + b$, которое далее выступает как знаменатель, от нуля. Указатель "усм(2)" определяет обработку второго антецедента идентифицирующим оператором.

Указатель "список(фикс(1 1 1 1))" нужен для того, чтобы треугольник ABC идентифицировался с произвольным порядком перечисления его вершин в первом антецеденте. Заметим, что в скобочном представлении этот антецедент имеет вид "актив(площадь(фигура(набор(A B C))))", так что указатель вхождения "фикс(1 1 1 1)" относится к подвыражению "набор(...)".

Указатели "копия(фикс(0 1 1))", "копия(фикс(0 2 1))" отменяют обработку нормализатором "нормплощадь" выражений S(фигура(ABD)), S(фигура(BCD) в утверждениях "актив(...)" правой части теоремы. Если бы эти площади были выражены через какие-либо параметры задачи, то утверждения "актив(...)", как отметки о рассмотрении площадей конкретных треугольников, оказались бы испорчены.

Указатель "титр(...)", обеспечивающий текст-формульное сопровождение срабатывания приема, и нормализаторы приема - стандартные.

12.1.9 Операторы фильтра

Операторы фильтра используются в решателе крайне редко, и общее число их едва ли насчитывает сейчас полтора десятка. Одна из веских причин прибегнуть к оператору фильтра - появление чрезмерно большой дизъюнкции, перечисляющей подслучаи, в которых применение приема допустимо. Помимо того, что эта дизъюнкция может не поместиться в окне описания приема, успешная компиляция ее также не гарантирована. В то же время, оператор фильтра позволяет компилировать программы проверки отдельных подслучаев по отдельности, и каких-либо существенных ограничений на число его приемов нет. В качестве примера рассмотрим оператор фильтра "фильтркквадратов", используемый для принятия решения о переходе от произведения суммы двух выражений на их разность к разности квадратов этих выражений. Он имеет всего 4 приема. Первый из них проверяет наличие внешней дроби, у которой переход к разности квадратов дает возможность сокращения. Второй - проверяет, не являются ли слагаемые константами, хотя бы одна из которых - радикал четной степени. Третий прием проверяет, не является ли заменяемое вхождение корневым, и четвертый - проверяет, не приводит ли данное преобразование к устранению нигде более не встречающихся радикалов. При срабатывании хотя бы одного из приемов оператор получает значение "истина". Рассмотрим запись первого приема (см. раздел "Элементарная алгебра", "Умножение", "Усмотрение разности квадратов", "Определение целесообразности перехода к разности квадратов", "Возможность сокращения дроби при переходе к разности квадратов").

Приемы операторов фильтра имеют фиктивную теорему - логический символ "блок". Однако, эта теорема на экран не выводится (чтобы посмотреть на нее, нужно войти в режим редактирования теоремы текстовым редактором, нажав Ctr-т). Вместо теоремы выводится текст первого пункта оглавления для раздела оператора фильтра - он содержит напоминание о том, какие переменные определены на момент обращения к оператору и что они означают. В данном случае имеем текст: "x1,x2 - вхождения суммы и разности выражений x3,x4. x5 - результат нормализации разности квадратов".

Прием имеет заголовок "контекст(фильтркводратов)". Его единственный фильтр - "подтерм(дробь(умножение(x6 теквхожд(x1))умножение(x5 x7)))". Заметим, что для фильтров приема оператора фильтра не обязательно размещение их внутри конструкции "контекст(...)" - здесь она играет роль отбрасываемых по умолчанию внешних скобок. Компилятор восстанавливает эти внешние скобки автоматически. Поэтому указанный выше фильтр "подтерм(...)" и не помещен внутри "контекст'а", хотя он вводит новые переменные x6,x7. Смысл этого фильтра - усмотрение внешней дроби, у которой преобразуемые выражения служат множителями числителя, а знаменатель имеет множитель x5, т.е. разность квадратов. Указатель "дробь" (фактически возникший из-под отброшенного "контекст'а") разрешает менять в этой идентификации местами числитель и знаменатель. Указатель "единица(1 x7)" разрешает отсутствовать множителю x7.

12.2 Упражения по работе с редактором приемов

12.2.1 Поиск приемов

Приемы решателя распределены по разделам подробного оглавления, устроенного по тематическому принципу. Ссылка на каждый прием осуществляется из единственного пункта оглавления, причем иногда бывают возможны различные естественные решения о выборе такого пункта. Поэтому для быстрой ориентации в накопленном материале полезна некоторая тренировка. Приведем ряд простых упражнений по поиску нужного приема либо раздела.

1. Найти корневое меню оглавления базы приемов. Найти разделы "Тригонометрия"; "Окружность"; "Эллипс"; "Вычисление определенных интегралов"; "Признаки сходимости знакопеременных рядов"; "Степень матрицы"; "Уравнение в полных дифференциалах"; "Суммы и разности мощностей множеств"; "Закон Пуассона"; "Движение по наклонной плоскости".

2. Найти приемы логарифмирования показательных уравнений. Выбрать те из них, у которых уравнение имеет ноль в правой части, а основанием логарифма служит основание неизвестной степени.

3. Найти приемы решения строгих квадратных неравенств. Объяснить назначение каждого из них.

4. Найти приемы, выводящие соотношение пропорциональности для длин сторон треугольника и длин отрезков, на которые биссектриса делит сторону треугольника. Определить отличие контекстов, в которых срабатывают эти приемы.

5. Найти приемы, выражающие площадь треугольника по формуле Герона. Объяснить контексты срабатывания этих приемов.

6. Найти прием проверочного оператора "усмменьшеилиравно", усматривающий неотрицательность синуса угла, заключенного между 0 и пи.

7. Найти прием для доказательства индукцией по натуральному параметру.

8. Найти приемы для вычисления суммы геометрической прогрессии.

9. Найти приемы интегрирования с помощью подстановок Эйлера.

10. Найти прием вычисления математического ожидания по известной плотности распределения случайной величины.

11. Найти все приемы вывода, теорема которых содержит символы "биссектриса" и "трапеция".

12. Найти все приемы, вводящие комментарий "промежуток" к текущему терму задачи.

Указания

1. Из главного меню системы нажатием клавиши "г" войти в оглавление приемов. Затем нажимать клавишу "курсор влево" до тех пор, пока картинка не перестанет изменяться. Это и будет корневое меню оглавления базы приемов. Дальнейший поиск ведется сообразно названиям разделов - например, к разделу "Тригонометрия" через пункт "Элементарная алгебра". Раздел "Окружность" можно найти как в подразделе "Элементарная геометрия" (через подраздел "Фигуры"), так и в разделе "Аналитическая геометрия" (через подраздел "Линии второго порядка"). Естественно, приемы последнего раздела связаны только с уравнением окружности.

2. Нужные приемы расположены в подразделе ("Элементарная алгебра", "Степени", "Решение уравнений", "Показательные уравнения", "Логарифмирование показательных уравнений"). В этом разделе имеется около десяти различных приемов. При записи нового приема в раздел он попадает в начало раздела, поэтому просмотр приемов раздела "в хронологическом" порядке нужно начинать с конца раздела, перейдя к нему нажатиями клавиши "курсор вниз" - пока картинка не перестанет изменяться.

   Наличие нескольких различных приемов логарифмирования уравнения объясняется следующими причинами. Во-первых, по-разному можно выбирать основание логарифма - иногда это основание одной из имеющихся в уравнении степеней с неизвестным показателем, иногда - основание другого логарифма, уже имеющегося в задаче, иногда, если из прочих соображений разумного основания выбрать не удается, просто константа 2. Далее, по-разному может происходить группировка членов в уравнении. Если есть всего два неизвестных слагаемых, то оба они группируются в левой части, а правая часть - нулевая. Если одно слагаемое неизвестное, а другое - известное, то они находятся в разных частях уравнения. Наконец, выделено несколько особо простых случаев, для которых предусмотрены специальные приемы, срабатывающие на малом уровне.

   Приемы, которые требуется найти в упражнении - это второй и седьмой приемы, считая с конца.

3. Приемы находятся в разделе ("Элементарная алгебра", "Неравенства", "Меньше", "Решение неравенств", "Квадратные неравенства"). Их всего шесть. Просмотр их начинаем, как обычно, с конца списка.

   Первый и второй приемы относятся к случаю, когда вычисленный дискриминант d оказался отличным от нуля, причем не удалось усмотреть его отрицательность. В первом приеме квадратный двучлен расположен справа от знака "меньше", во втором - слева от этого знака. Заметим, что известный свободный член при стандартизации неравенств попадает в противоположную часть неравенства. Оба приема совершенно аналогичны. Иногда в таких случаях удается создать один общий прием, используя указатели идентификации. Однако, не всегда для этого хватает возможностей текущей версии ГЕНОЛОГа. С другой стороны, интерфейс предоставляет возможность быстрого "тиражирования" уже введенной версии приема - она копируется, и вводятся необходимые изменения. Таким образом могут появляться достаточно большие группы (десяток и более) однотипных приемов. Надобность в этом "тиражировании" возникает не очень часто, а на быстродействии решателя оно практически никак не отражается, так что особых стимулов в развитии указателей ГЕНОЛОГа для борьбы с ним пока не имеется.

   Третий и четвертый приемы относятся к случаю, когда удалось усмотреть равенство дискриминанта нулю; пятый и шестой - к случаю отрицательного дискриминанта. Хотя во всех этих приемах дискриминат вычисляется как бы заново, но в действительности он будет вычислен лишь однократно, а затем занесен в буфер, и все последующие приемы на протяжении достаточно большого отрезка времени будут брать его прямо из буфера. Это же замечание относится и к проверке знака дискримината - результаты ее (в том числе при неудаче) тоже хранятся в буфере.

4. Приемы расположены в разделе ("Элементарная геометрия", "Фигуры", "Треугольник", "Отрезок, соединяющий вершину треугольника с точкой на противоположной стороне или ее продолжении", "Биссектрисы треугольника", "Отрезки, на которые основание биссектрисы делит противоположную сторону, пропорциональны боковым сторонам"). Всего имеется три таких приема. Первый прием (считая с конца) анализирует четыре участвующих в соотношении пропорциональности величины - длины боковых сторон и длины отрезков, на которые сторона делится биссектрисой. Если не менее трех из них выражены через числовые параметры, известные или неизвестные, то уровень срабатывания приема равен 6; в противном случае прием срабатывает, если две из них известны, но уровень срабатывания тогда повышается до 8. Второй прием срабатывает, если удается усмотреть соотношение пропорциональности с известными коэффициентами для длин боковых сторон; уровень срабатывания его равен 4. Третий прием аналогичен второму, но соотношение пропорциональности усматривается для длин отрезков, на которые сторона делится биссектрисой.

5. Приемы находятся в разделе ("Элементарная геометрия", "Фигуры", "Площадь", "Треугольник", "Формула Герона"). Число их равно трем. Первый прием срабатывает, если все три длины сторон треугольника известны, а соотношение для его площади еще не выписывалось; уровень срабатывания равен 6. Второй срабатывает, если выражения a, b, c для длин сторон либо содержат неизвестную y внешней задачи, входящую также в некоторое соотношение

с площадью треугольника, либо известны, а указанная неизвестная y входит в соотношение с площадью треугольника. Наконец, третий срабатывает, если площадь треугольника известна, а выражения для длин сторон пропорциональны некоторой неизвестной.

6. Прием находится в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Проверочный оператор УСММЕНЬШЕИЛИРАВНО", "Тригонометрические функции", "Синус"). Он является третьим с конца списка. Фильтр этого приема - ускоряющий; он разрешает попытку применения приема лишь тогда, когда в посылках имеется обозначение какого - либо угла либо имеется неравенство, содержащее хотя бы одну общую переменную с аргументом синуса.

7. Прием находится в разделе ("Элементарная алгебра", "Целые числа", "Доказательство индукцией по целочисленному параметру", "Простейшая схема индукции по натуральному параметру").

8. Приемы расположены в разделе ("Элементарная алгебра", "Комбинаторные функции", "Суммавсех", "Сумма геометрической прогрессии и суммы, извлекаемые из нее почленным дифференцированием"). Первый из них (с начала списка) дает общую формулу суммирования, включающую случай равенства знаменателя прогрессии единице; второй - рассчитан на случаи, когда усматривается отличие этого знаменателя от 1.

9. Приемы расположены в разделе ("Математический анализ", "Интегралы", "Нормализатор нахождения первообразной НОРМИНТЕГРАЛ", "Замена переменной интегрирования", "Степенная функция", "Квадратичная функция под радикалом"). Для каждой из подстановок Эйлера выделен свой подраздел.

10. Прием находится в разделе ("Теория вероятностей", "Случайные величины", "Математическое ожидание", "Вычисление математического ожидания случайной величины, для которой известна плотность распределения"). Он является приемом вывода, заносящим в посылки равенство для математического ожидания. Если бы этот прием был реализован как прием замены, то при возникновении того же самого математического ожидания в той же задаче (например, при вычислении дисперсии) выкладки пришлось бы повторять.

11. Помимо оглавления базы приемов, для нахождения нужных приемов можно использовать реализованную на ЛОСе процедуру "текприем". Собственно говоря, программа этой процедуры представляет собой бланк запроса на поиск приемов, и для очередного поиска переписывается. Обращения к ней происходят в цикле полного просмотра всех приемов решателя, реализованных на ГЕНОЛОГе. При рассмотрении очередного приема процедуре сообщаются следующие данные: x1 - логический символ, за которым закреплен прием; x2 - номер узла статьи этого символа в 8-м информационном блоке, хранящем описания приемов ГЕНОЛОГа; x3 - теорема приема; x4 - заголовок приема; x5 - описание приема. Кроме того, ей сообщаются некоторые дополнительные сведения о хранении компонент описания приема в 8-м информационном блоке. Подробнее о том, как организовано это хранение, будет рассказано в следующей главе.

    Программа процедуры "текприем" должна начинаться с операторов "обращение(0) иначе 1 метка(икс(десять))", которые при написании очередного запроса

на поиск изменять не следует. Заканчиваться эта программа должна оператором "ответ(набор(См))", обращение к которому вызывает прерывание и выход в просмотр описания текущего приема. После этого продолжение поиска вызывается нажатием клавиши "курсор влево". Если же нужно прервать поиск и начать работу с найденным приемом, то нажимается клавиша "Esc", выводящая в главное меню, далее нажимается клавиша "г", возвращающая в оглавление базы приемов, причем в том именно концевой пункт, где находится найденный прием, и нажимается "курсор вправо" для входа в просмотр этого приема.

В нашем случае нужно отобрать приемы, теорема которых содержит символы "трапеция" и "биссектриса". Для этого входим в просмотр программы ЛОСа для логического символа "текприем" (войти в просмотр и редактирование этой программы можно также нажатием F9 из любой точки оглавления базы приемов). Нажимаем "р" (кир.) и набираем текстовым редактором, после оператора "метка(...)", операторы "не(логсимвол(х3)) входит(трапеция х3) входит(биссектриса х3)". Затем должен быть размещен указанный выше оператор "ответ(...)". Заметим, что значением переменной х3 служит теорема приема, однако для приемов операторов фильтра вместо нее используется логический символ "блок". Чтобы избежать замечаний о некорректном обращении к оператору, перед проверкой вхождения в х3 какого - либо символа, сначала нужно убедиться, что х3 действительно является термом, что и делает оператор "не(логсимвол(х3))".

По окончании ввода программы, завершаемого, как обычно, нажатием Enter, для возвращения в оглавление базы приемов (если мы вышди из нее по F9), достаточно нажать "End". Далее поиск активируется нажатием F8 из любого пункта оглавления. В нашем случае будут найдены три приема. По завершении поиска произойдет выход в отладчик ЛОСа на операторы "трассировка(стоп 0) трассировка(0)"; далее для возвращения в главное меню нажимается Esc.

12. Как и в предыдущем случае, используем процедуру "текприем". Примечание к текущему терму задачи вводится указателем приема "примечание(...)". Поэтому в программе "текприем" помещаем единственный (кроме стандартных начала и конца программы) оператор - "входит(запись(примечание промежуток) х5)". Напомним, что х5 - набор указателей приема, к которому добавлен элемент "условие(и($F_1 \dots F_n$))", перечисляющий все фильтры приема $F_1, \dots, F_n$. При просмотре обнаруживается единственный прием, вводящий комментарий "промежуток".

Заметим, что процедуру "текприем" можно использовать не только для поиска приемов нужного типа, но и для автоматического изменения всей базы приемов, согласно заданной программе. Это позволяет предпринимать различные глобальные преобразования решателя при его оптимизации. Впрочем, после любого такого преобразования необходима полная прогонка по задачнику - для выявления остаточных случаев, требующих дополнительной проработки. Часто эти случаи возникают из-за небольших изменений траектории решения, приводящих к необходимости применять отсутствующий пока прием. Вообще, даже обычная перекомпиляция нескольких приемов не в том порядке, в каком они были скомпилированы изначально, может (хотя и достаточно редко) изменить поведение решателя. Разумеется, по мере развития исследований по взаимодействию между приемами устойчивость поведения решателя (в том числе и

по отношению к случайным изменениям порядка размещения приемов после перекомпиляций) будет повышаться. Даже сейчас она не так уж мала. Однако, единичные случаи потери или замедления решений задач могут возникать, и при перекомпиляции лучше либо сохранять первоначальное размещение приемов, либо убеждаться прогонкой по задачнику, что нежелательные ситуации не возникли.

12.2.2 Просмотр описания приема

Первые упражнения этого раздела относятся к приему решения квадратного уравнения. Поэтому перед их выполнением следует найти данный прием в оглавлении базы приемов, нажать клавишу "курсор вправо", и выбрать последний из приемов списка.

1. Перейти от формульного режима просмотра теоремы к текстовому и обратно.

2. Выделить цветовой указкой в теореме приема левую часть эквивалентности. Убрать выделение, перейти в текстовый режим просмотра, и в нем снова выделить цветовой указкой данную часть.

3. Активизировать для просмотра последовательно первое, второе, третье и четвертое окна приема.

4. Убрать с экрана первое окно приема, затем восстановить его на экране.

5. Выделить многоцветной указкой фильтр "альтернатива(...)". Внутри этого фильтра выделить терм "неизвестные(2)". Прочитать пояснения к символу "неизвестные". Прочитать пояснения к подтерму "числонеизвестных(корень)", встречающемуся в фильтрах приема. Продолжить рассмотрение фильтров и указателей приема, получая информацию о всех встречающихся в них служебных словах ГЕНОЛОГа.

6. Просмотреть нормализаторы к подтерму $0 \leq e$ теоремы приема. Просмотреть все нормализаторы приема.

7. Просмотреть текстформульный шаблон пояснений к срабатыванию приема. Найти ссылки на его фрагменты из различных точек описания приема.

8. Прочитать пояснения к заголовку приема. Войти в оглавление заголовков приемов ГЕНОЛОГа и найти в нем тот же поясняющий текст.

9. Войти в оглавление типов фильтров ГЕНОЛОГа и найти в нем несколько типов фильтров, использованных в приеме. Аналогичное упражнение проделать для оглавления типов указателей приемов.

10. Из просмотра приема перейти в просмотр справочной информации о логическом символе "целое", затем вернуться обратно.

11. Перейти к просмотру программы приема. Прочитать несколько первых операторов этой программы и объяснить смысл выполняемых ими действий. Вернуться к просмотру описания приема.

12. Перейти из просмотра описания приема в оглавление задачника, затем снова вернуться к просмотру описания приема. Аналогичное упражнение - для перехода в оглавление программ.

13. Найти последний из приемов раздела "Соотношение пропорциональности длин отрезков, отсекаемых параллельными прямыми". Просмотреть все фильтры этого приема, учитывая то, что список их разбит на несколько последовательно выдаваемых на экран фрагментов.

Указания

1. Для перехода к текстовому режиму нажимаем "т", для перехода к формульному - "ф". Текущий режим будет сохранен при переходе через оглавление базы теорем к другому приему. Однако, при возвращении в главное меню установка на него будет забыта, и при последующем переходе к просмотру приемов будет, по умолчанию, выбран формульный режим.

2. Чтобы выделить цветовой указкой часть теоремы, нужно сначала инициировать ее просмотр - либо нажать левую кнопку мыши, курсор которой находится в первом окне, либо нажать клавишу "1", либо нажать два раза клавишу "курсор вправо". Первое нажатие клавиши выделит малиновым цветом номер "1" первого окна; второе - переведет в режим просмотра теоремы. Сначала малиновым цветом будет выделен первый антецедент. Используя клавишу "курсор вправо", выделим сначала весь консеквент. Затем нажмем "курсор вниз", выделяя первый операнд консеквента, что и требуется. Для сброса режима просмотра теоремы нажимаем клавишу пробела. Чтобы перейти в текстовый режим просмотра, нажимаем далее "т". После этого можно либо войти в режим просмотра теоремы двумя нажатиями "курсор вправо", и далее манипулировать клавишами курсора для выделения нужного подтерма, либо сразу подвести курсор мыши к началу нужного подтерма теоремы и нажать левую кнопку мыши. Для сброса режима просмотра снова нажимаем клавишу пробела.

3. Для входа в режим просмотра i - го окна; $i = 1, 2, 3, 4$, можно нажать клавишу "i" номера этого окна. Другой способ - нажать сначала клавишу "курсор вправо". Тогда будет выделен малиновым цветом номер "1" первого окна. С помощью клавиш "курсор вверх" - "курсор вниз" можно менять текущее окно. Однако, выделение номера окна - это еще не вход в режим просмотра окна. Для перехода к такому просмотру нужно, выделив текущее окно, нажать клавишу "курсор вправо". Заметим, что для второго окна, состоящего только из заголовка приема, надобности в просмотре с применением цветовой указки не возникает, и в этом случае нажатие "курсор вправо" игнорируется. При входе в просмотр 1,3 либо 4 окон происходит выделение первого терма цветовой указкой (для первого окна выделяется малиновым цветом первый антецедент теоремы, для 3 и 4 - светло-голубым цветом выделяется первый фильтр либо первый указатель). Для выхода из просмотра нажимается клавиша пробела.

4. Если описание приема не помещается на экране целиком, то можно убирать некоторые из окон. Для этого служат клавиши Ctr-Fi; i - номер окна, $i = 1, 2, 3, 4$. Восстановление убранных окон - нажатием той же клавиши, либо за счет выхода в оглавление базы приемов и повторного входа в просмотр приема.

5. Есть две возможности - либо выделить слово "альтернатива" курсором мыши и нажать ее левую кнопку, либо войти в просмотр третьего окна и выделить фильтр "альтернатива(...)", используя клавишу "курсор вправо". Подтерм "неизвестные(2)" теперь можно выделить, нажав клавишу "курсор вниз", либо переведя курсор мыши на слово "неизвестные" и нажав ее левую кнопку. Мышью можно было выделить этот подтерм и сразу, не выделяя "альтернатива(
...)".

   Чтобы прочитать пояснения к выделенному терму, нажимаем либо правую кнопку мыши (безотносительно к тому, использовалась ли мышь при выделении), либо клавишу "Ctr-y" (кир.). Снизу, под четвертым окном, появляется текст пояснений. Иногда одно и то же служебное слово используется в различных контекстах, и тогда можно просмотреть пояснения ко всем возможным его использованиям. Сначала выдается какой - то один текст; для смены текстов списка используются клавиши "курсор вверх" - "курсор вниз". В нашем случае сначала будет выдан текст пояснений к использованию "неизвестные(x1)" в качестве фильтра, уточняющего число неизвестных задачи. После нажатия "курсор вниз" появится другой текст - "неизвестные(x1)" может служить обозначением списка неизвестных задачи. Обычно такая неоднозначность легко устраняется из контекста. Очевидно, "неизвестные(2)" подпадает под первый текст пояснений.

   Чтобы убрать с экрана тексты пояснений, нажимаем либо произвольную кнопку мыши, либо клавишу пробела. После этого выделяем слово "числонеизвестных" курсором мыши и нажимаем правую ее кнопку. Сначала появится текст "Число неизвестных задачи"; после нажатия "курсор вниз" - "Число неизвестных в терме, определяемом выражением x1". Так в нашем случае символ "числонеизвестных" относится к выражению "корень", то берем второй поясняющий текст. Можно посмотреть пояснения к слову "корень" - будут выданы тексты "Заменяемый подтерм является корневым" и "Текущий терм задачи". Первый из них явно относится к случаю использования слова "корень" как фильтра, у нас же оно используется как выражение. Поэтому берем второй текст. Окончательно получаем, что "числонеизвестных(корень)" означает число неизвестных в текущем терме задачи, т.е. в рассматриваемом квадратном уравнении.

   Чтобы привыкнуть к простейшим конструкциям ГЕНОЛОГа, можно рекомендовать последовательный просмотр фильтров и указателей приема с выделением их левой клавишей мыши и вызывом сопровождающих текстов правой клавишей.

6. Чтобы просмотреть нормализаторы к заданному вхождению в теорему, фильтр либо указатель, сначала нужно выделить это вхождение. Заметим, что не все вхождения в теорему могут быть выделены при формульном режиме. Поэтому в сомнительных случаях (они бывают сравнительно редко и связаны с различными специальными логическими конструкциями) лучше переходить в текстовый режим просмотра теоремы. В нашем примере можно выделить $0 \leq e$ и непосредственно из формульного просмотра.

   После того, как нужное вхождение выделено, нажимается Enter. Тогда под четвертым окном возникает текст нормализаторов, относящихся к выделенно-

му вхождению. При этом появляется также курсор текстового редактора, чтобы при необходимости изменить нормализаторы. Если вхождение не сопровождалось нормализаторами, то курсор текстового редактора возникает на пустой строке. Так как в данном упражнении не нужно изменять нормализаторы, то после просмотра нажимается Esc либо Enter (последнее - только при отсутствии изменений в тексте).

Чтобы последовательно просмотреть все нормализаторы приема, следует нажать клавишу "5", и далее менять текущий выделенный подтерм клавишами "PageUp" - "PageDown". Как и ранее, нормализаторы будут прорисовываться под четвертым окном. Выход из такого просмотра - только по нажатию "End" (!).

7. Просмотр текстформульного шаблона вызывается нажатием клавиши "6". Этот шаблон оказывается состоящим из двух фрагментов: "¬1 Решаем квадратное уравнение относительно (). Дискриминат равен ()"; "¬2 Разложение на множители дискриминанта". Очевидно, первый из них комментирует срабатывание приема "в целом", два последних - комментируют вспомогательные преобразования, выполняемые приемом. Для первого легко находится относящийся к нему указатель "титр(набор(1 терм(x4 x5)))". Согласно этому указателю, вместо первых скобок будет подставляться неизвестное выражение d, вместо вторых - результат e вычисления дискриминанта. Второй текст указывает на обращение к нормализатору разложения на множители дискриминанта $b^2 + 4ac$. Выделяя в теореме его вхождение, просматриваем список нормализаторов. В этом списке находим элемент "видумножение(замечание(нормстепень) замечание(новый) замечание(титр(набор(2))))", содержащий ссылку на второй фрагмент.

8. Для просмотра пояснений к заголовку нужно выделить номер "2" окна заголовка малиновым цветом и нажать "Ctr-з" (кир.). Пояснение появится под четвертым окном. Для входа в оглавление типов заголовков нажимаем ту же клавишу "Ctr-з", но без выделения второго окна приема. Находим корень оглавления, и от него прослеживаем цепочку переходов ("Замена подтерма", "Замена слева направо"). Нажимая далее "курсор вправо", находим соответствующий заголовок "второйтерм".

9. Для входа в оглавления типов фильтров и типов указателей используем, соответственно, клавиши "Ctr-у" (кир.) и "Ctr-х". Заметим, что эти оглавления немного более полны, чем приведенные в книге списки типов фильтров и указателей, так как они постоянно пополняются в процессе развития языка.

10. Для получения общей справочной информации о логическом символе нажимаем клавишу "с", и далее вводим текстовым редактором этот логический символ. После нажатия Enter, завершающего ввод символа, появляется требуемая информация. По нажатии любой клавиши, отличной от используемых для просмотра информации служебных клавиш (например, при нажатии пробела) - возвращение в просмотр описания приема.

11. Для просмотра программы приема нажимаем клавишу Home. Сразу после этого появляется текст фрагмента программы, содержащий ключевой оператор

"контрольприема(...)", адресующий данный прием. В нашем случае это оператор "контрольприема(степень набор(1 6 0) второйтерм)". Здесь логический символ "степень" и число 160 - координаты описания приема в восьмом информационном блоке. Заметим, что хотя прием закреплен в этом блоке за символом "степень", программа его закреплена за символом "плюс", и попытка применения приема будет предприниматься при обнаружении вхождения в терм задачи последнего символа.

Напомним, что при сканировании задачи имеются следующие стандартные значения программных переменных: x1 - задача; x2 - вхождение текущего логического символа; x3 - вхождение текущего терма задачи (посылки либо условия) в соответствующий список задачи; x4 - указатель посылки (0) либо условия (1); x5 - текущий уровень сканирования. Переменные, начиная с переменной x6, изначально не определены. Однако, до того момента, как программа обратилась к фрагменту, содержащему оператор "контрольприема(...)", уже могли быть предприняты некоторые проверки и присвоения. Поэтому, для анализа той ситуации, которая сложилась на момент входа в фрагмент, прежде всего нужно просмотреть цепочку предшествовавших ему фрагментов. Это делаем, используя клавиши курсора: "курсор вверх" - к надфрагменту; "курсор влево - курсор вправо" - выделение желтым цветом того оператора перехода в текущем фрагменте, через который требуется войти в подфрагмент; "курсор вниз" - вход в подфрагмент.

В нашем случае, нажав "курсор вверх", оказываемся в фрагменте, имеющем начальные операторы "ветвь 1 уровень(1 2) новый равно(x4 1) ветвь 2 тип(x1 описать) равно(второйоперанд(x2) последнийоперанд(x2)) ветвь 3", причем через "ветвь 3" переходим обратно к начальному фрагменту приема. Эти несколько операторов оказались общими у нашего приема с другими приемами, и они вынесены наружу. Оператор "уровень(1 2)" проверяет равенство текущего уровня сканирования единице либо двойке; оператор "новый" блокирует попытки применения приема при локальных циклах повторного сканирования, предшествующих переводу терма задачи из "теневой" зоны в активную; оператор "равно(x4 1)" проверяет, что символ "плюс" встретился в условии задачи. Оператор "тип(x1 описать)" фиксирует тип текущей задачи; оператор "равно(второйоперанд(x2)последнийоперанд(x2))" - проверяет, что сумма имеет два слагаемых. Одно из них соответствует квадрату неизвестного выражения, другое - его первой степени. Если уравнение изначально имело несколько членов с квадратом либо с первой степенью, то после срабатывания приемов приведения подобных членов с известными коэффициентами слагаемых останется ровно два.

Можно было бы просмотреть и фрагменты, предшествующие данному, однако в них ничего существенного не происходит - идет "главный ствол" программы символа "плюс", ответвления от которого соответствуют различным значениям уровня срабатывания. Поэтому возвращаемся через "ветвь 3" в начальный фрагмент приема. Пропуская фиктивный оператор "контрольприема", переходим к следующим операторам. Оператор "равно(x6 неизвестные(x1))" присваивает переменной x6 цель задачи x1, перечисляющую ее неизвестные. Эта цель имеет вид (неизвестные $x_1 \dots x_n$). Оператор "альтернатива(не(длинаменее(x6 3)) равно(x5 1) равно(x5 2))" проверяет, имеет ли задача одну неизвестную - тогда длина набора x6 будет меньше 3, или больше одной - тогда эта длина бу-

дет не менее 3. Соответственно, в первом случае текущий уровень сканирования должен быть равен 2, а во втором случае - 1.

Оператор "операнд(x7 x2)" переводит от текущего символа "плюс" к внешнему символу, присваивая переменной x7 вхождение этого символа. Оператор "символ(x7 равно)" проверяет, что по вхождению x7 расположен символ "равно". Оператор "корень(x7)" проверяет, что вхождение x7 - корневое, т.е. рассматриваемое условие имеет вид равенства, и т.д.

По окончании анализа программы можно вернуться в просмотр описания приема, нажав клавишу End. Заметим, что нажатие End приведет к этому результату вне зависимости от того, в каком фрагменте программы символа "плюс" она была нажата. Если при просмотре окрестности начального фрагмента программы нужно быстро вернуться к этому начальному фрагменту, то нажимается клавиша F8.

12. Для перехода к оглавлению задачника нажимаем Shift-3. Чтобы вернуться оттуда в оглавление приемов, нажимаем Shift-2. Для перехода к оглавлению программ далее нажимается Shift-1; обратно - снова Shift-2.

13. Прием находится в разделе ("Элементарная геометрия", "Параллельны", "Пропорциональность отрезков, отсекаемых параллельными прямыми", "Соотношение пропорциональности длин отрезков, отсекаемых параллельными прямыми"). Нажимая "3", входим в просмотр окна фильтров. Перемещая с помощью клавиши "курсор вправо" выделенный текущий терм, доходим таким образом до запятой, и снова нажимаем "курсор вправо". В результате появляется второй фрагмент списка фильтров. Проходим через него аналогичным образом, и попадаем в третий, последний фрагмент. Обратные переходы - с помощью "курсор влево". Этот процесс можно ускорить, используя мышь: нажатие ее левой кнопки на запятой приводит к переходу через эту запятую - вперед либо (для запятой в начале фрагмента) назад.

12.2.3 Редактирование приема

На нескольких простых упражнениях продемонстрируем технику ввода нового приема, изменения старого, а также тестирования приема после редактирования. Для ввода новых приемов следует сначала создать специальный концевой пункт оглавления базы приемов, разместив его там, где покажется удобнее (можно и в соответствии с имеющейся классификацией разделов). При первом знакомстве с редактором приемов лучше не размещать учебные приемы в "старых" концевых пунктах, чтобы случайно не удалить имеющиеся там приемы. Впрочем, такие случайности не очень страшны - первоначальная версия удаленного либо измененного приема попадает в буфер, откуда ее можно вернуть обратно.

1. Ввести прием, упрощающий тригонометрические выражения с помощью тождества для суммы квадратов синуса и косинуса. Сначала набрать теорему приема $\forall_{ax}(a(\sin x)^2 + a(\cos x)^2 = a)$. Затем ввести заголовок приема "второйтерм". Далее ввести единственный фильтр - "уровень(0)". Ввести указатели "единица(1 x1)" и "заменазнака(минус x1)". Сохранить введенный прием и откомпилировать его.

2. Проверить, что прием работает, введя какую-либо тестовую задачу на упрощение, например, $2(\sin b)^2 + 2(\cos b)^2$, и найдя в трассировке по шагам решения задачи срабатывание введенного приема. Войти в просмотр описания приема из текущего шага трассировки. Перейти к просмотру текущего фрагмента программы и найти программные переменные, значениями которых служат: а) вхождение косинуса; б) вхождение операции сложения; в) терм, идентифицированный с квадратом синуса.

3. После тестирования вернуться в редактор приемов и проделать следующие действия: а) удалить программу приема, не удаляя самого приема; б) восстановить программу приема; в) добавить фильтр "тип(преобразовать)" и сохранить измененный прием, не изменяя пока его программу; г) еще раз перекомпилировать прием, причем провести компиляцию так, чтобы при тестировании можно было определять те термы, с которыми идентифицируются переменные теоремы приема.

4. После выполнения предыдущего упражнения снова войти в тестирование приема и, остановившись на моменте его применения, найти идентифицированные с теоремными переменными термы, не переходя в отладчик ЛОСа.

5. Создать в оглавлении базы приемов какой-нибудь новый концевой пункт и перенести в него созданный прием из того концевого пункта, где он находился до этого. Старый концевой пункт (если он после этого оказался пустым) удалить.

6. Сбросить буфер базы приемов. Затем удалить введенный прием, после чего вернуть его на старое место из буфера базы приемов. Снова сбросить буфер базы приемов. Изменить прием, убрав из него фильтр "тип(преобразовать)". Посмотреть старую версию приема в буфере. Восстановить по буферу версию приема, имевшуюся до изменения.

7. Создать копию приема, изменив в ней фильтр "уровень(0)"на "уровень(1)".

8. Создать простую версию приема, использующего теорему Пифагора. Сначала изобразить на чертеже прямоугольный треугольник ABC с прямым углом B. Ввести теорему приема:

$$\forall_{ABC}(\text{прямая}(AB)\perp\text{прямая}(BC) \to (l(AC))^2 = (l(AB))^2 + (l(BC))^2).$$

Затем ввести заголовок приема "вывод" и фильтры "уровень(1)", "посылка", "тип(исследовать)", "лексикопредшествует(x26 x28)", "усм(актив(расстояние(x26 x28)))", "усм(актив(расстояние(x26 x27)))", "усм(актив(расстояние(x27 x28)))". Ввести указатели приема: "усм(1)", "теквхожд(1)", "биключ(перпендикулярно x27 интервал(x26 x28))". Ввести нормализаторы общей стандартизации для обработки подтермов выводимого утверждения. Ввести нормализатор "норминтервал" для обработки терма "интервал(x26 x28)", встречающегося в указателе. Сохранить прием и откомпилировать его.

9. Разбить список фильтров приема на две части, оборвав первую часть на фильтре "лексикопредшествует(x26 x28)". Найти в оглавлении базы приемов старый прием на теорему Пифагора, срабатывающий на уровне 10, и перенести из него в только что созданный прием фильтр "контекст(посылка(x1) заголовок(x1 равно) вхождениетерма(x1 расстояние(x26 x28)))".

10. Ввести простую геометрическую задачу на нахождение длины одного из катетов прямоугольного треугольника, у которого длина гипотенузы равна 5, а другого катета - 4. Протестировать на ней созданный прием, установивши прерывание при его применении.

11. Удалить все нормализаторы приема, после чего ввести их в режиме автоматического пополнения описания приема.

12. Ввести прием для сложения дробных выражений. Теорема приема имеет вид:

$$\forall_{abcdef}(\neg(c=0)\ \&\ \neg(d=0)\ \&\ \neg(f=0) \to \frac{ab}{cd}+\frac{ae}{cf}=\frac{a(bf+de)}{cdf}).$$

Заголовок приема - "второйтерм"; фильтры - "уровень(1)", "условие", "тип(преобразовать)", "или(заголовок(фикс(0 1 1)дробь) и(заголовок(фикс(0 1 1)минус) первыйсимвол(фикс(0 1 1)дробь)))", "или(заголовок(фикс(0 1 2)дробь) и(заголовок(фикс(0 1 2) минус) первыйсимвол(фикс(0 1 2) дробь)))". При вводе фильтров использовать автоматическое выписывание в третьем окне приема указателей вхождения "фикс(...)" после выделения этих вхождений в теореме цветовой указкой, заметив, что "фикс(0 1 1)", "фикс(0 1 2)" здесь ссылаются на вхождения первого и второго дробных слагаемых. Смысл этих фильтров в том, чтобы при обращении в единицу некоторых переменных из знаменателей не пропали дроби. Указатели приема - "блокпроверок(1 2 3)", "единица(1 x1 x2 x3 x4 x5 x6)", "заменазнака(минус x2 x5)". Снабдить прием указателями нормализации, причем сумму в числителе результата снабдить, кроме обычного нормализатора "нормплюс", также нормализатором "видумножение", выполняющим попытку разложения на множители. Ввести в прием какое-либо текст-формульное сопровождение, поясняющее срабатывание приема. Создать тестовую задачу для приема, выйти в точку срабатывания при трассировке и убедиться в том, что пояснение выдается.

13. Удалить все приемы, введенные в перечисленных выше упражнениях.

Указания

1. Выберем для нового приема какой-нибудь подходящий раздел. Например, в подразделе "Тригонометрия" раздела "Элементарная алгебра" выберем подраздел "Синус", далее - "Общая стандартизация выражений", и здесь введем новый концевой пункт, нажав клавишу "к". При появлении курсора текстового редактора нужно набрать какое-либо название пункта; пусть это будет "Тест". Далее нажимаем "курсор вправо" и попадаем в чистый экран - обычное начало ввода приема. Для набора теоремы формульным редактором нажимаем Ctr-ф. Возникает курсор формульного редактора, и далее набираем указанное выше тождество, не забыв начать его с квантора общности и перечислить в кванторной приставке все переменные теоремы. Напомним, что неполное перечисление переменных в кванторной приставке может, хотя и редко, привести к неправильной компиляции. Если переменная оказывается связана внутри теоремы, то она в общую кванторную приставку не выносится. Если при наборе формулы нужно вспомнить какую-либо клавишу формульного редактора, нажимается F1, переводящее в справочник по системе, и нужная информация находится в

подразделе "Формульный редактор" корневого меню справочника. Чтобы продолжить набор формулы, далее нажимается End (сначала - для возвращения в оглавление справочника, затем - для возвращения в набор теоремы).

По завершении набора теоремы нажимается Enter, и под теоремой прорисовывается горизонтальная линия. Далее нажимается Enter, и во втором окне набирается заголовок приема. После того, как заголовок введен и нажато Enter, появляется следующая горизонтальная линия, причем сразу же под ней появляется курсор текстового редактора для набора фильтров. Вводим фильтр "уровень(0)"; после завершающего набор нажатия Enter вводим указатели "единица(1 x1)" и "заменазнака(минус x1)", и снова нажимаем Enter. Под четвертым окном появляется голубая линия, указывающая на то, что новый либо измененный прием еще не сохранен. Чтобы сохранить его без немедленной компиляции (что естественно делать, если прием еще не полностью проработан), нажимается F4, и тогда голубая линия меняет цвет на красный. В нашем случае сохраняем прием и одновременно компилируем его, нажимая F3.

2. Переходим из просмотра приема в задачник - либо через главное меню, либо за один переход, нажимая Shift-3. В задачнике выбираем какой-либо подходящий раздел, например, ("Элементарная алгебра", "Упрощение выражений", "Тригонометрические выражения", "Упрощение выражений - 2"). Прокрутив меню номеров пунктов этого оглавления вверх так, чтобы внизу появилось свободное место, нажимаем "к" (кир.). Тогда появляется поле для ввода новой задачи, ограниченное сверху горизонтальной линией. Нажимаем "ц" для входа в оглавление типов целевых установок, выбираем пункт ("Преобразовать выражение", "Упростить выражение в области допустимхы значений") и нажимаем "курсор вправо". Далее вводим упрощаемое выражение $2(\sin b)^2 + 2(\cos b)^2$ - сначала нажимаем Enter, и используем формульный редактор. После набора этого выражения начинаем трассировку процесса решения. Здесь имеются две возможности. Одна - начать нажатием "р" (кир.) и просмотреть все шаги решения задачи, нажимая каждый раз Enter для перехода к следующему шагу. В нашем тривиальном случае это сразу выведет на срабатывание тестируемого приема. Другая возможность - сразу установить прерывание при срабатывании нужного приема. Для этого нужно сначала нажать "г" - появится оглавление базы приемов, затем войти в просмотр нужного приема, и нажать либо Enter, либо Ctr-Enter. В первом случае прерывание произойдет после срабатывания приема, во втором - при выходе на начальный оператор "контрольприема(...)" программы приема, причем здесь включается отладчик ЛОСа и устанавливается пошаговый режим трассировки. При отладке применяется именно второй способ, так как он позволяет понять причины несрабатывания приема. Мы нажимаем Enter.

   Чтобы войти из просмотра текущего преобразования в просмотр описания примененного приема, нажимается клавиша "б". Это делается не только для приемов, реализованных на ГЕНОЛОГе, но и для приемов, запрограммированных непосредственно на ЛОСе. В первом случае появится описание приема, во втором - пункт оглавления программ, соответствующий примененному приему. Чтобы вернуться в просмотр текущего преобразования, нажимаем End. Отсюда можно выйти в отладчик ЛОСа для просмотра текущего фрагмента программы - нажатием клавиши "ф".

Чтобы теперь проанализировать значения программных переменных, нужно помнить смысл первых пяти переменных, определенных при сканировании задачи: x1 - текущая задача; x2 - вхождение в нее текущего символа (название его прорисовано в верхней левой части экрана); x3 - вхождение текущего терма задачи (посылки либо условия) в его внешний список; x4 - указатель посылки (0) либо условия (1); x5 - текущий уровень сканирования. Сразу ясно, что вхождение косинуса является значением переменной x2 - так как прием инициализируется при обнаружении косинуса. Для получения предыстории текущего фрагмента нажимаем Home - переходим в просмотр фрагмента, из которого по "иначе 2" перешли в данный фрагмент. Замечая оператор "операнд(x6 x2)", запоминаем, что значением переменной x6 является вхождение внешней по отношению к косинусу операции. После этого нажимаем End и возвращаемся в первоначально выданный на экран фрагмент программы. Видно, что в начале этого фрагмента расположены операторы "символ(x6 степень)", "равно(x2 первыйоперанд(x6))", фиксирующие размещение косинуса в основании некоторой степени. Можно не читать программу оператор за оператором, а сразу искать оператор "символ(... плюс)", находящий корневую сумму преобразуемого выражения. Такой оператор имеется - это "символ(x9 плюс)", так что вхождение суммы является значением переменной x9. Аналогичным образом, не читая подробным образом всего текста программы, обнаруживаем операторы "заголовок(x19 степень)", "первыйсимвол(x19 синус)", что и дает нам программную переменную x19, значением которой служит терм $(\sin x)^2$.

3. Для обрыва трассировки нажимаем Esc, возвращаясь к просмотру исходного условия задачи. Возвращаемся в базу приемов - либо через главное меню, либо выйдя в оглавление задачника и нажа Shift-2. Переходим к просмотру нашего приема. Чтобы удалить его программу, не удаляя приема, достаточно нажать F7. Нижняя горизонтальная линия перекрасится из черного цвета в красный - сигнал об отключении приема. Для восстановления программы нажимаем F5. Надо заметить, что время на восстановление программы приема, затрачиваемое компилятором ГЕНОЛОГа, совсем невелико, и если сравнивать его со временем решения иных задач средней сложности, то станет ясно, что во многих случаях можно было бы прибегать к помощи компилятора и генератора приемов "в реальном времени", практически не замедляя хода решения задачи. Это означает, что по мере развития техники автоматического синтеза приемов можно будет сохранить в базе приемов лишь малую ее часть - наиболее часто работающие приемы, а остальные создавать по мере надобности на основе базы теорем для одноразового применения.

   Для вставки нового фильтра нажимаем Ctr-3 (это легко запомнить - Ctr и номер изменяемого окна, за исключением случая первого окна). В третьем окне открывается текстовый редактор; с его помощью вводим фильтр "тип(преобразовать)". В данном приеме этот фильтр, конечно, совсем не нужен, и вводится здесь лишь для знакомства с редактором приемов. Чтобы сохранить измененное описание приема, не изменяя его программы, нажимаем F4. Заметим, что при этом нижняя линия приема перекрасится из голубого цвета в черный - так как программа приема все-таки есть, хотя это и старая версия. Для приведения программы приема в соответствие с его описанием нужно было бы нажать F5. Однако, в упражнении требуется применить другой режим компиляции, после которого просмотр термов, идентифицированных с теоремными

переменными, станет возможен без помощи отладчика ЛОСа. Такая компиляция включается нажатием клавиши F6. Внешне она ничем не отличается от обычной, но в специальном информационном блоке запоминается соответствие между теоремными и программными переменными, устанавливаемое компилятором. Если впоследствии перекомпилировать прием с помощью F5, то эта информация сбрасывается.

4. Начнем с того момента, как при трассировке появляется кадр преобразования, выполняемого нашим приемом. Заметим, что в отсутствии специально созданного текст-формульного сопровождения, при срабатывании приема будет выдаваться текст того концевого пункта оглавления базы приемов, где размещен прием. Этот текст заключается в скобки и располагается после описания преобразования.

   Для входа в просмотр описания примененного приема нажимаем "б", и после прорисовки приема нажимаем клавишу "п" ("переменные"). Тогда все окна приема, кроме первого, удаляются с экрана, и под теоремой прорисовывается равенство $a = 2$ - т.е. теоремная переменная a идентифицирована с двойкой. Чтобы перейти к просмотру следующей переменной, нажимаем PageDown - появляется равенство $x = b$, т.е. теоремная переменная x идентифицирована с термом b. Для возвращения к ранее просмотренным переменным используется клавиша PageUp. Для обрыва просмотра нажимаем любую другую клавишу, например, пробел. Описанная возможность уточнения идентификации переменных при срабатывании приема - отладочная и обычно дополняется информацией, предоставляемой отладчиком ЛОСа. При наличии подробного поясняющего срабывание приема текста она становится излишней. По окончании упражнения выходим из трассировки и возвращаемся в просмотр приема.

5. Чтобы перенести прием в другое место оглавления базы приемов, сначала нажимаем Insert. Это можно делать, даже если тот концевой пункт, куда предполагается перенести прием, еще не создан. Например, выйдя в оглавление, переходим в раздел ("Косинус", "Общая стандартизация выражений"), и вводим новый концевой пункт "Тест" в этом разделе. Введя его и выделив голубым цветом, но не входя внутрь, снова нажимаем Insert. Теперь можно зайти внутрь пункта - наш прием окажется там. Из старого пункта он при этом удаляется. Возвращаемся к старому пункту "Тест", проверяем, что он пуст, выделяем его в оглавлении и нажимаем для удаления Ctr-Del.

6. Для сброса буфера базы приемов нужно выйти в оглавление этой базы и нажать "О" (кир.). Если текущие пункты оглавления относились к сбрасываемому буферу, то после сброса буфера произойдет выход в главное меню, иначе - сохранится текущий пункт оглавления. Обычно буфер сбрасывается тогда, когда все новые или измененные приемы уже протестированы - например, предпринята прокрутка по тем разделам задачника, где они могут проявить активность. После сброса буфера возвращаемся в просмотр приема, и для удаления его нажимаем Ctr-Del. Прием будет удален, программа его - тоже. Однако, исходная версия удаленного приема будет сохранена в буфере базы приемов. Чтобы перейти к ней, нужно выйти в оглавление базы приемов и нажать "ф". Буфер представляет собой автоматически вводимую ветвь корневого меню оглавления базы приемов. Нажатие "ф" переводит в некоторый раздел этой ветви. В корневой части ветви находим раздел "Удаленные приемы", в нем - подраздел

"Тест". Входим в подраздел и обнаруживаем в нем только что удаленный прием, причем нижняя его линия - красная, так как программа приема отсутствует. Далее обычными средствами переносим прием на старое место - сначала нажимаем Insert в описании данного приема, затем выходим в оглавление, находим в разделе "Косинус" пустой теперь концевой пункт "Тест", и снова нажимаем Insert, выделив этот пункт.

Чтобы еще раз сбросить буфер, перед входом в просмотр описания приема нажимаем "О".

Войдя в просмотр описания приема, нажимаем Ctr-З и удаляем текстовым редактором фильтр "тип(преобразовать)". Затем сохраняем прием и компилируем его, нажав клавишу F3. Чтобы просмотреть старую версию, сохраненную в буфере, далее нажимаем "б". Появляется старая версия приема, у которй нижняя линия - красная (программы это версии нет). Чтобы вернуться от нее к текущей (основной) версии, нажимаем "о" (кир.). Наконец, для восстановления старой версии приема нажимаем "Б". Новая версия уничтожается; старая извлекается из буфера, компилируется и переносится на ее место. Нужно предупредить, что такое восстановление старой версии приема не гарантирует восстановления исходного размещения ее программы. Обычно компилятор присоединяет ветвь новой программы, при прочих равных условиях, после ранее введенных ветвей. Если точка входа в программу измененного приема располагалась в середине некоторого линейного списка точек входа в другие приемы, то после перекомпиляции она попадет в конец этого списка. Чаще всего, это несущественно, однако иногда имеются "состязания" между приемами одного списка, и тогда перекомпиляция может существенно повлиять на поведение решателя в отдельных задачах (не всегда, впрочем, в худшую сторону). В особых случаях после перекомпиляции можно переставить ветви программ вручную, используя редактор ЛОСа, хотя более радикальный способ здесь - обычное уточнение решающих правил приемов.

7. Чтобы создать копию приема, нажимаем из просмотра его клавишу Ctr-д. Появляется тот же самый текст описания, но нижняя его линия - синия. На самом деле, это уже копия приема. Сначала сохраняем ее без компиляции (F4). Затем нажимаем Ctr-З, входим в редактирование окна фильтров и изменяем первый фильтр на "уровень(1)". Далее сохраняем прием и компилируем (F3). Заметим, что можно было откомпилировать копию приема и не изменяя его - тогда программы приемов склеятся вплоть до последних операторов, выполняющих преобразование, а перед ними произойдет ответвление одной программы от другой.

8. Прием поместим в том же самом разделе "Тест", что и предыдущий. Для инициализации его нажимаем "Ctr-п" (когда раздел был пустой, нажатия специальной клавиши для инициализации приема не требовалось - можно было сразу вводить теорему). Появляется чистый экран, и начинается ввод компонент приема. Заметим, что если оборвать набор приема до его сохранения и выйти, например, в оглавление либо даже в просмотр соседних приемов того же самого концевого пункта (это делают клавиши "курсор вверх", "курсор вниз"), то вся работа по набору начальной части приема пропадет - как если бы клавиша Ctr-п не нажималась вовсе.

В нашем случае новым является наличие чертежа. Он вводится до набора теоремы. Нажимаем Ctr-ч, после чего появляется синяя рамка геометрического редактора. Выбираем курсором мыши в верхней левой части экрана пункт меню "Точка" и нажимаем левую кнопку мыши. В правом верхнем окне рамки редактора возникает указатель режима "Точка". Теперь можно вводить точки чертежа. Выбираем подходящее место для вершины A треугольника и нажимаем левую кнопку мыши. Кроме точки, появляется ее обозначение, автоматически вводимое редактором - берутся последовательные большие буквы латинского алфавита. Далее выбираем место для вершины прямого угла B и место для вершины C, нажимая каждый раз левую кнопку мыши. Введя точки, включаем режим "Отрезок" (см. меню в верхней части экрана). Далее проводим стороны треугольника: подводим курсор к точке A и нажимаем левую кнопку мыши; затем подводим курсор к точке B и снова нажимаем левую кнопку мыши. Появляется отрезок AB. Таким же образом создаем отрезки AC и BC. Если точки легли не очень удачно, то выбираем ту из точек, которую желательно сдвинуть - опять нажатием левой кнопки мыши. Для сдвига используем клавиши курсора - каждое нажатие сдвигает выделенную точку на один пиксел в соответствующем направлении. Чтобы одновременно сдвигать и букву, обозначающую точку, аналогичным образом используем клавиши "Ctr-курсоры". Наконец, чертеж сдвигаем в верхнюю часть экрана, чтобы как можно больше места оставить для прочих компонент описания приема. Для этого включаем режим "перемещение". Перемещение производится указанием начала и конца вектора перемещения - нажатиями левой кнопки мыши. Этот вектор можно откладывать от любой точки на экране, не связывая ее с чертежом. После указания конца вектора смещается все изображение, причем если места для сдвига не хватает, то он блокируется. Для инициализации нового сдвига - повторное указание его вектора. Ввод чертежа завершается нажатием Enter, и сразу возникает курсор текстового редактора для ввода теоремы.

Все остальные шаги по вводу приема аналогичны первому разобранному примеру - только при вводе нормализатора "норминтервал" придется выделять цветовой указкой не вхождение в теорему, а вхождение в указатель "биключ(...)" из четвертого окна приема. Когда это вхождение выделено, нажимается Enter и вводится нормализатор.

9. Чтобы разрезать список фильтров на две части, выделяем нажатием левой кнопки мыши фильтр "лексикопредшествует(x26 x28)" и нажимаем "р" (кир.). После данного фильтра появляется запятая, а оставшиеся фильтры переносятся "на следующую страницу". Для перенесения фильтра из другого приема сначала находим последний - это второй от конца прием раздела "Теорема Пифагора", расположенного в конце цепи ("Элементарная геометрия", "Фигуры", "Треугольник", "Прямоугольный треугольник"). Выделяем нужный фильтр "контекст(...)" левой кнопкой мыши и нажимаем "у" (кир.). Затем клавишей пробела сбрасываем выделение. Возвращаемся в просмотр "своего" приема и нажимаем клавишу "у" повторно, но не выделяя каких-либо окон или фильтров. Тогда фильтр "контекст(...)" автоматически добавится к списку фильтров рассматриваемого приема. Так как наш список фильтров разрезан на две части, а новый фильтр заносится в конец списка фильтров, то на экране он не появится, пока не будет вызвана для просмотра последняя "страница" списка фильтров. Чтобы сохранить добавленный фильтр и перекомпилировать прием,

лучше до этого нажать F3.

10. Вводим посылки задачи - "прямая(AD) $\perp$ прямая(DC)"; $l(AC) = 5$; $l(AD) = 4$. Входим в меню выбора целевой установки и выбираем пункт ("Найти значения неизвестных", "Выразить значения неизвестных через заданные параметры"). После подсказки "Выразить" набираем x и нажимаем Enter. Так как известных параметров в задаче нет, подсказку "через" игнорируем и сразу нажимаем Enter. Под целевой установкой набираем условие: $x = l(CD)$. Чертеж вручную строить не нужно - для его автоматического создания нажимаем Ctr-ч. На точку срабатывания приема попадаем так, как это уже объяснялось выше.

11. Для удаления нормализаторов входим в просмотр описания приема и нажимаем Ctr-End. Чтобы восстановить эти нормализаторы, нажимаем "н" (кир.). Тогда включается режим ручного анализа последовательности предложений, вносимых генератором приемов. Здесь могут появляться предложения на добавление либо удаление нормализаторов и указателей. Если предлагается добавить элемент, то первый пункт меню в левой верхней части экрана имеет вид "Новый указатель", иначе - вид "Старый указатель". Чтобы ввести новый элемент либо сохранить старый, нужно нажать Insert; чтобы отказаться от нового элемента либо удалить старый - нажать Delete. В нашем случае последовательно появляются нормализаторы, так что естественно все время нажимать Insert - пока предложения не будут исчерпаны. После этого нажимаем F3.

12. Ввод теоремы приема сложения дробных выражений и его заголовка происходит обычным образом. При наборе фильтра "или(заголовок(...)...)" после открывающей скобки символа "заголовок" нажимаем клавишу Shift-1, приводящую к появлению цветовой указки в первом окне. Выбираем этой указкой первое слагаемое и нажимаем клавишу "ф". Сразу вслед за этим в третьем окне начиная с текущей позиции курсора будет прорисован указатель вхождения "фикс(0 1 1)", а цветовая указка в первом окне пропадет. Остальные указатели вхождения вводятся таким же образом. Нормализаторы приема можно создать частично с помощью клавиши "н", а частично - добавить вручную.

    Для ввода текст-формульного шаблона нажимаем клавишу "6" и вводим, например, следующий текст: "Выражение () равно сумме дробей () и ()". Чтобы первая буква слова "Выражение" не пропала при выдаче на экран, отступаем при наборе этого слова от левого края экрана на один символ. По завершении набора нажимаем Enter. Если понадобится изменить текст шаблона, то снова нужно будет нажать "6", затем - Enter для входа в текстовый редактор.

    Чтобы определить выражения, подставляемые в шаблон вместо скобок, вводим указатель "титр(набор(1 терм(фикс(0 2))терм(фикс(0 1 1))терм(фикс(0 1 2))))", в котором указатели вхождений выделяют, соответственно, правую часть тождества и вхождения дробных слагаемых. Здесь снова применяется описанный выше интерфейс полуавтоматического набора указателей вхождения. Для тестирования приема вводим, например, задачу на упрощение выражения $a/3b + c/b^2d$. При срабатывании приема, кроме поясняющего текста, появятся также сообщения о вспомогательных проверках отличия от нуля знаменателей. При желании, эти проверки можно будет тут же повторить. Однако, ввиду тривиальности проверок, сообщения о них лучше вообще не выдавать. Чтобы убрать сообщения, следует вернуться в прием и добавить указатели "комментарий(1

титр(стоп))", "комментарий(2 титр(стоп))", "комментарий(3 титр(стоп))", передающие комментарии (титр стоп) проверочным операторам первых трех антецедентов.

13. Для удаления всех введенных в упраженениях приемов последовательно нажимаем Ctr-Del из просмотра их описаний. Затем убираем пустой концевой пункт "Тест", выходим в оглавление базы приемов и нажимаем "О".

12.2.4 Анализ приема

Разберем ряд простых упражнений на анализ описаний уже созданных приемов. Так как многие компоненты приема имеют эвристический характер, то чтобы понять их назначение, нужно прежде всего найти задачи, ради которых они были введены. Если возникло предположение о целесообразности изменения данных компонент, то необходимо провести тестирование измененного приема на тех же задачах, а при ослаблении ограничений на срабатывание - также и на дополнительных.

1. Найти раздел "Углы при параллельных прямых и секущей", и в этом разделе войти в просмотр описания третьего с конца приема. Объяснить фильтры этого приема. Найти в разделе задачника ("Элементарная геометрия", "Задачи на вычисление") все задачи, в которых данный прием срабатывает. Определить, для каких из них его срабатывание избыточно. Предложить дополнительный фильтр, блокирующий хотя бы часть ненужных срабатываний.

2. Найти раздел "Произведение в основании степени", и в этом разделе войти в просмотр описания последнего приема. Объяснить фильтры этого приема. Найти в разделе задачника "Исследование сходимости числовых рядов - 3" задачу, для которой удаление фильтра "не(контекст(...))" приводит к зацикливанию процесса решения.

3. Найти раздел "Потенцирование логарифмических уравнений", и в этом разделе войти в просмотр второго с конца приема. Объяснить фильтры этого приема. Найти в задачнике задачу, в которой данный прием срабатывает, и выйти на точку его срабатывания.

4. Найти раздел "Возведение в квадрат уравнения с одним неизвестным радикалом", и в нем войти в просмотр последнего приема. Объяснить фильтры этого приема. Придумать систему уравнений, в которой фильтр "или(равно(число-неизвестных(корень)1)...)" ускоряет получение ответа. Найти такую систему в задачнике.

5. Найти в ветви раздела "Плюс" концевой подраздел "Вычитание уравнения, домноженного на известный коэффициент, с целью получения линейного уравнения, либо уравнения с одной неизвестной, либо разложения на множители левой части уравнения", и войти в просмотр единственного приема этого подраздела. Объяснить фильтры приема. Придумать задачу, в которой прием сработал бы, занести ее в задачник, и выйти на точку срабатывания приема.

6. Чтобы найти прием, суммирующий квадраты синусов элементов арифметической прогрессии, занести в задачник пример на такое суммирование, выйти при трассировке на срабатывание искомого приема, и определить его место в оглавлении базы приемов.

7. Найти в ветви раздела "Описанная около фигуры окружность" концевой подраздел "Теорема синусов". Рассмотреть последние четыре приема этого подраздела, и для каждого из них найти все задачи на вычисление по планиметрии, в которых он срабатывает. Определить, в каких задачах срабатывание приема являлось избыточным. Предложить фильтры, отсекающие хотя бы часть таких срабатываний.

8. Найти в ветви раздела "Касательная" концевой подраздел "Длина касательной и длины отрезков секущей". Рассмотреть последние три приема этого подраздела, и для каждого из них найти все планиметрические вычислительные задачи, в которых он срабатывает. Объяснить различия между контекстами срабатывания этих приемов. Определить, в каких задачах срабатывание приема было избыточным.

9. Найти раздел ("Тангенс", "Уравнения", "Переход к тангенсу половинного аргумента") и войти в просмотр его последнего приема. Объяснить фильтры этого приема. Придумать задачу, в которой прием сработал бы; занести ее в задачник, и выйти на точку срабатывания.

10. В ветви раздела "Плюс" найти концевой подраздел "Выражение неизвестной X из уравнения $X + A = B$" и войти в просмотр единственного приема этого подраздела. Определить последствия удаления фильтра "или(и(тип(описать) уровень(5))...)".

Указания

1. Прием находится в разделе ("Элементарная геометрия", "Параллельны", "Углы при параллельных прямых и секущей"). Уровень срабатывания его равен 9, причем срабатывание происходит при рассмотрении некоторой посылки задачи на доказательство либо на исследование. Заметим, что почти все приемы по элементарной геометрии предназначены для срабатывания в списке посылок задач этих типов. Задачи на вычисление либо на построение, изначально оформляемые как задачи на описание, решаются (в особенности первые) с помощью вывода следствий в своем блоке анализа.

   Фильтр "неизв(...)" означает, что величина угла DFE между касательной и секущей должна быть не известной и связанной с "основными" неизвестными, т.е. неизвестными внешней задачи на описание. Такая связь понимается либо как вхождение неизвестной в выражение t для величины угла, либо как наличие цепочки уравнений, в первое из которых входит какой-либо угол или расстояние из t, в последнее - неизвестная, а любые два соседних уравнения имеют какое - либо общее расстояние либо угол. Заметим, что выражение t для величины угла DFE находится в приеме с помощью нормализатора "нормугол".

   Следующий фильтр отсекает вырожденный случай совпадения рассматриваемых параллельных прямых (хотя и не гарантирует этого несовпадения, так как совпадающие прямые могут иметь различные обозначения).

   Последний фильтр отсекает случаи, когда отрезок секущей является стороной параллелограмма, трапеции либо ромба - для углов этих фигур имеются свои приемы. Квадрат и прямоугольник не рассматриваются потому, что для них угол DFE известен.

В общем-то, этот фильтр не доведен до конца - нужно было бы еще уточнить, что смежные стороны не лежат на наших параллельных прямых. Такого рода недоработок в текущей версии решателя имеется немало; обычно они легко устраняются, хотя иногда для этого может понадобиться дальнейшее пополнение ГЕНОЛОГа. Наличие их объясняется, во-первых, объемом системы, а во-вторых, теми целями, которые ставились при ее развитии - проверкой и уточнением общих принципов моделирования логических процессов, выявлением общих архитектурных и технологических подходов. На этом этапе работы решатель должен рассматриваться не как готовый продукт, а лишь как первая черновая версия проработки предметных областей, которая могла бы стать организующим началом для последующего систематического их освоения. Впрочем, принцип преодоления при обучении решателя "всех задач из задачника подряд" делает свое дело - в тех случаях, где недоработки действительно ощутимы на реальном потоке задач, они достаточно быстро обнаруживаются и, так или иначе, подправляются. Это приводит к тому, что даже данная версия уже может рассматриваться как некоторая заявка на создание системы компьютерной математики нового типа, в особенности если сравнивать ее возможности по решению плохо алгоритмизируемых задач с возможностями других систем компьютерной математики.

Следующий пункт упражнения - поиск задач, на которых прием срабатывает. Чтобы можно было выполнить быстрый поиск в заданном разделе задачника всех таких задач, должна была иметь место хотя бы одна прокрутка решателя на этом разделе после того, как прием был введен. Если после прокрутки прием был изменен, то поиск даст задачи, в которых прием срабатывал до изменения. Для инициализации поиска следует перейти от просмотра приема к задачнику, нажав клавишу Shift-3. Затем следует выделить тот раздел, для которого будет выполняться поиск, нажать Shift-2 и таким образом снова оказаться в оглавлении базы теорем. Здесь нужно войти в просмотр приема и нажать "A" (кир.). Если нужно получить данные для серии приемов, то после нажатия "A" можно перейти к другим приемам, нажимая при просмотре каждого из них на "A". После нажатия на "A" происходит просмотр протоколов задачника, который может создавать небольшую паузу (обычно только для первого нажатия "A"). По завершении указанных действий нажимается клавиша Shift-3, снова переводящая в оглавление задачника. Данные просмотра сохраняются в буфере задачника, и для перехода к нему нажимается "ф".

В нашем примере, после выполнения перечисленных действий, находим в буфере раздел "Углы при параллельных прямых и секущей" - в нем и будут собраны все найденные задачи. Нажимаем "курсор вправо" и обнаруживаем номера 8 отобранных задач. В действительности это лишь повторные ссылки на задачи, уже имеющиеся в основных разделах задачника. Поэтому изменять задачи в буфере не следует. Однако, можно запускать процессы решения их так же, как для основной части задачника - поштучно либо сразу серией из 8 задач.

Нам нужно определить, в каких из 8 задач рассматриваемый прием в действительности не нужен. Чтобы сделать это, можно было бы зайти в оглавление базы приемов, отключить прием нажатием клавиши F7, запустить процессы решения отобранных задач, и посмотреть, каковы будут результаты. Для начала рекомендуем реализовать этот способ. Здесь надо иметь в виду, что при решении задачи с отключенным приемом решение ее может замедлиться на-

столько, что для перехода к следующей задаче серии нужно будет нажать последовательно Break и Esc (впрочем, в нашем примере этого делать не нужно). Напоминаем, что серийный запуск выполняется, после возвращения в буфер, нажатием Ctr-з.

По окончании серийного решения просматриваем его итоги, нажав клавишу "з". Видно, что одна задача вообще не решена, а решение трех - очень сильно замедлилось. Для выхода на задачу с утерянным решением нажимаем "у", для просмотра серии задач с замедлением решения (упорядоченных по убыванию замедления) - нажимаем "з", и для перехода к каждой очередной задаче - "ш". Любую из этих задач можно выбрать для пошагового просмотра решения и анализа возможностей его доработки. Впрочем, нам известна причина перечисленных ухудшений; интерес представляют в первую очередь другие задачи - те, где ответ не изменился, а решение ускорилось. Если анализ точек срабатывания приема в этих задачах покажет, что оно бессмысленно или крайне слабо мотивировано, то появится возможность ввести дополнительный фильтр, отсекающий данные точки. Разумеется, после этого нужно будет еще проверить, не оказались ли отсечены срабатывания приема в тех задачах, которые без него не решаются либо замедляются. Здесь могут появиться еще какие-то коррекции фильтра, и т.д. В этом итеративном процессе нужно помнить, что замедление либо потеря решения после отключения приема, вообще говоря, еще не означают, что он действительно необходим. К сожалению, достаточно часто задача решается из-за случайного "рикошета", возникшего после срабатывания приема, и тогда приходится продолжать работу с ней до выявления тех средств, которые в действительности ей нужны.

Мы не будем использовать данные серийной прокрутки, полученные указанным выше способом. Вместо этого восстановим программу приема нажатием клавиши F5 (из просмотра приема) и повторим прокрутку отобранных 8 задач при подключенном приеме - для восстановления исходной ситуации. Дело в том, что анализ избыточности приема на отобранных задачах можно выполнить автоматически. Для этого достаточно нажать клавишу "й", инициирующую серийный запуск с блокировкой срабатываний нашего приема без удаления его программы. Если время решения задачи с заблокированным приемом оказывается значительно больше исходного, то процесс ее решения обрывается автоматически, и начинается рассмотрение следующей задачи. За ходом процесса можно следить по тому, как выделенный синим цветом номер текущей задачи перемещается вниз. В тех случаях, когда ответ на задачу сохранился, а время решения сократилось, после номера задачи ставится знак вопроса - здесь применение приема не нужно.

После прокрутки возникают знаки вопроса на третьей, четвертой и шестой задачах. Чтобы теперь анализировать контексты срабатывания приема, лучше всего дополнить его указателем "стоп" и перекомпилировать. Тогда при каждом его применении, непосредственно до выполнения преобразования, будет происходить выход в отладчик ЛОСа. Чтобы просмотреть преобразование, достаточно будет в этой ситуации нажать "пробел" для перехода в трассировку по шагам решения задачи и Enter - для начала такой трассировки. На первом же шаге будет выведена информация о текущем применении приема. В дальнейшем, по завершении работы с приемом, указатель "стоп" следует удалить - вручную либо, если не было внесено других изменений в прием, нажатием "Б"

для восстановления исходной версии.

Чтобы понять, какие фильтры можно было бы ввести для отсечения найденных трех ненужных срабатываний, сначала стоит посмотреть на те задачи, в которых преобразование оказалось нужным. Возможно, при этом выяснятся какие-то дополнительные связи величин, входящих в выводимое приемом соотношение, с прочими объектами задачи, при наличии которых увеличивается вероятность извлечь пользу из соотношения. Просматривая контексты срабатывания приема в пяти задачах, не помеченных знаком вопроса, обнаружим, например, что в каждой из них вершина E нового угла BEF, вводимого приемом, уже являлась вершиной некоторого рассматриваемого в задаче угла. Просматривая после этого контексты срабатывания в остальных трех задачах, обнаруживаем, что для них это условие выполнялось не всегда. Это уже дает основание для ввода нового фильтра "контекст(усм(актив(угол(х2 х30 х3))))", означающего наличие выделенного в задаче угла с вершиной E; здесь вместо переменной E формульного редактора используем ее обозначение х30 текстовым редактором. Вводим данный фильтр в третье окно описания приема, удаляем указатель "стоп" и перекомпилируем прием нажатием F3. После тестовой прокрутки восьми задач из буфера обнаруживаем, что решение четвертой задачи ощутимо ускорилось - вместо 53 млн.шагов интерпретатора оно стало требовать лишь 40 млн. шагов.

Разумеется, проведенный здесь анализ весьма поверхностный, и в данном примере можно было бы попытаться продолжить оптимизацию решения выделенных задач, добиваясь более веской мотивировки применения приема. В тех случаях, когда эта мотивировка неясна, а без приема задача решается хуже, нужно искать другие хорошо мотивированные действия, идущие в обход применения данного приема, и создавать для них новые приемы. Материала для подобной оптимизации в решателе более чем достаточно. Продолжение ее нужно не только ради получения эффективной системы для пользователя, но главным образом для создания средств автоматического синтеза приемов. Многие компоненты описаний приемов вынужденным образом имеют эвристический характер, однако даже в них прослеживается определенная логика. Расшифровка такой логики неразрывно связана с оптимизацией решателя, отсечением "предрассудков" и выявлением тех взаимодействий между приемами, которые реально нужно учитывать при принятии решений. Можно надеяться, что продвижение в этом направлении позволит постепенно перейти к автоматическому развитию решателей.

2. Путь к разделу в оглавлении базы приемов - ("Элементарная алгебра", "Степени", "Общая стандартизация выражений", "Простейшие свойства степени", "Произведение в основании степени"). Прием предназначен для преобразования степени произведения двух неотрицательных сомножителей в произведение их степеней. Он срабатывает сразу же, как только предоставляется такая возможность - на уровне 0. Переход к степеням с более простыми основаниями рассматривается в решателе как направление общей стандартизации выражений, принимаемое на промежуточных этапах решения задачи. Как правило, перед выдачей ответа выполняется обратный переход - группировка под общий показатель степени. Однако, в ряде случаев разгруппировка степени произведения в произведение степеней сомножителей блокируется, и для ознакомления с тем, когда это признано целесообразным, рассматриваем фильтры приема.

Прежде всего, идет фильтр, запрещающий срабатывание приема в условии задачи на преобразование, имеющей комментарий "длина". Такой комментарий означает, что в задаче уже предпринималась итоговая компактная переформулировка результата; в частности, могла быть предпринята группировка сомножителей под общую степень. После этого срабатывание данного приема, конечно, нежелательно. Аналогичным образом объясняется следующий фильтр: срабатывание приема блокируется при завершающем редактировании ответа задачи на описание. Исключение могут составлять лишь случаи, в которых цель "редакция" появляется не на завершающем этапе поиска неизвестных, а для специального преобразования списка утверждений. В фильтре и указан один из таких случаев - наличие цели "редуцирование", используемой при логическом выводе в базе теорем.

Далее идет ускоряющий фильтр: если число сомножителей равно двум, то в силу симметрии можно потребовать, чтобы тот из них, который идентифицируется с a, лексикографически предшествовал идентифицированному с b. Таким образом вместо двух случаев будет рассматриваться только один. Заметим, что если число сомножителей более двух, то один из множителей (какой именно - решает компилятор) будет идентифицроваться как одиночный операнд произведения, а другой - как произведение всех оставшихся операндов.

Далее - фильтр, запрещающий разгруппировку основания степени, если показатель c ее связан внешним описателем "класс" либо "отображение", а основание не имеет связанных этим описателем переменных. В таких случаях выгоднее упрощать выражение в первую очередь относительно переменных связывающей приставки описателя, так как это может позволить устранить описатель. Например, применение разгруппировки степени при рассмотрении суммы геометрической прогрессии нецелесообразно - выражение потеряет "явный" вид геометрической прогрессии, и прием ее суммирования может не сработать.

Следующий фильтр аналогичен только что рассмотренному - он блокирует разгруппировку степени при рассмотрении показательных уравнений, если основание не содержит неизвестных, а показатель содержит. Общая эвристическая мотивировка похожая - в уравнениях выгодно, при отсутствии особых соображений, упрощать выражения по числу вхождений неизвестных. Для рассматриваемого приема такие конкурирующие соображения в задачах не возникали, что и нашло свое отражение в данном фильтре.

Наконец, последний фильтр нужен для указания на разделение труда между различными приемами. Данный прием срабатывает, если усматривается неотрицательность обоих сомножителей основания степени. Однако, если бы показатель степени имел вид числовой дроби с нечетным знаменателем, то проверки неотрицательности были бы излишни, и для этого случая предусмотрен другой прием (в данном разделе он идет вторым от конца). Так как уровни срабатывания обоих приемов одинаковы (равны 0), то приходится вставлять фильтр, блокирующий применение данного приема в ситуациях, подпадающих под область действия другого. Разумеется, этот фильтр имеет лишь ускоряющий характер - при его отсутствии задачи все равно решались бы, но с несколько большей трудоемкостью.

Чтобы протестировать фильтр "не(контекст(...))", нужно найти такие задачи, в которых степенное выражение входит в определение функции, причем пока-

затель степени содержит варьируемую переменную, а основание - не содержит. Удалим этот фильтр из описания приема и перекомпилируем прием; затем (как сказано в задании) войдем в раздел задачника ("Математический анализ", "Ряды", "Исследование сходимости числовых рядов - 3"). Для серийного запуска решателя в этом разделе нажимаем Ctr-з. Через некоторое время одна из задач "залипает" на экране. Нажимаем Break и выходим в просмотр отладчиком ЛОСа какого-то момента выполнения программы. Для выхода на уровень трассировки "по шагам решения" нажимаем клавиши "пробел" и Enter. Используя для смены кадров нажатия Enter, обнаруживаем зацикливание: сначала срабатывает анализируемый прием, заменяющий $(2(\sin x)^2)^i$ на $2^i(\sin x)^{2i}$), затем - прием, устраняющий возникающие после такой замены вложенные умножения, и далее - прием, группирующий под общий показатель степени i два только что возникших множителя. Последний прием при суммировании действительно бывает нужен - например, для выявления геометрической прогрессии либо каких-либо других стандартных сумм с "показательными" подвыражениями, а значит, рассматриваемый фильтр в этих ситуациях тоже придется вводить - без него решатель будет зацикливаться.

Для обрыва серийного решения возвращаемся в отладчик ЛОСа (нажатие "ф") и нажимаем Ctr-з, Esc. Затем возвращаемся в просмотр нашего приема и нажимаем "Б" для восстановления его исходного вида.

3. Прием находим в разделе ("Элементарная алгебра", "Логарифмы", "Решение уравнений", "Потенцирование логарифмических уравнений"). Прием имеет в качестве возможных уровней срабатывания значения от 1 до 5. Он применяется для преобразования условия задачи на описание либо посылки задачи на исследование. Вообще, посылки задачи на исследование, содержащие неизвестные, часто преобразуются так же, как условия задачи на описание, хотя полного совпадения здесь нет. Теорема приема выделяет логарифмический множитель слагаемого левой части уравнения, причем этот множитель должен содержать неизвестную - либо в основании логарифма, либо в выражении под логарифмом. Если решается система уравнений (число неизвестных более одной), то уровень срабатывания 2 отбрасывается, иначе - отбрасывается уровень 3. С учетом ограничений на уровень срабатывания, определяемых дальнейшими фильтрами, это означает, что для систем уравнений прием не спешит с потенцированием - здесь имеются возможности решить задачу и другими средствами, в то время как для одного уравнения его необходимость более вероятна.

   Далее идет фильтр, учитывающий наличие комментария "логарифм". Такой комментарий к задаче вводится приемом, выполняющим логарифмирование показательного уравнения. Чтобы понять это, можно было бы, например, воспользоваться оператором "текприем", занеся в него условие проверки наличия указателя "замечание(логарифм)" и просмотрев все приемы, имеющие данный указатель. Ясно, что если уже был предпринят шаг перехода от показательных уравнений к логарифмическим, то обратный переход нужно как-то заблокировать. Но даже и здесь, когда потенцирование очевидным образом приводит к упрощению ситуации, его все же следует разрешать. Фильтр допускает потенцирование уравнения при наличии комментария "логарифм" лишь в тех случаях, когда левая часть уравнения представляет собой линейную комбинацию логарифмов, у которой все основания и коэффициенты известны - тогда после потенцирования неизвестные не будут входить в показатели степени.

Далее располагается фильтр, разрешающий срабатывание данного приема в двух случаях:

а) Под логарифмом находится степенное выражение с неизвестным показателем. Тогда исключение логарифма может привести к обычному показательному уравнению. Уровень срабатывания 1 выбран здесь потому, что маловероятно решение такого "двухэтажного" по логарифмическим и показательным конструкциям уравнения другими средствами.

б) Каждое неизвестное слагаемое остатка d левой части уравнения, имеющее общую неизвестную с выделенным логарифмом, является произведением логарифма по основанию a на известный коэффициент. При этом уровень срабатывания 1 разрешается лишь для системы уравнений, в которой другие уравнения не имеют ни логарифма выражения b, ни степени с основанием b. Если бы такое уравнение нашлось, то имелась бы некоторая вероятность решить систему без потенцирования - рассмотрением линейных комбинаций уравнений и логарифмированием других уравнений.

Далее идет фильтр, запрещающий наличие неизвестных логарифмов внутри выражения c - здесь либо возможно упрощение логарифмических выражений до потенцирования, либо потенцирование приводит к столь же сложной для решения уравнения ситуации, что и исходная.

Два последних фильтра связаны с задачами, возникающими в математическом анализе и дифференциальных уравнениях; их пока рассматривать не будем. Чтобы распознавать такие фильтры, заметим только, что обычно в них встречаются символы "связка", "нормИнтеграл", "диффильтр".

Чтобы найти задачу, в которой данный прием срабатывает, выходим в оглавление задачника, выделяем раздел "Элементарная алгебра", нажимаем Shift-2 для перехода в оглавление приемов, входим в просмотр приема, нажимаем из этого просмотра "А" (кир.), и далее нажимаем Shift-3 (после небольшой паузы, в течение которой происходит просмотр архива задачника). Тогда снова возникает оглавление задачника, в котором нажимаем "ф" для перехода к буферу. В буфере обнаруживаем список всех задач раздела "Элементарная алгебра", в которых срабатывает рассматриваемый прием. Этот список находится в разделе, заголовок которого воспроизводит заголовок раздела приема в базе приемов. Выбираем, например, первую задачу списка, нажимаем "г" для запуска ее решения через оглавление приемов, входим в просмотр приема и нажимаем Enter. Первое срабатывание приема отображается на экране, и далее устанавливается режим трассировки по шагам решения задачи. По завершении упражнения расчищаем буфер задачника - в произвольном месте его оглавления нажимаем "О" (кир.).

4. Искомый прием находится в разделе ("Элементарная алгебра", "Степени", "Решение уравнений", "Уравнение с радикалами", "Возведение в квадрат уравнения с одним неизвестным радикалом").

   Он используется для возведения в квадрат уравнения, имеющего в левой части ровно одно слагаемое, среди множителей которого есть неизвестные радикалы четной степени. Известно, что возведение в квадрат позволяет уменьшать число слагаемых с неизвестными радикалами лишь для малого числа таковых. Если

их нельзя перегруппировать так, что в каждой части будет не более двух радикалов, то возведение в квадрат нецелесообразно. Поэтому в решателе отдельно рассмотрены все 4 случая для возможного числа слагаемых с радикалами - от 1 до 4. В случае нескольких радикалов в приемы вводится учет особых обстоятельств, позволяющих рационально сгруппировать радикалы перед возведением в квадрат. Уровень срабатывания приемов достаточно высок - начиная с 5. Это сделано для того, чтобы перед возведением в квадрат был, хотя бы бегло, рассмотрен блок анализа задачи и, по мере возможности, использованы более простые средства решения. Например, в случае системы уравнений более эффективным может оказаться использование для исключения радикала линейных комбинаций уравнений; в случае одной неизвестной уравнение может быть решено как квадратное относительно радикала, и т.п.

Первые фильтры рассматриваемого приема фиксируют общий контекст срабатывания: уровень 5; уравнение является условием задачи на описание; выражение под радикалом содержит неизвестные; правая часть уравнения известна. Далее идет фильтр, требующий, чтобы среди остальных слагаемых левой части не имелось ни одного, чьим множителем был бы радикал четной степени. Фактически в фильтре сформулировано немного более общее условие - отсутствие показателя степени, коэффициент знаменателя которого делился бы на 2.

Далее расположено несколько странное, на первый взгляд, условие - если уравнение имеет более одной неизвестной, а число слагаемых его левой части более одного, то не должно иметься другого уравнения, в левой части которого находится не менее двух слагаемых с неизвестными радикалами. Оно возникло при рассмотрении задачи, где после возведения в квадрат второго уравнения появился такой же радикал, как и в первом, и далее для исключения неизвестного радикала удалось использовать линейную комбинацию уравнений. Эта ситуация подсказала, что разумнее начинать возведение в квадрат с уравнений, имеющих более одного радикала. Разумеется, такой фильтр можно было бы сделать более точно учитывающим возможность получения в разных уравнениях "подобных" членов с неизвестными радикалами после возведения в квадрат, но и в приведенной выше упрощенной версии он оказался пока вполне достаточным.

Следующий фильтр запрещает наличие в оставшихся слагаемых левой части неизвестных логарифмов при отсутствии неизвестного логарифма под радикалом. Он возник из рассмотрения уравнений, которые вообще элементарными средствами не решаются. Такие уравнения часто появляются, например, при отыскании корней производной. Рассматриваемый фильтр ускоряет выдачу отказа в этих ситуациях, позволяя решателю переключиться во внешней задаче на другие пути решения.

Наконец, последние фильтры, судя по встречающемуся в них символу "связка", нужны при рассмотрении дифференциальных уравнений, и мы их пропускаем.

Подбираем уравнение для проверки полезности указанного в упражнении фильтра согласно приведенным выше объяснениям. Например, можно расмотреть систему $\sqrt{2x+3}\sqrt{2y+1}+y=45/8; \sqrt{2y+1}+\sqrt{2x+3}-x=5$. Отключая рассматриваемый фильтр, убеждаемся, что система не решается, в то время как при использовании фильтра находим корни $x=-3/2, y=5+5/8$.

5. Раздел приема находим, следуя вдоль пути ("Элементарная алгебра", "Плюс",

"Решение уравнений", "Системы уравнений"). Прием срабатывает на втором уровне и предназначен для преобразования уравнения - условия задачи на описание путем перехода к линейной комбинации его с другим уравнением. Применяется в ситуациях, когда задача имеет более одной неизвестной. Фильтр "не(цель(редакция))" блокирует попытки применения его на том этапе, когда происходит завершающее редактирование найденного ответа.

Выражение a идентифицируется как произведение всех неизвестных множителей некоторого слагаемого второго уравнения, причем оно же, домноженное на некоторый известный коэффициент h, встречается и среди слагаемых первого (преобразуемого) уравнения. Левые части уравнений не должны иметь дробных слагаемых - так как для устранения таких слагаемых имеются другие приемы.

Фильтр "или(числонеизвестных(...)...)" требует, чтобы результирующее уравнение либо имело единственную неизвестную, либо, после разложения на множители его левой части f, распалось на несколько уравнений, соединенных связкой "или", либо было линейным по всем своим неизвестным, либо его левая часть имела меньшее число слагаемых с неизвестными дробными степенями, чем преобразуемое уравнение. Любая из этих ситуаций является некоторым эвристическим "достижением", ради которого и выполняется преобразование.

Фильтр "не(входит(независит цели))" относится к ситуациям, в которых значения неизвестных не должны зависеть от явно указанных "варьируемых" параметров задачи. Обычно в этих ситуациях значения неизвестных подбираются так, чтобы при подстановке их пропадала зависимость условий от варьируемых параметров (например, содержащее эти параметры выражение было бы домножено на 0). Поэтому надобность в данном приеме, ориентированном на обычные системы уравнений, отпадает.

Следующий фильтр - "или(равно(число(неизвестная(...)...)...)...)" - ускоряющий. Его проверка предшествует вычислению левой части f нового уравнения, и здесь происходит предварительная оценка преобразования. Он анализирует выражения b, d, которые составлены из слагаемых левых частей уравнений, остающихся в линейной комбинации. Будем называть их "остальными слагаемыми". Фильтр разрешает продолжение действий в следующих случаях: общее число неизвестных в остальных слагаемых равно 1 (тогда и f будет иметь единственную неизвестную); остальные слагаемые линейны относительно входящих в них неизвестных (тогда и f будет линейно); остальные слагаемые содержат члены, пропорциональные уничтожаемым в линейной комбинации выделенным слагаемым (тогда эти члены тоже пропадут в линейной комбинации); выражение a имеет своим множителем неизвестную дробную степень, а слагаемые выражения b - не имеют таких множителей (тогда в линейной комбинации станет одним слагаемым с дробными степенями меньше).

Далее отсекается случай, когда до и после преобразования уравнение имеет ровно одну неизвестную. Без этого легко можно было бы привести пример с зацикливанием на данном приеме.

Фильтр "не(Входит(известно цели))" - ускоряющий; он отсекает преобразования для задач, ответ которых должен быть выражен через специально заданное множество параметров. В этом случае решение происходит только за счет вывода следствий в блоке анализа.

Заметим, что все перечисленные выше фильтры приема возникали в процессе проработки потока задач и аккумулировали в себе лишь те особые случаи, которые этим потоком были подсказаны. Никакого общего критического анализа их вне данного процесса не предпринималось - это лишь сырой материал. По-видимому, наиболее естественными рамками, в которых такой анализ стоило бы предпринять, является разработка процедур автоматического синтеза приемов. Этому будет посвящена третья книга данной монографии. Однако, даже и с такими фильтрами, решатель, в целом, оказывается достаточно работоспособен.

В качестве примера системы уравнений, на которой прием должен сработать, рассмотрим какую-либо систему с двумя неизвестными, дающую линейную комбинацию с единственной неизвестной. Например, пусть это будут уравнения $2x^2 + 3y^2 = 8, 3x^2 + 4y = 1$. Заметим, что если бы члены с y оба были линейными, то сработал бы другой прием для получения линейной комбинации, имеющий меньший уровень срабатывания.

6. Введем в задачник задачу на упрощение в о.д.з. выражения

$$\sum_{i=1}^{n} \sin(3n+1)^2.$$

Напомним, что степень тригонометрической операции ставится формульным редактором не сразу после ее обозначения, а лишь после обозначения ее аргумента (чтобы получить степень аргумента, нужно взять ее в скобки). Далее нажатием "р" (кир.) запускаем трассировку по шагам решения. Первым же шагом выполняется суммирование, о чем и сообщает поясняющий текст. Нажимаем "б" и входим в просмотр приема, выполнившего преобразование. Затем последовательно нажимаем End (возвращение в просмотр текущего преобразования), Esc (возвращение в исходный текст задачи), End (возвращение в главное меню) и "г" (вход в оглавление базы приемов). В результате окажемся на концевом разделе, содержащем сработавший прием суммирования. Возвращаясь от него "обратным ходом" к корню оглавления приемов, определяем цепочку надразделов приема: ("Элементарная алгебра", "Комбинаторные функции", "Суммавсех", "Тригонометрические суммы", "Суммирование квадратов синусов либо косинусов").

7. Прием находится в разделе ("Элементарная геометрия", "Фигуры", "Окружность", "Описанная около фигуры окружность", "Треугольник", "Теорема синусов"). Переходим в оглавление задачника, входим в раздел "Элементарная геометрия", нажимаем Shift-2 и, переходя в просмотр последних четырех приемов на теорему синусов, каждый раз нажимаем "А" (кир.). Затем возвращаемся в задачник, нажимая Shift-3. Переходим в буфер задачника и предпринимаем анализ избыточности срабатываний приемов - последовательно для каждого подраздела буфера, используя для запуска автоматического анализа клавишу "й". Для первого приема имеем серию из 49 задач, причем в 14 из них его срабатывание оказывается избыточным. Второй прием сработал в единственной задаче, да и там его срабатывание оказалось избыточно. Видимо, на момент создания приема он применялся в какой-то другой задаче, решение которой впоследствии пошло по другому руслу. Третий прием сработал тоже на единственной задаче, но здесь уже он оказался нужен. Наконец, четвертый прием

срабатывает в 12 задачах, и в 4 из них - избыточен. Попытки оптимизации этих приемов предоставляем читателю.

8. Приемы расположены в разделе ("Элементарная геометрия", "Фигуры", "Касательная", "Касательная и секущая", "Длина касательной и длины отрезков секущей"). Последние три приема отличаются друг от друга только уровнями срабатывания и фильтрами. Каждый из них выписывает соотношение равенства квадрата длины касательной произведению длин отрезков секущей.

   Первый прием срабатывает при малых значениях текущего уровня - 3 либо 7. Условия срабатывания одинаковы для каждого из этих уровней; если на третьем уровне прием не сработает, то по достижении седьмого уровня могут появиться новые посылки, при которых его срабатывание уже станет возможным. Если бы не был предусмотрен дополнительный уровень срабатывания, то нужно было бы специально заботиться о переключении внимания - уменьшении до 3 веса посылки "касательная(...)", на которой инициализируется применение приема.

   Ограничения на срабатывание необременительны - они сводятся к тому, чтобы некоторое количество расстояний, относящихся к рассматриваемым объектам, уже были явно указаны в задаче. Один из фильтров требует, чтобы в задаче уже встречались либо длина касательной, либо обе длины отрезков секущей, либо чтобы секущая проходила через центр окружности. Другой фильтр требует, чтобы уже была рассмотрена длина хотя бы одного из трех отрезков, выделяемых на секущей. Последние два фильтра усиливают эти ограничения, если результирующее соотношение имеет более одного расстояния, не выраженного через числовые параметры.

   Второй прием срабатывает на уровне 9. Для его срабатывания нужно, чтобы хотя бы два из трех выделяемых на секущей отрезков были либо вычислены, либо достаточно тесно связаны с неизвестными (в стандартном для планиметрии смысле, определяемом фильтром "неизв(...)"). Кроме того, должно быть выделено расстояние от какого-либо конца рассматриваемого отрезка касательной до другой точки на этой же касательной.

   Уровень срабатывания третьего приема равен 13. Для его срабатывания нужно, чтобы длина отрезка касательной была сязана с неизвестными, а число проходящих через рассматриваемую вне окружности точку секущих было не более 2.

   В том же разделе есть еще четыре приема, выписывающих то же самое соотношение. Каждый раз, когда при решении задачи в нем возникала потребность, причем имелась какая-то специфическая особенность, отсутствовавшая в предыдущих ситуациях, - вводился дополнительный прием. Взятые вместе, все эти приемы составляют некоторую эвристическую аппроксимацию множества случаев, в которых данное соотношение оказывается полезным, разумеется, пока чрезвычайно сырую. Фильтры геометрических приемов, как правило, прослеживают возможность срабатывания каких-то других приемов, следующих за срабатыванием данного. Чтобы они не отсекали нужных срабатываний, такой учет последействий должен быть достаточно полным, а чтобы не происходило бесполезных срабатываний, он должен быть продолжаем на возможно большую глубину. Пока учет последействий в приемах достаточно эпизодиче-

ский, и ценность его состоит главным образом в том, что он дает большое количество примеров для дальнейшего анализа.

Для определения задач, в которых срабатывание приема избыточно, поступаем так же, как в предыдущем упражнении. Первый прием срабатывает в 22 задачах, причем в трех из них - избыточным образом; второй прием срабатывает в трех задачах, и во всех из них нужен; третий прием срабатывает в единственной задаче и для нее необходим.

9. Прием находится в разделе ("Элементарная алгебра", "Тригонометрия", "Тангенс", "Уравнения", "Переход к тангенсу половинного аргумента"). Прием срабатывает на шестом уровне, то есть после того, как все более простые попытки решения уравнения оказались неудачными. Некоторых пояснений требует вид теоремы приема. Левая часть эквивалентности - преобразуемое уравнение $a = b$; указатель "контекст(позиция(x7 x1)вид(x7 косинус(x4)))" определяет идентификацию неизвестного тригонометрического аргумента d по косинусу этого аргумента, встречающемуся в уравнении. Нормализатор "половинныйугол" выражает встречающиеся в левой части уравнения синусы, косинусы и тангенсы через тангенс половинного аргумента (тангенс выражается через половинный угол лишь тогда, когда этот половинный угол уже возник в выражении). Фильтры приемы требуют, чтобы кроме косинуса d уравнение имело либо синус d, либо тангенс половинного аргумента. Кроме того, проверяется, что каждая неизвестная тригонометрическая операция в уравнении совпадает либо с синусом d, либо с косинусом d, либо с тангенсом d, либо с тангенсом половинного аргумента e. Чтобы отсечь случаи чрезмерного возрастания сложности уравнения при переходе к половинному аргументу, не разрешаются степени неизвестных тригонометрических операций, отличные от второй и третьей. Наконец, прием блокируется при рассмотрении дифференциальных уравнений.

   В качестве задачи для тестирования приема будем искать уравнение вида $\sin x + \cos x + \operatorname{tg} x = a$. Величину a подберем так, чтобы уравнение решалось: проследим действия решателя до появления уравнения $(a+1)y^4 - 2y^2 + 4y = a - 1$ относительно вспомогательной неизвестной y, после чего положим $y = 1/2$ и определим $a = 41/15$. При таком значении a решатель находит все корни. Уравнение имеет две их серии, и для одной из них применяется формула Кардано.

10. Прием находится в разделе ("Элементарная алгебра", "Плюс", "Решение уравнений", "Системы уравнений", "Выражение неизвестной X из уравнения $X + A = B$"). Он преобразует уравнение $b + c = a$, где b - неизвестная; c - выражение с неизвестными, не содержащее b; a не содержит неизвестных, к виду $b = a - c$. Такое преобразование немедленно вызывает подстановку в остальные условия задачи выражения $a - c$ вместо b, т.е. фактически равносильно исключению этой неизвестной.

    Указано три различных уровня срабатывания - 0,1 и 5. Фактически уровни 0,1 - альтернативные: в случаи задачи на описание допустим только уровень 1, в случае задачи на исследование - уровень 0. Такое разделение уровней по типу задачи, видимо, объясняется тем, что в какой-то задаче на исследование (при анализе объединенного списка посылок и условий внешней задачи на описание) понадобилось ускорить выражение неизвестной из данного уравнения, чтобы не возникло состязание с другим приемом, срабатывающим на первом уровне.

Подробности можно уточнить, скорректировав данный фильтр так, чтобы и в случае задач на исследование прием срабатывал на уровне 1, и выполнив прокрутку по задачнику для поиска тех задач, которые отреагируют на данное изменение негативным образом.

Срабатывание приема на малых уровнях допускается в двух случаях: либо $a = 0$, а число неизвестных равно 2, либо уравнение линейно относительно всех входящих в него неизвестных. В прочих случаях (и только для задачи на описание) уровень срабатывания равен 5. Кроме того, для срабатывания приема необходимо выполнение одного из следующих условий: уравнение имеет ровно две неизвестные; задача - на описание и имеет не более одного нелинейного уравнения; задача - на исследование, $a = 0$, а уравнение содержит ровно 3 неизвестных.

Если неизвестная b - целочисленная, а число неизвестных в c более двух, то для задачи на описание прием блокируется: целочисленные уравнения обычно решаются другими средствами. Наконец, последний фильтр учитывает ограничения на зависимость ответа от заданных переменных - в этой ситуации также используются другие средства решения.

Чтобы определить последствия удаления указанного в упражнении фильтра, проведем прокрутку по разделам задачника ("Элементарная алгебра", "Решение уравнений", "Системы алгебраических уравнений 1,2,3,4") с предварительным удалением фильтра. Так как здесь область применения приема не сужена, а расширена, определение списка задач, в которых прием срабатывал, не поможет. Придется прибегнуть к более трудоемкой полной прокрутке по избранным разделам. Обычная практика при развитии решателя - периодические (раз в несколько дней) полные прокрутки, занимающие около 2 часов. Есть не очень сильно развитые пока средства сужения класса задач, для которых предпринимается прокрутка. Они основаны на учете всех логических символов, встречавшихся при решении задачи; если для срабатывания приема необходимо наличие какого-то символа, а при решении он не возникал, то задачу можно не рассматривать. Однако, для простейших приемов, типа рассматриваемого, такое отсечение малоэффективно.

При удалении фильтра прокрутка первых двух разделов для систем уравнений, казалось бы, показывает его избыточность: несколько задач даже "ускоряются". Однако, на 44-й задаче третьего раздела неожиданно появляется сообщение "переполнение буфера текстов". Нажимая "ф", входим в отладчик ЛОСа, поднимаемся до уровня кадра программы "исследовать" и просматриваем значение переменной x1 - текущей задачи на исследование (удобно сделать это нажатием K1). Видно, что в посылках задачи присутствует введенное данным приемом явное выражение неизвестной z через x, y, а возникшие после исключения z уравнения чрезмерно громоздки. Переполнение (получение слишком длинного набора) произошло при регистрации в задаче на исследование результатов вывода анализатором "сложениеуравнений" линейных комбинаций имеющихся уравнений. Такой результат неудивителен: при решении систем нелинейных уравнений выражение одной неизвестной через другие нужно применять очень осторожно. На ограничение этих действий и был направлен устраненный фильтр.

12.2.5 Создание нового приема

Приведем несколько несложных упражнений на самостоятельную разработку приемов. Иногда эти упражнения будут дублировать уже существующие в системе приемы. Рекомендуется сначала создать свою версию, не прибегая к подсказкам решателя, и протестировать ее на какой-либо специально введенной для этого задаче. Если при тестировании выяснится, что старые приемы решателя опережают новый прием и не дают ему сработать, то следует либо взять уровень срабатывания этого приема меньшим, чем у старых приемов, либо временно отключить старые приемы. Разумеется, приводимых здесь примеров недостаточно, чтобы научиться программировать на ГЕНОЛОГе. Технике такого программирования посвящен весь второй том данной монографии, в котором подробным образом будут рассматриваться разделы базы приемов решателя.

1. Создать прием, упрощающий тангенс утроенного арктангенса.

2. Создать прием для упрощения радикала второй степени, под которым расположен полный квадрат суммы с другим радикалом. То же самое - для радикалов третьей степени.

3. Создать прием, решающий простейшее уравнение для секанса: $\sec x = a$.

4. Создать прием, находящий пересечение двух арифметических прогрессий с общим начальным членом, положительные разности которых относятся как целые числа.

5. Создать прием для суммирования кубов синусов элементов арифметической прогрессии.

6. Создать прием, связывающий площадь вписанного в окружность четырехугольника с длинами его сторон и полупериметром.

7. Создать прием, основанный на теореме о том, что точки пересечения высот, медиан и срединных перпендикуляров в треугольнике лежат на одной прямой. Придумать планиметрическую задачу на вычисление, которая могла бы быть решена решателем только после добавления данного приема.

8. Создать прием, переформулирующий через коэффициенты квадратного трехчлена условие, что заданная величина x больше его корней. Придумать задачу, в которой этот прием мог бы понадобиться.

9. Создать прием, дающий разложение тангенса в ряд с использованием чисел Бернулли.

10. Создать прием, позволяющий вычислять длину кривой, заданной уравнением $y = y(x)$ в прямоугольной системе координат.

11. Создать прием, позволяющий переходить от прямоугольных координат множества точек в трехмерном пространстве к сферическим.

Указания

1. Разумеется, начинается синтез приема с того, что определяется его теорема. Обычно это связано с решением несложной задачи, приводящей к нужной теореме, либо с обобщением какого-то известного утверждения путем ввода в него дополнительных параметров, учитывающих типичные контексты его применения в задачах. В нашем случае нужно вывести формулу тангенса тройного арктангенса и получить теорему $\operatorname{tg}(3\operatorname{arctg} a) = (3a - a^3)/(1 - 3a^2)$. Так как это - теорема приема, то условие на область допустимых значений (отличие знаменателя от 0) можно опустить - при правильной работе других приемов оно должно усматриваться из контекста преобразуемого выражения.

   Далее выбираем раздел, в котором будет размещен прием. В нашем случае естественно пойти по цепочке подразделов ("Элементарная алгебра", "Тригонометрия", "Тангенс", "Тангенс выражения, содержащего обратные тригонометрические функции"). Вводим для приема новый концевой раздел (нажатием "к"), сопроводив его, например, текстом: "Тангенс утроенного арктангенса". Далее набираем теорему приема и вводим его заголовок - "второйтерм". Так как прием выполняет преобразование, которое, по-видимому, практически всегда будет полезным, уровень срабатывания его выбираем небольшим - например, равным 3 (все-таки, здесь лучше немного подождать - вдруг на уровнях от 0 до 2 этот тангенс тройного арктангенса будет устранен как-то иначе, например, умножен на 0). Других фильтров не вводим. Указатели приему также не требуются. Сохранив (F4) набранную заготовку приема, нажимаем "н" для получения подсказок от генератора приемов о возможных дополнительных элементах описания приема. Сначала возникает указатель "логсимвол(3 умножение)", который говорит компилятору, что тройка здесь должна идентифицироваться как непосредственный операнд произведения. Однако, в нашем случае компилятор и сам это понимает, так что указатель можно не вводить. Далее идут подсказки о вводе нормализаторов общей стандартизации - их принимаем, нажимая каждый раз клавишу Insert. По окончании появления подсказок (возникает голубая линия) прием снова нужно сохранить. Так как он, по существу, уже готов, то вместо F4 сразу нажимаем F3. Наконец, тестируем работу приема. Например, вводим выражение $\operatorname{tg}(3\operatorname{arctg}(1/5)$ и на первом же шаге трассировки получаем $37/55$.

   Заметим, что попытки протестировать прием на каком-либо значении, для которого арктангенс известен, например, для тангенса $\pi/12$, сразу приведут к срабатыванию старых приемов, и чтобы в этом случае протестировать новый прием, их придется найти и временно отключить. Если эта работа будет проделана, то мы обнаружим, что наш прием вводит сравнительно громоздкое выражение с радикалом, которое вызывает длинную цепочку упрощающих его простых преобразований. Чтобы избежать этого, можно добавить нормализаторы "видумножение" для обработки числителя и знаменателя - после этого пример будет решаться примерно вдвое быстрее.

2. Будем рассматривать простейший случай, когда под радикалом находится выражение $a + b\sqrt{c}$, представляющее собой полный квадрат другого выражения $p + q\sqrt{c}$, причем p, q не имеют радикалов. Возводя в квадрат и рассматривая систему уравнений для неизвестных p, q при известных a, b, c, получаем: $q^2 = (a + \sqrt{a^2 - cb^2})/2c \ \vee \ q^2 = (a - \sqrt{a^2 - cb^2})/2c$; $p = b/2q$. Это подсказывает

следующую схему вычислений, выполняемых приемом. Сначала предпринимается попытка разложить на множители выражение $a^2 - cb^2$ и проверить, что результат является полным квадратом некоторого выражения r. Затем рассматриваются выражения $a + r, a - r$ и предпринимается попытка представить какое-либо из них в виде произведения $2c$ на полный квадрат некоторого выражения - оно и будет равно q. Наконец, по b и q определяется p. Теорема приема, соответствующая такой последовательности действий, имеет вид:

$\forall_{abcpqr}(r^2 = a^2 - cb^2 \ \& \ (a + r = 2cq^2 \vee a - r = 2cq^2) \ \& \ b = 2pq \to \sqrt{a + b\sqrt{c}}) = |p + q\sqrt{c}|)$

После набора этой теоремы вводим остальные компоненты описания приема: заголовок - "второйтерм"; фильтр для уровня срабатывания "уровень(0)" и проверяющий отсутствие радикалов в коэффициентах a, b фильтр "не(контекст(список(x5 a b)позиция(x6 x5)вид(x6 степень(x7 дробь(1 2)))))"; указатели "единица(1 b p q)", "заменазнака(минус b p)". Нулевой уровень срабатывания здесь введен, чтобы при тестировании не срабатывали другие приемы, уже имеющиеся в решателе. Впрочем, их уровень срабатывания также маленький - лучше сразу усмотреть возможность избавиться от двойного радикала, чем выполнять с ним какие-то действия. Выражения $a + r, a - r$ могут не иметь заголовка "умножение", а быть числовыми константами. В этом случае идентификация их с произведением $2cq^2$ будет заключаться в делении на число $2c$ и проверке того, что частное является полным квадратом. Чтобы предусмотреть эту ситуацию, добавляем указатель "пересечениесписков(фикс(2 1 2)фикс(2 2 2))", отменяющий проверку наличия у данных выражений заголовка "умножение".

Далее вводим указатель "идентификатор(1 2 3)" и расставляем указатели общей стандартизации. Наконец, добавляем указатели нормализации "видумножение" для разложения на множители выражений $a^2 - cb^2$, $a + r$, $a - r$.

Тестируем прием на какой-либо задаче, например, $\sqrt{2\sqrt{2} + 3}$.

Создание аналогичного приема для радикалов третьей степени, а также возможные обобщения приведенного выше приема оставляем читателю.

3. Теорему приема нетрудно получить, перейдя от уравнения $\sec x = a$ к уравнению $\cos x = 1/a$:

$\forall_{ax}(\sec x = a \leftrightarrow (0 \le 1 + 1/a \ \& \ 0 \le 1 - 1/a \ \& \ (\exists_n(\text{целое}(n) \ \& \ x = \arccos(1/a) + 2\pi n) \ \vee \ \exists_n(\text{целое}(n) \ \& \ x = -\arccos(1/a) + 2\pi n))))$.

Вместо приведенных здесь двух неравенств для a можно было бы использовать неравенство для модуля $1 \le |a|$. Однако, следует помнить, что эти неравенства будут обеспечивать сопровождение по области допустимых значений для арккосинуса. Так как проверки здесь будут относится к выражению $1/a$, то переход к модулю существенно их затруднит, и лучше его не вводить. С другой стороны, от знаменателей в неравенстве можно избавиться, после чего теорема приема примет вид:

$\forall_{ax}(\sec x = a \leftrightarrow (0 \le a(a + 1) \ \& \ 0 \le a(a - 1) \ \& \ (\exists_n(\text{целое}(n) \ \& \ x = \arccos(1/a) + 2\pi n) \ \vee \ \exists_n(\text{целое}(n) \ \& \ x = -\arccos(1/a) + 2\pi n))))$.

Уровень срабатывания приема выбираем равным 2 (как и для прочих тригонометрических операций). Контекст срабатывания задается фильтрами "условие", "тип(описать)", "корень", "известно(a)", "не(известно(x))". Добавляем

также имеющийся у других аналогичных приемов фильтр "конец(не(фильтрсерии(a)))", который блокирует данный переход из общих соображений. Например, если уже получено параметрическое описание для какой-либо неизвестной, встречающейся в x, то вместо применения данного приема выгоднее подставить в выражение x явное значение этой неизвестной. Указатель приема вводим единственный - "примечание(серия)". Он сопровождает преобразованное условие комментарием "серия", поясняющим, что встречающиеся в нем кванторы существования дают параметрическое описание и нет смысла пытаться устранить их, получив явное разрешение подкванторного утверждения относительно связанной переменной.

В заключение расставляем нормализаторы общей стандартизации. При этом нормализатор "нормарккосинус" сопровождаем указателем "посылки(и(целое(n)меньшеилиравно(0 $a(a+1)$)меньшеилиравно(0 $a(a-1)$)))".

4. Множество членов арифметической прогрессии будем записывать в виде $\mathrm{set}_x(\exists_i($ целое(i) & $0 \le i$ & $x = a + bi))$. Если есть две прогрессии, с положительными разностями b и c, отношение которых - рациональное число, то для определения такого отношения m/n будем использовать в теореме приема процедуру сокращения дробного выражения b/c. Чтобы убедиться в несократимости дроби m/n, добавляем проверку взаимной простоты чисел m и n - тогда результирующая прогрессия будет иметь разность bn. Таким образом, получаем теорему приема:

   $\forall_{abcmn}(b/c = m/n$ & натуральное(m) & натуральное(n) & взаимнопросты(m,n) $\to \mathrm{set}_x(\exists_i($целое$(i)$ & $0 \le i$ & $x = a + bi)) \cap \mathrm{set}_x(\exists_j($целое$(j)$ & $0 \le j$ & $x = a + cj)) = \mathrm{set}_x(\exists_i($целое$(i)$ & $0 \le i$ & $x = a + bni)))$.

   Единственный фильтр приема - "уровень(1)", так как преобразование представляется полезнмы в любых ситуациях. Заголовок приема - "второйтерм". Указатели - "идентификатор(1)", "блокпроверок(2 3)", "единица(0 a)", "единица(1 b c n)". Нормализаторы приема - обычные, в том числе нормализатор "нормдробь" для частного b/c. В принципе, можно было бы несколько усилить прием, введя также нормализатор "видумножение" для разложения на множители выражений b, c перед попыткой сокращения дроби.

   Проверяем созданный прием на каком-либо простом примере; оказывается, что после его срабатывания применяется другой прием, преобразующий результат к виду "арифмпрогрессия(a bn)". Если бы существование последнего приема было известно заранее, то можно было бы прием для пересечения формулировать сразу с использованием обозначения "арифмпрогрессия".

5. В решателе имеются приемы, позволяющие вычислять сумму синусов либо косинусов элементов арифметической прогрессии. Поэтому все, что требуется для вычисления суммы кубов синусов либо косинусов - это преобразование их к функциям кратного аргумента, так как дальше сработают указанные выше приемы. Это соображение позволяет нам создать даже более общий прием - для вычисления сумм проивзольных натуральных степеней синусов и косинусов. Теорема такого приема будет, по существу, вырожденной - в ней должно содержаться только обращение к нормализатору "стандплюс" - он выполняет не только раскрывание скобок, но и переход к кратному тригонометрическому аргументу, если при обращении ввести комментарий "видумножение". Записы-

вается данное обращение так:

$$\forall_{abmnkp}((\sin(a+bi))^k = p(i) \to \sum_{i=m}^{n}(\sin(a+bi))^k = \sum_{i=m}^{n} p(i)).$$

Здесь $p(i)$ - вспомогательное обозначение для результата перехода к кратному аргументу. Заголовок приема - "второйтерм". Уровень срабатывания его выберем равным единице. Добавим фильтры "натуральное(k)" (показатель степени есть десятичная запись натурального числа) и "меньше(k 6)" (отсекаем слишком большие показатели степени, так как пока неясно, нужно ли будет для них применять данный переход). Вводим стандартный набор указателей - "идентификатор(1)", "отображение(p)", "единица(0 x1)", "заменазнака(минус x2)", "единица(1 x2)". Левую часть антецедента теоремы снабжаем нормализаторами "стандплюс(замечание(видумножение))", "посылки(и(целое(i) меньшеилиравно(m i) меньшеилиравно(i n)))". Указатель нормализации "посылки(...)" введен из-за того, что преобразуемое выражение содержит переменную i, определенную лишь в контексте суммирования, и нормализатору может понадобиться дополнительная информация о значении этой переменной.

6. Формулу, связывающую площадь вписанного четырехугольника с длинами его сторон a, b, c, d и полупериметром p, нетрудно вывести или взять из учебника: $S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$. Здесь имеются пять параметров - S, a, b, c, d. Из общих соображений, выписывать соотношение для них имеет смысл, если достаточно большая их часть известна либо представляет интерес для нахождения значений неизвестных. Желательно, чтобы хотя бы один из параметров при этом был неизвестен - иначе соотношение, в лучшем случае, может понадобиться лишь для усмотрения противоречивости описания чертежа. Вообще-то, новые приемы вводятся в решатель только в том случае, когда они понадобились для решения какой-то конкретной задачи. Здесь мы отступаем от этого правила - и это может сказаться на качестве приема. Если нет примера, подтверждающего полезность приема, то не исключено, что он вообще будет избыточен. Поэтому, доведя упражнение до конца, попробуем найти задачу, которая без данного приема решаться системой не будет.

   Прежде чем вводить теорему приема, нарисуем геометрическим редактором чертеж - проведем окружность EF и изобразим вписанный в нее четырехугольник $ABCD$. Теорема приема, кроме приведенного выше соотношения, должна содержать также описание геометрической ситуации:

   $\forall_{ABCDEFabcdpS}$(окружность(EF)описана около фигура($ABCD$) & актив(S(фигура($ABCD$))) & $p = (l(AB)+l(BC)+l(CD)+l(AD))/2 \to S$(фигура($ABCD$)) = $\sqrt{(p-l(AB))(p-l(BC))(p-l(CD))(p-l(AD))}$).

   Второй антецедент означает, что в задаче уже рассматривается площадь четырехугольника - в этом случае автоматически вводится явная ссылка на площадь, представляющая собой фиктивную посылку вида "актив(площадь(...))". Третий антецедент вводит вспомогательное обозначение p для полупериметра, который четырежды упоминается в консеквенте теоремы.

   Прием имеет заголовок "вывод". Уровень его срабатывания выберем средним - для геометрии в качестве такого уровня годится, например, шестой. Соответственно, имеем фильтры "уровень(6)", "тип(исследовать)".

Переходим к основному фильтру, который будет рассматривать список связываемых выводимым равенством параметров. Начнем этот фильтр с перечисления значений данных параметров: "контекст(равно(х5 набор(терм(расстояние(AB)) терм(расстояние(BC)) терм(расстояние(CD)) терм(расстояние(AD)) терм (площадь(фигура(набор($ABCD$)))))).... Так как мы перечисляем в наборе х5 термы, определяемые теоремными выражениями, то каждое из них заключаем внутри записи "терм(...)". В действительности, нас будут интересовать не выражения "расстояние(...)", "площадь(...)", а значения этих выражений, определяемые на текущий момент из контекста задачи. Чтобы получить такие значения, будем обрабатывать выражения нормализаторами "нормрасстояние" и "нормплощадь".

После того, как набор х5 значений параметров определился, продолжаем запись фильтра: ...входит(х6 х5) неизв(х6) меньше(число(входит(х7 х5) не(известно(х7)) не(неизв(х7)))2) меньше(1 число(входит(х7 х5) известно(х7)))". Здесь требуется, чтобы хотя бы один параметр был не известен и связан по цепочке соотношений с неизвестными внешней задачи на описание; чтобы имелось не более одного неизвестного параметра, не связанного таким образом с неизвестными внешней задачи; чтобы не менее двух параметров были известны. Разумеется, это - лишь нулевая версия возможного фильтра, и ее стоит проверить путем прокрутки по задачнику. Если окажется, что в каких-то задачах срабатывание приема является вредным, то фильтр придется усиливать. Заметим, что пока в область допустимых срабатываний попадают все ситуации прямого определения единственного неизвестного параметра через известные остальные параметры.

Единственный указатель приема - "идентификатор(3)"; нормализаторы берем все, которые подсказывает система автоматического пополнения описания приема.

Для тестирования приема вводим простейшую задачу, чертеж которой в точности совпадает с условием теоремы - четырехугольник, вписанный в окружность и имеющий известные длины своих сторон, например, 3,5,7,4. Требуется в задаче найти площадь четырехугольника. Нажимая Ctr-ч, получаем достаточно точно воспроизводящий значения длин сторон чертеж. При трассировке убеждаемся в том, что прием срабатывает. При его отключении (на момент написания данного примера) решатель задачи не решает.

7. Начинаем создание приема с построения геометрическим редактором чертежа. Рисуем треугольник ABC, проводим высоты AD и BE, медианы BF и AG, а также срединные перпендикуляры FQ и GQ, продолженные до их точки пересечения Q. Вводим также точки P, R пересечения высоты и медиан соответственно. Затем переходим к набору теоремы:

$\forall_{ABCDEFGPQR}(\triangle(ABC)$ & $D \in$ прямая(BC) & $E \in$ прямая(AC) & прямая(AD) $\perp$прямая(BC) & прямая$(BE)\perp$прямая(AC) & $G \in$ прямая(BC) & $l(BG) = l(CG)$ & $F \in$ прямая(AC) & $l(AF) = l(CF)$ & прямая$(FQ)\perp$прямая(AC) & прямая$(GQ)\perp$прямая(BC) & $P \in$ прямая(AD) & $P \in$ прямая(BE) & $R \in$ прямая(AG) & $R \in$ прямая$(BF) \rightarrow Q \in$ прямая$(PR))$.

Заголовок приема - "вывод"; уровень срабатывания выбираем не очень большим, так как утверждение о принадлежности трех точек одной прямой может оказаться существенным для всего дальнейшего ходя решения. Например, можно взять его равным 4. Обычно такой произвольный выбор уровня срабатывания корректируется при регулировочной прокрутке приема на задачах. Вводим фильтр "или(тип(доказать)тип(исследовать))". Каких-то особых соображений об ограничениях на срабатывание данного приема не возникает, и других фильтров пока не вводим. Для ввода указателей и нормализаторов используем автоматическое пополнение описания приема - нажимаем "н" и соглашаемся с появляющимися предложениями.

Для тестирования приема введем задачу, чертеж которой совпадает с чертежом теоремы; зададим какие-то конкретные длины сторон треугольника, и потребуем найти отношение длин отрезков PR и RQ. При срабатывании приема решатель сравнительно быстро находит ответ; при его отключении ответ не выдается.

8. Условие на x запишем в таком виде: $\forall_y(ay^2 + by + c = 0 \to y < x)$. Заметим, что оно вовсе не означает наличие у трехчлена корней; если корни есть, то они должны быть меньше x, а если их нет, то никаких дополнительных ограничений не требуется. В этой ситуации, как легко видеть, условие эквивалентно следующей альтернативе: либо дискриминант неотрицателен, значение трехчлена в точке x имеет тот же знак, что и коэффициент a, а сама точка x расположена правее точки $-b/2a$ экстремума трехчлена, либо дискриминант отрицателен. Это дает следующую теорему приема:

$\forall_{abcx}(\neg(a = 0) \to (\forall_y(ay^2 + by + c = 0 \to y < x) \leftrightarrow (0 \le b^2 - 4ac \ \& \ 0 <$

$a(ax^2 + bx + c) \ \& \ -b/2a < x) \vee (b^2 - 4ac < 0))).$

Заголовок приема - "второйтерм". Уровень срабатывания его выберем не очень большим - устранение квантора может существенно упростить задачу, и откладывать его не имеет смысла. Пусть, например, этот уровень будет равен 2. Для ввода указателей и нормализаторов прибегаем к подсказкам системы автоматического доопределения приема; со всеми этими подсказками соглашаемся. Чтобы протестировать прием, вводим какую-либо несложную задачу, например, задачу с неизвестной b и условием $\forall_x(3x^2 + bx + 5 = 0 \to x < 4)$. Запускаем трассировку по шагам решения, и обнаруживаем, что наш прием не срабатывает, а вместо него применяется другой прием, предпринимающий попытку явного разрешения подкванторного утверждения относительно переменной кванторной приставки (т.е. относительно x). На этом пути решатель, разумеется, получает ответ. Однако, такое решение, все же, более трудоемко, чем предлагаемое нашим приемом. Чтобы "опередить" старый прием, понижаем уровень срабатывания нового приема до 1. Снова запускаем трассировку, и опять оказывается, что прием не сработал. При более внимательном рассмотрении ситуации обнаруживаем, что решатель, учитывая условия на область допустимых значений, ввел под квантор, кроме условия равенства трехчлена нулю, также утверждение "число(x)". Такого рода пополнение антецедентов кванторных импликаций условиями на тип значений переменных - следствие используемой в решателе общей стандартизации, изменять которую сейчас нежелательно. Поэтому приходится изменить сам прием - ввести в него под квантор общности дополнительный антецедент "число(y)". Лишь после этого трас-

сировка покажет срабатывание приема, а при выдаче ответа будет отмечено небольшое ускорение по сравнению с запуском без нового приема.

Этот пример демонстрирует типичную ситуацию при обучении решателя - как правило, новый прием обязательно приводит к каким-то неожиданностям, и лишь после некоторого цикла коррекций удается получить желаемый результат. К такому явлению при обучении системы придется привыкать. Не следует воспринимать его как что-то негативное, свидетельствующее либо о малом опыте учителя системы, либо о принципиально неправильном подходе к ее обучению. Это - совершенно нормальное явление, своего рода неизбежное "трение" процесса обучения. Создать и откомпилировать прием - это лишь малая часть труда, который приходится в него вкладывать. Основная работа состоит в том, чтобы после этого преодолеть все те неожиданные барьеры, которые выявляются при тестировании. В общем-то, это неудивительно - логические процессы сложны и разнообразны, а изучение их только начинается. Для преодоления трудоемкости дрессировки решателей необходимо иметь предусмотрительность, обеспечить которую, по-видимому, сможет лишь постепенное формирование, пусть хотя бы и эмпирическим путем, и пусть хотя бы и нематематической, но все же теории логических процессов.

9. Для разложения в ряд Тейлора решатель использует нормализатор "рядтейлора". Ему передается входной комментарий (рядтейлора x b), указывающий переменную x, для которой выполняется разложение, и точку b, в которой оно выполняется. Число Бернулли с индексом i обозначается в решателе "числобернулли(i)" (см. раздел "Элементарная алгебра", "Комбинаторные функции"). Разложение в степенной ряд выражения $\mathrm{tg}\,x$ сразу обобщаем до разложения выражения вида $tg((a(x+c)^k)/d)$. Теорема приема получается следующая:

   $\forall_{abcdxk}$(натуральное(k) & $c=-b \to \mathrm{tg}((a(x+c)^k)/d) =$

   $\sum_{i=1}^{\infty}((-1)^{i-1}2^{2i}(2^{2i}-1)$числобернулли$(2i)a^{2i-1}(x+c)^{(2i-1)k)}/((2i)!d^{2i-1})$.

   Здесь b - точка разложения, передаваемая приему через комментарий к нормализатору.

   Заголовок приема - "замена(второйтерм рядтейлора)". Вводим фильтры приема "не(входит(x a))", "не(входит(x c))", "не(входит(x d))", "не(входит(x k))", необходимые для того, чтобы бесконечная сумма представляла собой степенной ряд. Комментарий, определяющий x и b, извлекаем указателем "вход(рядтейлора x b)". Далее вводим указатели "блокпроверок(1)", "идентификатор(2)", "единица(0 c)", "единица(1 a d k)", "заменазнака(минус a)" и расставляем нормализаторы общей стандартизации. Заметим, что прием служит для получения бесконечной суммы; если нужно применить локальную формулу Тейлора и ограничиться заданным числом членов, то используется нормализатор "формулатейлора". Попробуйте самостоятельно ввести аналогичный прием для последнего нормализатора, отключив имеющийся в нем прием разложения тангенса через разложения синуса и косинуса. Нахождение чисел Бернулли осуществляется нормализатором общей стандартизации "нормчислобернулли".

10. Будем считать, что кривая P задается выражением "точки(A K)", определяющим множество точек плоскости, координаты которых в системе координат K образуют множество A. На тот случай, если условие принадлежности пары чисел (x,y) множеству A не имеет вида, явно разрешенного относительно y,

предпримем в приеме попытку такого разрешения. Для полученной в результате зависимости $y = f(x)$ и определившегося при разрешении отрезка $[a, b]$ далее используем известную формулу длины кривой. Таким образом, имеем следующую теорему приема:

$$\forall_{abfKPAB}(\text{прямкоорд}(K)\ \&\ P = \text{точки}(A, K)\ \&\ ((x, y) \in A) = (y = f(x)\ \&\ B(x))$$
$$\&\ B(x) = (a \le x\ \&\ x \le b\ \&\ \text{число}(x)) \to \text{длина}(P) = \int\limits_a^b \sqrt{1 + \left(\frac{df(x)}{dx}\right)^2}\, dx).$$

Второй антецедент теоремы идентифицирует определяющее кривую выражение "точки(A,K)". Либо изначально кривая задана таким выражением, либо в контексте имеется равенство, дающее это выражение для P. Третий антецедент нужен для получения явной зависимости $y = f(x)$. Он выражает эквивалентность двух утверждений - утверждения о принадлежности "произвольной" пары (x, y) множеству координат точек кривой A, и результата $y = f(x)\ \&\ B(x)$ явного разрешения этого условия относительно неизвестной y. Такие эквивалентности в посылках теорем приемов записываются в виде равенств - для удобства компиляции. После разрешения условия на координаты точки относительно y остаются какие-то условия $B(x)$ на независимую переменную. Чтобы получить отрезок интегрирования, эти условия необходимо явно разрешить относительно x. Это и делает четвертый антецедент теоремы.

Заголовок приема - "второйтерм". Уровень срабатывания его выбираем сравнительно небольшим. Так как в решателе уже имеется прием для вычисления длины кривой, срабатывающий на третьем уровне, лучше положить уровень срабатывания нашего приема равным 2.

Кроме стандартных указателей "идентификатор(2 3 4)", "отображение(f B)", вводим указатели "новаяпеременная(x)" и "новаяпеременная(y)", объясняющие компилятору, что переменные x, y он может выбрать произвольным образом (но без совпадения их с уже имеющимися в задаче). При разрешении условия $B(x)$ относительно x вместо нестрогих неравенств могут появиться строгие. Чтобы обобщить прием и на эти случаи, добавляем указатели "вариант(фикс(4 2 1)меньше)", "вариант(фикс(4 2 2)меньше)". Они разрешают появляться строгим неравенствам вместо нестрогих в задании промежутка для x, не изменяя прочих действий приема.

Расставляем нормализаторы приема. Во-первых, для идентификации выражения "точки(A,K)" снабжаем переменную P (изначально идентифицируемую с кривой из выражения "длина(P)") нормализатором "смточки". Это - очень простой вспомогательный нормализатор, заменяющий P на выражение "точки(...)", если в контексте идентификации встречается равенство "P =точки(...)". Далее, для обработки левой части третьего антецедента вводим обращение к вспомогательной задаче на описание, разрешающей эту часть относительно y. Оно имеет следующий вид: "задача(6 тип(описать)полный явное буфер прямойответ упростить цель(неизвестная(y)))". Цель "буфер" добавлена для того, чтобы на некоторое время запомнить результат решения данной задачи и при повторных попытках ее решения, предприниаемых другими приемами, сразу извлекать из него готовый результат. Надобность в этом может возникнуть, если создать прием, предпринимающий попытку разрешения урав-

нения кривой относительно y для усмотрения распадения кривой на несколько ветвей и разбиения задачи на подзадачи вычисления длин отдельных ветвей.

Для обработки левой части четвертого антецедента вводим аналогичный указатель нормализации: "задача(5 тип(описать)полный явное буфер прямойответ упростить цель(неизвестная(x)))".

Наконец, для вычисления интеграла сопровождаем его указателем нормализации "задача(6 упростить)"; радикал под интегралом - указателем "задача(4 упростить одз)"; производную - указателем "задача(5 упростить одз)". Такое принудительное выделение этапов вычисления во многих случаях существенно ускоряет получение результата.

11. Пусть в прямоугольной системе координат K множество точек A задается соотношением коорд$(A,K) = set_{xyz}(f(x,y,z) = 0$ & число(x) & число(y) & число$(z))$. Тогда для перехода к сферическим координатам можно записать следующую теорему приема:

$\forall_{AKf}$(прямкоорд(K) & коорд$(A,K) = set_{xyz}(f(x,y,z) = 0$ & число(x) & число(y) & число$(z)) \rightarrow$ сферкоорд$(A,K) = set_{abc}(f(a\cos c\cos b, a\cos c\sin b, a\sin c) = 0$ & $0 \le a$ & $-\pi \le 2c$ & $2c \le \pi$ & $-\pi \le b$ & $b \le \pi))$.

Так как решатель обычно устраняет знаменатели в уравнениях и неравенствах с дробями, то неравенства для широты домножены на 2 уже в теореме.

Заголовок приема - "вывод". Поводом для его срабатывания пусть будет упоминание в посылках задачи выражения "сферкоорд(A K)". В этом случае уровень срабатывания можно сделать небольшим, например, равным 3. Лучше проявить предусмотрительность на тот случай, если соотношение для прямоугольных координат множества A появится в задаче несколько позднее, и добавить еще одну попытку применения приема на более высоком уровне - например, на шестом. В результате появится фильтр "уровень(3 6)". Далее добавляем стандартные фильтры "посылка", "тип(исследовать)". Чтобы избежать повторных применений приема, нужно ввести еще какой-то ограничитель. Пусть это будет фильтр, проверяющий отсутствие в посылках равенства, дающего выражение для сферических координат множества A через описатель "класс" - "не(контекст(посылка(x1)вид(x1 равно(сферкоорд(AK)x2))заголовок(x2 класс)))".

Далее вводим указатели "отображение(f)" и "контрольвывода(сферкоорд(A K))". Последний из них и обязывает компилятор начать программу приема с усмотрения в задаче выражения "сферкоорд(A K)". Чтобы решатель не поменял местами левую и правую части равенства (при прочих равных условиях влево переносится более длинное выражение), добавляем указатель "примечание(ориентацияравенства)". Он вводит комментарий "ориентацияравенства" к новой посылке, блокирующий на последующих шагах перестановку частей равенства. Далее, вводим указатель "примечание(нормкоорд)". Он сопровождает посылку комментарием "нормкоорд", блокирующим попытки исключения описателя "класс" путем явного разрешения условия принадлежности этому классу. Такие попытки для множеств, заданных уравнениями, обычно бесполезны.

Чтобы упростить уравнение для новых координат, сопровождаем утверждение под описателем "класс" нормализатором "задача(6 тип(описать) полный

прямойответ явное упростить цель(неизвестная(a b c)))". Это накладывает достаточно сильные требования на уравнение множества - прием пытается получить явное соотношение, выражающее одну из сферических координат через две другие. Чтобы избежать слишком долгого времени его применения, вводим в заключение ограничитель трудоемкости - указатель "лимит(10000000)".

Глава 13

Программы общего интерфейса системы

Использование логической системы в качестве инструмента для исследования логических процессов связано с постоянным обращением к процедурам ее интерфейса - для включения в них новых блоков либо адаптации старых к новым потребностям. Еще более интенсивная работа с процедурами общего интерфейса будет нужна при использовании ее в качестве основы для различного рода прикладных систем. В этой главе мы дадим краткое описание архитектуры программ общего интерфейса логической системы и архитектуры ряда наиболее важных стандартных вспомогательных блоков, используемых в этих программах.

Начиная с этой главы, рекомендуется при чтении следить за текстами соответствующих программ, отображенных на экране компьютера. Основная часть материала имеет характер развернутых комментариев к этим программам.

13.1 Главное меню

13.1.1 Начало программы общего интерфейса

При запуске логической системы интерпретатор ЛОСа (программа LOGSYST.EXE) создает фиктивную задачу на исследование, имеющую единственную посылку, состоящую из однобуквенного терма "вход", и запускает процесс сканирования этой задачи. При сканировании сначала происходит обращение к программе символа "исследовать" - оно не сопровождается срабатываниями каких-либо приемов, и затем осуществляется обращение к программе логического символа "вход". Эта программа и реализует общий интерфейс логической системы. Проследим первые шаги, выполняемые реализуемой в сканировании задачи программой символа "вход". Для удобства рекомендуем войти в просмотр корневого фрагмента данной программы и перемещаться по ней согласно дальнейшему тексту.

Собственно корневой фрагмент программы "вход" обслуживает справочник "арность"; при сканировании задачи он игнорируется, и по переходу "иначе 1" попадаем в фрагмент, начинающийся с оператора "решить".

Этот фрагмент прежде всего проверяет наличие первого информационного блока (файл I0000.lsi), хранящего необходимые для прорисовки элементов интерфейса ресурсы (шрифт, текстовые заготовки, битмэпы). Проверка выполняется оператором "не(файл(проверка(1)))". Если такого блока нет, то сразу же происходит выход из

логической системы (оператор "ответ(отказ)"). Заметим, что выход из системы реализуется при выдаче ответа на исходную фиктивную задачу, т.е., в частности, при обращении к оператору "ответ(...)" в любой точке программы "вход". В логической системе не предусмотрена специальная кнопка для выхода - он обеспечивается средствами, предоставляемыми операционной системой. Помимо этого, выход реализуется лишь в аварийных ситуациях и означает порчу либо некомплектность файлов системы.

Из текущего фрагмента программы "вход" по переходу "ветвь 2" попадаем в фрагмент, создающий дополнительный драйвер клавиатуры. Этот драйвер может понадобиться при работе с клавиатурой, имеющей только латинские символы. Тогда нажатие "Ctr-F9" в режиме текстового редактора позволяет использовать латинские символы как аналоги символов кириллицы (g - г, u - у, и т.п.). Выключение драйвера - повторное нажатие данной клавиши. Роль драйвера играет комментарий (текстредактор AB) к посылкам исходной задачи, у которого B - список кодов клавиатуры (логических символов), соответствующих символам кириллицы; A - список кодов для соответствующих символов латиницы.

Далее по переходу "ветвь 1" попадаем в фрагмент, начинающийся с операторов "файл(проверка 2) файл(актив 2)". Этот фрагмент анализирует второй информационный блок, в котором содержится задачник системы. Из корневого указателя-каталога данного блока по метке "метка" имеется переход к логическому терминалу, хранящему терм "задачи(N)". Число N является индикатором цикла серийного решения задач некоторого раздела задачника. Если имеет место установка на проведение такого цикла, то при запуске логической системы не происходит выход в главное меню, а сразу возобновляется цикл решения задач (начиная с задачи, следующей после той, на которой была прервана работа решателя). Рассматриваемый фрагмент инициирует такое продолжение прерванного цикла - вводит специальные комментарии к посылкам исходной задачи, в зависимости от N. Более подробно об индикаторе N - см. справочную информацию к символу "вид", оглавление типа "задачи". Автоматическое возобновление циклов обработки обучающего материала (задачи, теоремы, приемы) введено для удобства восстановления нормального состояния массивов интерпретатора после возможных ошибок, обусловленных длительностью цикла обработки. Нажатие *Esc* в отладчике ЛОСа (например, при чрезмерно затянувшемся решении задачи) вызывает внутренний перезапуск системы с расчисткой всех рабочих массивов, после чего цикл решения продолжается в нормальном режиме.

Далее через переход "ветвь 1" попадаем в фрагмент, начинающийся с операторов "файл(проверка 6) файл(актив 6)". Он аналогичен предыдущему фрагменту - инициирует возобновление цикла серийной обработки раздела базы теорем (например, при тестировании логического вывода). Еще один переход по метке "ветвь 1" приводит снова к аналогичному фрагменту - на этот раз для возобновления цикла обработки раздела базы приемов.

Наконец, последний перед прорисовкой главного меню переход по метке "ветвь 1". Он приводит к фрагменту, начинающемуся с оператора "файл(актив 1)". Прежде всего, здесь реализуются операторы "указатель(список набор(0 3)видео х6) файл(видео начало(х6))". Они обеспечивают загрузку в оперативную память интерпретатора битмэпов используемого решателем шрифта - лишь после этого будут работать операторы прорисовки текстов.

Далее идет оператор "повторение" - к нему происходят откаты при повторных прорисовках главного меню. Наконец, следующий оператор вида "или(A и(Меню(0) Меню(актив)главноеменю(0 0)равно(х7 0)))" выполняет прорисовку главного меню,

если не выполнено условие A (упоминавшиеся выше режимы возобновления серийной обработки). При прорисовке главного меню в верхней части экрана убирается полоска с меню операционной системы. Присвоение нуля переменной x7 в данном операторе - фиктивное; оно делает его перечисляющим и позволяет сохранить старые величины откатов к перерисовке меню (в ранних версиях этот оператор являлся существенным образом перечисляющим). Заметим, что какие-бы то ни было изменения программы "вход" следует выполнять с особой осторожностью, так как после них может оказаться ЗАБЛОКИРОВАНА ВОЗМОЖНОСТЬ ПОВТОРНОГО ВХОДА В ЕЕ РЕДАКТИРОВАНИЕ, и система окажется полностью выведенной из строя (для продолжения работы придется излекать резервную копию из директории SLCOPY). Напомним, что программа символа "вход" - единственная программа ЛОСа, в которой заблокирован автоматический сдвиг номеров программных переменных.

После оператора прорисовки главного меню идет оператор "ветвь 1", а за ним - группа операторов, выполняющих проверку наличия указателя на немедленный переход от главного меню к задачнику либо к справочнику по системе. Такой указатель хранится в логическом терминале первого информационного блока, достижимом от корневого указателя - списка по метке "установка". Если содержимое этого терминала - символ 2, то инициируется переход к задачнику; если символ 3, то - к справочнику по системе. Заметим, что в начале указанной группы располагается оператор "трассировка(вход)", который истинен только при первом запуске системы, а при последующих ее внутренних перезапусках с помощью оператора "трассировка(0)" (без выхода в операционную систему) - ложен. Если бы этого оператора не было, то выход в главное меню при последующей работе с системой оказался вообще невозможен. Для перехода к задачнику либо справочнику создается комментарий "автоклавиатура ..."к посылкам исходной задачи, содержащий коды необходимых для перехода клавиш.

Следуя по "ветвь 1", попадаем в подфрагмент, начинающийся с оператора "повторение". К этому оператору происходят откаты при повторных обращениях к главному меню. Далее следует большой оператор "альтернатива(...)", реализующий обращение к главному меню для ввода кода клавиатуры x9. Он представляет собой цепочку вложенных друг в друга операторов "альтернатива", которые пытаются определить значение x9 до фактического обращения к вводу с клавиатуры либо мыши, если это можно сделать согласно различным комментариям к посылкам текущей задачи. Например, x9 будет автоматически определяться при наличии одного из упоминавшихся выше циклов серийной обработки. Часто используется для автоматического входа при откате в нужный пункт главного меню комментарий "новпозиция A" к посылкам исходной задачи, у которого A - символ клавиатуры, обеспечивающий вход в этот пункт. Лишь последний из вложенных друг в друга операторов "альтернатива" обращается к вводу x9 через главное меню (см. подоператор "главноеменю(1 x9)").

После получения значения x9 - переход через оператор "ветвь 1" к началу цепочки фрагментов, осуществляющих обработку различных пунктов главного меню. Перечислим эти пункты. Заметим, что к большинству из них можно перейти непосредственно из подраздела "Общий интерфейс" оглавления программ.

13.1.2 Обращение к оглавлению приемов

Первый из пунктов цепочки обработки команды главного меню (x9 = "внешнийквантор") - вход в оглавление приемов ГЕНОЛОГа. Здесь сначала делается активным 8-й информационный блок, содержащий оглавление базы приемов и описания приемов

на ГЕНОЛОГе (подробнее о его строении будет рассказано в разделах, посвященных ГЕНОЛОГу). Затем вводится комментарий (трассировка ...), который делает возможным просмотр отладчиком ЛОСа термов в стандартной математической записи и может оказаться полезным при отладке процедур редактора ГЕНОЛОГа. Оператор "равно(x10 0)"резервирует переменную x10, которой далее будет присвоена ссылка на корневой указатель - список оглавления приемов. Это присвоение осуществляется в конце фрагмента, после оператора "ветвь 2", к которому произойдет откат после определения значения x10.

Фрагмент, к которому переходим через "ветвь 2", начинается с оператора "повторение". К этому оператору будут происходить откаты для повторных входов в оглавление приемов после обработки очередного выбранного концевого пункта этого оглавления. Далее идут оператор, удаляющий ранее имевшийся комментарий (вид ...)к посылкам исходной задачи (если он был), и оператор, вводящий новый такой комментарий - (вид прием), указывающий, что происходит работа с оглавлением приемов. Удаляется также комментарий (прием ...) - ссылка на последний просматривавшийся до этого прием базы приемов. Наконец, оператор "или(и(равно(x11 0)...)и(оглавление(x10 x11)замена(x11 конец(x11))))" обеспечивает собственно обращение к оглавлению базы приемов и присваивает переменной x11 ссылку на выбранный концевой логический терминал этого оглавления. Заметим, что первый операнд оператора "или(...)"осуществляет присвоение значения x11 без обращения к оглавлению - если нужно вернуться к известному концевому пункту, на который ссылается комментарий (контрольпрограммы ...).

При работе с оглавлением имеются две возможности - либо выйти из него без выбора концевого пункта, либо выбрать концевой пункт. В первом случае оператор "оглавление(...)" ложен, и согласно переходу "иначе 1"попадаем в ветвь программы, обрабатывающую обрыв работы с оглавлением. Если в комментариях к посылкам исходной задачи к этому моменту появилось указание на переход к некоторому заданному пункту главного меню, то снова выполняется переход по "иначе 1", и далее выполняется оператор "сброс(3)" для отката к определению значения x9; если такого указания нет, то происходит внутренний перезапуск и выход в главное меню.

Если в оглавлении приемов был выбран концевой пункт, то далее происходит обращение к редактору ГЕНОЛОГа - оператору "блокприемов(x11)". После выхода из него проверяется наличие комментариев, определяющих переключения на другие пункты главного меню либо внутренний перезапуск. Если таких комментариев нет, то происходит откат к упоминавшемуся выше оператору "повторение" и повторное обращение к оглавлению. Заметим, что повторный вход в оглавление осуществляется на тот концевой пункт, который был выбран в последний раз, и при работе создается впечатление, что обращение к редактору ГЕНОЛОГа происходит непосредственно из процедуры "оглавление".

13.1.3 Обращение к редактору ЛОСа

Следующий пункт цепочки обработки кода x9 начинается с оператора "равно(x9 анализатор)". Это - начало ветви, реализующей редактор программ ЛОСа. Попасть в него можно также через оглавление программ - подпункт "Вход в редактор ЛОСа для заданного лог.символа" пункта "Общий интерфейс". В начале фрагмента расположен оператор "файл(актив 1)", активизирующий первый информационный блок. После оператора "равно(x11 0)" (резервирующего пока не используемую переменную x11) идет оператор "повторение", к которому будут происходить откаты при

повторных запросах на ввод того логического символа, программу которого нужно редактировать. Далее оператор "видео(0)"расчищает буфер текстов перед обращением к текстовому редактору, а оператор "или(существует(...)существует(...)существует(x12 текстредактор(0 ... x12)))" - реализует обращение к текстовому редактору, с помощью которого будет введено название логического символа. Заметим, что в этой дизъюнкции первые два оператора проверяют наличие комментариев, непосредственно указывающих на нужный символ; если такие комментарии есть, то ручной ввод символа отменяется.

Если при наборе символа была нажата клавиша "Esc", то оператор ввода символа оказывается ложным. Тогда переходим через "иначе 2" к подфрагменту, завершающему исключение из программы ранее отключенных ветвей (ссылки на них извлекаются из комментария (копияветви ...)), и осуществляющему внутренний перезапуск с возвращением в главное меню.

Если символ был введен, то выполняется оператор "равно(x12 0)" (резервирующий пока не используемую переменную x12), и далее - оператор "альтернатива(A_1A_2 альтернатива(B_1B_2 и(кодтекста(3 x13 x14)равно(x14 0))))". Здесь снова проверяется наличие комментариев, явно указывающих на тот символ, программа которого будет просматриваться, и если эти комментарии есть, то символ присваивается переменной x13 (см. операторы A_2, B_2). Если же таких комментариев нет, то выполняется оператор "кодтекста", считывающий название символа из буфера текстов, где оно было сформировано текстовым редактором.

Далее название символа прорисовывается в верхней части экрана (операторы "видео(прямоугольник ...)" и "существует(x15 видео(логсимвол ...))").

В конце фрагмента происходит присвоение переменным x15,x16 сначала некоторых фиктивных значений (в качестве таковых выступают вхождения левого края исходной задачи), а затем - с помощью оператора "прогфайл(логсимвол ...)" - ссылки на корневой фрагмент программы логического символа x13 (если такой фрагмент к данному моменту уже создан). Предварительно проверяется наличие комментария "альтернатива" к посылкам исходной задачи; если он есть, то ссылка берется по блоку программ, хранящемуся в версии решателя из поддиректории ALT. Это позволяет просматривать из текущей версии программы альтернативную версию, отбирать нужные ее ветви и переносить в текущую версию. После определения ссылки (x15,x16) - переходим к следующему фрагменту через оператор "ветвь 3".

Этот фрагмент начинается с оператора "замена(x14 пустоеслово)", после которого x14 используется как буфер ссылок на отключенные (но пока не удаленные) ветви программы. Далее идет оператор "замена(x12 пустоеслово)". Переменная x12 начиная с этого момента используется в качестве буфера, определяющего путь к текущему просматриваемому фрагменту программы символа x13 от корня этой программы. Буфер будет представлять собой набор троек ($A_1A_2A_3$), причем у очередной тройки (A_1A_2) - ссылка на фрагмент в блоке программ, A_3 - номер перехода от него к подфрагменту. Последняя тройка соответствует корню программы символа x13, первая - фрагменту, предшествующему текущему фрагменту.

Следующий оператор "замена(x11 1)"инициализирует переменную x11, которая будет далее хранить номер выделенного оператора перехода текущего фрагмента.

Далее идет оператор "повторение", к которому будут происходить откаты при перерисовках текущего фрагмента программы. После него идет оператор "сборкаменю(набор(5 1))", вводящий в верхней части экрана меню системы, сопровождающее обычный режим просмотра программы. Если вход в редактор программ произошел согласно комментарию (цепьвхождений ...), непосредственно указывающему на тот

фрагмент, к просмотру которого нужно перейти, то следующий оператор "длялюбого(...)"переустанавливает значения x12, x15, x16 на просмотр нужного фрагмента согласно данному комментарию и затем удаляет его. Далее оператор "альтернатива(равно(x15 левыйкрай(x1))равно(x17 пустоеслово)...)" считывает текущий фрагмент программы и присваивает его переменной x17 (если программа символа x13 не была создана, то x17 становится равно пустому набору).

Перед прорисовкой фрагмента программы происходит коррекция этого фрагмента: каждый оператор "замена(N t)" заменяется, для удобства чтения программы, на "замена(xN t)". Ее выполняют операторы, расположенные после оператора "ветвь 1". После коррекции - переход к новому фрагменту через "ветвь 1".

Оператор "или(существует(x18 и(...))видео(прямоугольник ...))" проверяет отсутствие комментариев, указывающих на ненужность прорисовки фрагмента программы; если их нет, то он расчищает ту часть экрана, которая расположена под названием текущего логического символа. Текущая позиция для последующей прорисовки устанавливается на левый край второй сверху текстовой полосы (оператор "видео(позиция 0 девятнадцать)"). Затем инициализируются используемые в цикле прорисовки данные. Прежде всего, это накопитель x18 пятерок (столбец - строка - вхождение оператора в фрагмент программы - исходная позиция в буфере текстов - заключительная позиция в буфере текстов), характеризующих размещение на экране операторов перехода "ветвь ...", "иначе ...". Он нужен для перекраски фона текущего выбранного оператора перехода. Первые два элемента пятерки указывают координаты начала оператора на экране; последние два - размещение его в буфере текстов. Инициализируются также x19 - счетчик прорисованных операторов перехода и x20 - номер первой свободной позиции в буфере текстов.

В начале следующего фрагмента расположен оператор "позиция(x21 x17)", перечисляющий для прорисовки вхождения операторов в текущий фрагмент программы. После него идут два оператора, проверяющие необходимость прорисовки фрагмента. Если они истинны, то выполняется оператор, распознающий операторы перехода по их заголовкам "ветвь","иначе". Если оператор не является оператором перехода, то фрагмент по "иначе 2" выполняет его прорисовку с текущей позиции, пропуск одной пустой позиции и коррекцию указателя x20 на первую незанятую позицию буфера текстов. В противном случае происходят пополнение накопителя x18 очередной пятеркой, прорисовка оператора перехода (заголовок оператора и через пробел - номер перехода x19), увеличение счетчика x19 на единицу и коррекция указателя x20. Если номер x11 выделенного оператора перехода равен x19, то фон при прорисовке выбирается желтый.

После прорисовки всех операторов фрагмента программы - переход по "иначе 1" к фрагменту, присваивающему переменной x19 вхождение в накопитель x18 той пятерки, которая соответствует текущему выделенному оператору перехода. Оттуда, по "ветвь 1", переход к фрагменту, начинающемуся с операторов "повторение", "прием(1)", "автоменю(x21)". Здесь начинается обработчик команд редактора ЛОСа; подробнее о программах этого редактора будет рассказано ниже в отдельной главе.

13.1.4 Обращение к оглавлению задачника

Следующий пункт цепочки обработки кода x9 - вход в оглавление задачника. Альтернативный способ найти этот пункт - через оглавление программ (Общий интерфейс - Обращение к оглавлению задачника). После оператора "равно(x9 подборнеизвест-

ных)" расположены операторы, запоминающие в x10 номер текущего активного информационного блока и при отсутствии задачника (второй информационный блок) - инициализирующие его. Далее - переход по "ветвь 2" к фрагменту, начинающемуся с оператора "файл(актив 2)", активизирующего информационный блок задачника.

Здесь переменной x11 присваивается исходная задача; если из корневого указателя-каталога задачника не было перехода по метке "оглавление" к некоторому указателю-списку x12, то такой переход вводится. Находится переход от x12 по метке "задачи" (если его не было, то он тоже вводится). В результате x13 - набор, содержащий ссылку на корень оглавления задачника. Вводится комментарий (вид задачи) к посылкам исходной задачи, указывающий, что начата работа с оглавлением задачника. Далее идет оператор "повторение", к которому будут происходить откаты при повторных входах в это оглавление после рассмотрения его концевых пунктов (то есть отдельных задач). Сразу же за ним - оператор "видео(конец)", сбрасывающий ранее введенный массив текстов. Этот массив накапливает текстовые заготовки, использкемые при прорисовке на экране задач, и его следует периодически сбрасывать во избежание переполнения.

Обращение к оглавлению задачника реализуется оператором "оглавление(начало(x13)x14)". Если при работе с оглавлением концевой пункт не был выбран, то оператор оказывается ложным; в этом случае переходим по "иначе 3". Если был введен комментарий (новпозиция ...), определяющий переключение на новый пункт главного меню, то предпринимается откат к точке, обрабатывающей обращения к этому меню, иначе осуществляется внутренний перезапуск с выходом в главное меню.

Если концевой пункт оглавления был выбран (либо определился автоматически согласно комментариям к посылкам исходной задачи), то далее проверяется наличие комментария "задачи", указывающего на наличие цикла решения задач из некоторого раздела задачника - см. оператор "входит(задачи комментариипосылок(x11))". После этого оператора предпринимается проверка того, что в данном цикле вообще следует предпринимать попытку решения задачи, соответствующей выбранному концевому пункту оглавления. Если это не так, то происходит откат к оператору "повторение" и очередной вход в оглавление задачника. Заметим, что в циклах автоматического решения серий задач поиск очередной задачи выполняется непосредственно оператором "оглавление", который далее снова выводит к оператору "входит(задачи комментариипосылок(x11))".

Проверка целесообразности решения очередной задачи в цикле предпринимается следующим образом. Если имеется комментарий "идент" к посылкам исходной задачи, то логический терминал, достижимый по метке "не" из корневого каталога задачника, содержит термы "набор($S_1 \dots S_m$)", указывающие на такие наборы логических символов $S_1, \dots S_m$, которые обязательно должны встретиться в процессе решения задачи - иначе заведомо не срабатает некоторый прием, и повторное решение задачи не интересно. Чтобы определить по текущей задаче, какие именно логические символы возникали ранее при ее решении, используется оператор "архивзадачи(x17 x18 x21)". После обращения к нему и происходит сравнение наборов $S_1, \dots, S_m$ с фактически имевшимися символами. Если задача проходит по указанному признаку, то далее выполняется контроль (см "ветвь 5") того, что ранее на нее был получен ответ. Для задач, не решенных при предыдущем обращении к ним, попытка решения в обычном серийном запуске не предпринимается (возможен запуск с отменой такого ограничения - об его отсутствии свидетельствует комментарий "префикснаярекурсия").

Если не было отмены рассмотрения задачи в цикле решения, то предпринимается

переход по "ветвь 4". При наличии комментария "задачи" (идет серийное решение) формируется комментарий (автоклавиатура (частичныйответ)), эмулирующий нажатие клавиши "о" при входе в рассмотрение текущей задачи и инициирующий ее решение. Иначе - переход к следующему фрагменту по "ветвь 1".

Здесь происходит обращение к интерфейсу задачника, реализуемому оператором "списокзадач(x16 конец(x17) конец(x15))". Этот интерфейс позволяет просмотреть условия всех задач текущего меню оглавления задачника, расположенных подряд и отделенных друг от друга горизонтальными линиями. В верхней части экрана вначале расположена линия, под которой находится текст условия выбранной в меню задачи. Более подробно программы интерфейса задачника описываются ниже в отдельной главе.

13.1.5 Операции со словарем

Операции со словарем сгруппированы в ветвь программы, начинающуюся с оператора "или(равно(x9 группировка) ... равно(x9 посылки))". Сюда относятся операции: ввода нового символа (x9 = "группировка"); изменения названия символа (x9 = "учетсимволов"); удаления символа (x9 = "конъюнктоперанд"); определения номера символа по его названию (x9 = "усмчисло") и определения названия символа по его номеру (x9 = "посылки"). Операторы "файл(актив 1)" и "видео(0)" делают активным первый информационный блок (из него извлекаются текстовые заготовки для окон диалога) и расчищают буфер текстов. Затем происходит расчистка экрана. Если операция отлична от определения названия символа по его номеру, то из текстового терминала первого информационного блока, достижимого по меткам "комментарии", "преобразование", извлекается текст "Логический символ:", который и прорисовывается в верхней части экрана. Расчищается буфер текстов; оператор "видео(точка x17 x18)" определяет позицию (столбец x17 - строка x18) справа от прорисованного текста, и начиная с этой позиции создается окно текстового редактора (оператор "текстредактор(0 x17 ... x19)"). При отказе от набора символа (переход "иначе 3") происходит откат к главному меню. Иначе - оператор "видео(прямоугольник 0 0 прогртерм наборчленов 0)" расширяет рамку выдачи текстов до всего экрана, не изменяя содержимого экрана. Дальнейшие действия зависят от того, какая операция была выбрана:

а) Ввод названия нового символа (x9 = "группировка"). Оператор "словарь(запись x20)" вводит новый символ с названием, хранящимся в начале буфера текстов (оно только что создано текстовым редактором). Если такое название уже имелось, то переход через "иначе 5" к фрагменту, прорисовывающему текст о том, что название ранее использовалось.

б) Удаление символа с данным названием (x9 = "конъюнктоперанд") Оператор "словарь(исключение)" удаляет название символа.

в) Определение номера символа с заданным названием (x9 = "усмчисло"). Оператор "словарь(проверка x20)" определяет номер x20 символа (в формате символьного числа). Далее определяется десятичное число x21, равное x20; прорисовывается символ "№", находится представление x22 числа x21 в виде терма, и этот терм прорисовывается. После прорисовки - обращение к операторам "клавиатура(x25)","обрыв", обеспечивающее возвращение к главному меню после нажатия любой клавиши.

г) Изменение названия логического символа (остаточный случай для x9). Прежде всего проверяется наличие символа с изменяемым названием и определяется его номер x20. Затем из первого информационного блока извлекается и прорисовывается

текстовая заготовка "заменяется на:". После изменяемого названия в буфер текстов заносится 0, отделяющий его от нового названия. Далее - ограничивается рамка для набора текстовым редактором нового названия; оператор "повторение" обозначает точку отката к повторным входам в текстовый редактор, если первоначально было введено уже использованное заменяющее название, и происходит обращение к текстовому редактору. При отказе от ввода символа - возвращение в главное меню (тогда исходное название сохраняется). Наконец, оператор "словарь(изменение)" осуществляет изменение названия. Если новое название уже использовалось (переход по "иначе 2"), то восстанавливается старое название, прорисовывается указатель на то, что новое название уже занято, и происходит повторный вход в текстовый редактор.

В приведенном выше списке не рассмотрена операция определения названия символа по его номеру. Она соответствует случаю x9 = "посылки"; в исходном фрагменте ветви обработки операций со словарем переход по "иначе 2" (после оператора "не(равно(x9 посылки))") ведет к фрагментам, выполняющим данную операцию. Здесь происходит прорисовка символа "№", обращение к текстовому редактору для ввода номера, определение символа по его номеру и прорисовка символа. По нажатию любой клавиши после этого - возвращение в главное меню.

13.1.6 Уплотнение и сохранение файлов

При x9 = "внутрвывод" происходит уплотнение измененных за период после предыдущего уплотнения файлов информационных и программного блоков. Вход в соответствующий фрагмент программы возможен через оглавление программ - раздел "Общий интерфейс - Уплотнение инф.блоков и прогр.блока". Оператор "файл(выписка x10)" присваивает переменной x10 набор номеров тех информационных блоков, для которых, возможно, потребуется уплотнение. Эти блоки поочередно активизируются, и оператор "файл(нормарксинус)"выполняет их уплотнение. Если при уплотнении обнаруживается ошибка в структуре файла (ложный оператор "файл(
нормарксинус)"), то выдается ответ на исходную задачу, то есть происходит выход из решателя в операционную систему. После обработки информационных блоков - обращение к оператору "прогфайл(нормарксинус)", уплотняющему программный блок. Если обнаруживается ошибка в программном блоке, то последний оператор истинен, и после его выполнения происходит выход из решателя.

При x9 = "нижняяоценка" происходит проверка корректности структуры информационных блоков и программного блока. Если обнаруживается ошибка (ложный оператор "файл(выход)"), то происходит выход из решателя в операционную систему. Иначе предпринимается сохранение всех файлов информационных блоков и программного блока (только после успешно завершенной проверки корректности) в резервной копии системы, хранящейся в директории $C:\backslash SLCOPY$.

13.1.7 Обращение к оглавлению программ

При x9 = "минимум" происходит обращение к оглавлению программ. Для перехода в соответствующий фрагмент можно использовать пункт "Общий интерфейс - Вход в оглавление программ" из оглавления программ. Прежде всего проверяется наличие 9-го информационного блока, содержащего оглавление программ, и если его не было, то он создается. После этого - переход по "ветвь 2".

Здесь активизируется 9-й информационный блок и вводится комментарий (трассировка ...) к посылкам исходной задачи, который может понадобиться при про-

смотре термов в стандартной записи из отладчика ЛОСа. Далее переменной x10 присваивается 0, после чего предпринимается поиск корневого указателя оглавления программ (либо создание такого корня, если его не было), и переменной x10 переприсваивается ссылка на найденный корень. Затем - переход по "ветвь 1".

Новый фрагмент начинается с оператора "повторение", к которому будут происходить откаты при повторных входах в оглавление программ. Затем вводится комментарий (вид программа) к посылкам исходной задачи, указывающий на тип оглавления, с которым будет происходить работа (старая ссылка на тип оглавления перед этим удаляется). Далее - обращение к оператору "оглавление(x10 x11)", осуществляющему собственно просмотр оглавления программ. Если был выбран некоторый концевой пункт оглавления, то x11 - пара ссылок: на текстовый терминал выбранного пункта, и на его логический терминал. Операторы "равно(x12 конец(x11)) файл(терм x12 x13) равно(x14 начало(x13))" определяют терм x14 вида "программа(AN)", хранящийся в логическом терминале. У него A - логический символ, N - номер, ссылающиеся на оператор "прием(N)" в программе символа A, к которому относится выбранный пункт оглавления. Вводятся комментарии (прогрблок A) и (подфрагмент A прием(N) x12) к посылкам исходной задачи, которые будут использоваться после отката к главному меню для последующего выхода на нужный фрагмент программы и выделения в нем оператора "прием(N)". В верхней части экрана (полоска высотой в 19 пикселей) сразу же прорисовывается название символа A, которое сохранится при прорисовке найденного фрагмента программы. Далее вводится комментарий (автоклавиатура (элементы)) к посылкам исходной задачи. Он будет эмулировать после отката нажатие клавиши F8 для поиска оператора "прием(N)" в программе символа A, и предпринимается откат к главному меню. Если теперь вернуться к большому оператору "альтернатива(... главноеменю(...))", то можно заметить внутри него присвоение "равно(x9 вариант(существует(x10 ключ(
комментариипосылок(x1)прогрблок x10))анализатор внешнийквантор))", усматривающий комментарий (прогрблок A) и выдающий значение x9 = "анализатор" для входа в редактор программ ЛОСа. Таким образом выполняется первый шаг цепочки, приводящей после отката к просмотру нужного фрагмента программы. Дальнейшие шаги легко проследить по программе редактора программ. Комментарий (подфрагмент ...) сохраняется вплоть до возвращения в оглавление программ из просмотра фрагментов, чтобы распознать в нажатии "End" команду на такое возвращение.

13.1.8 Редакция шрифта

По x9 = "схемаидентификации" происходит вход в интерфейс просмотра битмэпов используемого решателем шрифта и изменения этих битмэпов. В данную точку программы можно попасть из пункта "Общий интерфейс - Просмотр и изменение шрифта" оглавления программ. Битмэпы для используемого решателем шрифта хранятся в первом информационном блоке, в достижимом из корня этого блока по метке "видео" текстовом терминале. При запуске решателя они извлекаются из этого файла и переносятся в специальный массив оперативной памяти, который далее называем видеокассой. При изменении битмэпа для отдельного символа затрагивается только видеокасса; чтобы перенести эти изменения в файл, нужно выполнить операцию сохранения шрифта.

Программа интерфейса работы со шрифтом начинается с оператора "повторение", к которому будут происходить откаты после выполнения очередной операции. Далее расположены два оператора прорисовки прямоугольников: сначала - общий

черный фон экрана, затем - белый фон для зоны прорисовки битмэпов. Затем идет обращение к клавиатуре для запроса команды x12 на операцию со шрифтами. Перечислим пункты обработчика этих команд:

1. Если x12 = "Плюс"(клавиша Esc), то происходит возвращение в главное меню.

2. Если x12 = "внешдизъюнкция" (клавиша Insert), то содержимое видеокассы сохраняется в первом информационном блоке. Если изменить битмэпы и не выполнить эту операцию, то при перезапуске решателя изменения пропадут. После сохранения видеокассы предпринимаются уплотнение первого информационного блока и внутренний перезапуск, с возвращением в главное меню.

3. Если x12 = "стандменьше" (клавиша F1), то осуществляется обращение к текстовому редактору для пробной прорисовки измененных битмэпов символов.

4. Если x12 = "попыткапараметризации" (клавиша F2), то реализуется интерфейс ввода номера в видеокассе того символа, для которого нужно просмотреть либо изменить битмэп. Номер символа изменяется от 0 до 255. Этот интерфейс необходим из-за того, что не для всех символов существует соответствующая клавиша. К полученному номеру прибавляется 1, он переводится в формат символьного номера x17, и далее переменной x12 присваивается значение x17. После такого переприсвоения - переход к описываемому в следующем пункте "общему случаю" обработки команды.

5. Все неподпадающие под перечисленные выше случаи значения x12 рассматриваются как символьный номер, соответствующий некоторой букве из видеокассы. Логический символ, представляющий этот номер, прорисовывается в верхней левой части экрана. Если сивмольный номер больше 256, то после оператора "менее(x12 определение)" - переход по "иначе 1", обращение к оператору "клавиатура(x13)", и по нажатии любой клавиши - откат к запросу очередной команды. Если же номер попадает в диапазон видеокассы, то прежде всего оператор "видео(развертка x12 x13)"определяет битмэп символа в виде набора x13 длины 19, состоящего из 8 - элементных наборов из нулей и единиц (логических символов). Затем происходит прорисовка этого битмэпа в виде сетки (синие квадратики - 0, черные - 1). Одновременно слева на черном фоне прорисовывается белый "негатив" символа, соответствующего битмэпу. Этот негатив изменяется при любых изменениях битмэпа и позволяет контролировать качество изображения. Заметим, что после прорисовки сетки битмэпа выполняется оператор "повторение", к которому будут происходить откаты при элементарных изменениях этой сетки. Далее прорисовывается желтый квадратик - курсор работы с сеткой, и текущее состояние набора x13 переносится в видеокассу (оператор "видео(бланк x12 x13)"). В конце фрагмента программы находится обращение к оператору "клавиатура" - запрос на текущую команду x18 работы с сеткой. Здесь имеются следующие команды:

   а) x18 = "префиксныйфильтр" (клавиша 0) - текущий квадратик сетки устанавливается на 0;

   б) x18 = "унисборка" (клавиша 1) - текущий квадратик сетки устанавливается на 1;

в) x18 = "циклвариантов", либо x18 = "внешконъюнкция", либо x18 = "внешсумма", либо x18 = "подстановка" - перемещения курсора с помощью четырех клавиш курсора.

г) x18 = "тринадцать" (клавиша Enter) - завершение обработки сетки битмэпа.

13.1.9 Вход в просмотр и редактирование информационных блоков

Интерфейс просмотра и редактирования информационных блоков является чисто техническим; работа с основными содержащимися в информационных блоках объектами, такими, как база приемов, задачник, оглавление программ и др., осуществляется через их специальные интерфейсы. Описываемый здесь интерфейс оказывается необходим в аварийных ситуациях - для восстановления информационного блока после поломки, а также для создания ресурсов различного рода (битмэпы, текстовые фрагменты и т.п.).

Вход в просмотр информационного блока требует указания номера этого блока. Поэтому в данном случае оператор "главноеменю(...)"выдает в качестве значения x9 не логический символ "новыесвязки", соответствующий нажатой клавише "и", а терм "новыесвязки(i)", где i - десятичный номер информационного блока, введенный с помощью текстового редактора. Прежде всего определяется символьный номер x11 рассматриваемого блока; предпринимаются проверка наличия этого блока, его активизация и инициализация стэка x12. Этот стэк будет хранить набор $A_1 \dots A_n$ троек (ссылка на объект информационного блока - метка перехода к нему от предыдущего объекта - номер перехода от него к следующему объекту; нумерация берется относительно полного списка переходов от данного объекта). Начало стэка A_1 соответствует текущему просматриваемому объекту блока; A_n соответствует корню блока и имеет вид $((03), 0, 1)$. Если блок не был создан, то он при просмотре не создается и x12 остается равным 0. После инициализации x12 происходит переход по "ветвь 2".

Переменной x13, которая впоследствии будет хранить номер текущего выделенного перехода в списке переходов текущего просматриваемого указателя - списка, присваивается значение 1. Далее идет оператор "повторение", к которому будут происходить откаты при прорисовках очередного объекта информационного блока. После него предпринимается расчистка экрана; переменной x14 присваивается значение 0 (в случае указателя - списка ей будет переприсвоена некоторая структура данных, используемая для работы с этим указателем). Если x12 не равно 0 (т.е. блок существует), то прежде всего определяется тип x15 текущего объекта блока, ссылка на который имеется в начале первой тройки из x12. Это выполняет оператор "указатель(тип начало(начало(x12))x15)". В верхней левой части экрана прорисовывается название типа x15 (если x15 = "логсимвол", что соответствует случаю указателя-каталога, то вместо x15 прорисовывается слово "каталог"). Если текущий объект - указатель-каталог, то на этом его прорисовка завершается; если он представляет собой текстовый либо логический терминал, то перед завершением прорисовки переменной x14 переприсваивается единица.

Если текущий объект - указатель-список, то прежде всего находится список x17 всех его меток. Переменной x14 переприсваивается символ "пустоеслово" она становится накопителем шестерок для прорисовки указателя-списка: (столбец, с которого начинается прорисовка номера очередного перехода по метке - строка - исходная позиция в буфере текстов, где хранится этот номер - последняя позиция в буфере

текстов - метка - ссылка на объект, к которому происходит переход по метке). Выполняется расчистка буфера текстов; переменной x18 - указателю на первую свободную ячейку буфера текстов - присваивается 0. Далее просматриваются метки из списка x17. Каждая такая метка прорисовывается оператором "альтернатива(...)", и после нее прорисовываются номера переходов по данной метке к новым объектам. Нумерация в каждом случае начинается с 1 (x21 - текущий номер); после прорисовки очередного номера происходит пополнение накопителя x14 очередной шестеркой.

После прорисовки текущего объекта - переход по оператору "ветвь 1", расположенному после "равно(x14 0)". Здесь происходит выделение текущего перехода в случае указателя - списка: фон номера этого перехода делается красным. Текущим считается переход, номер которого равен x13. Переменной x15 присваивается вхождение в x14 шестерки для текущего перехода. Далее - переход по "ветвь 1".

В верхней части экрана прорисовывается меню для просмотра и редактирования информационных блоков. Затем выполняется оператор "повторение", к которому будут происходить откаты при выполнении команд, и предпринимается обращение к оператору "автоменю(x16)" для ввода очередной команды. Перечислим команды данного интерфейса:

а) x16 = "Плюс" (клавиша Esc) - завершение работы с блоком и возвращение в главное меню.

б) x16 = "внешсумма" (клавиша "курсор вверх") - возвращение к внешему для текущего объекта указателю. Если указатель был корневым, то возвращение в главное меню, иначе x12 укорачивается на единицу, x13 становится равно концевому символу первой тройки из x12, и возвращение в режим запроса очередной команды.

в) x16 = "циклвариантов" либо x16 = "внешконъюнкция" (клавиши "курсор вправо" и "курсор влево"). В случае текущего указателя-списка (что проверяется оператором "не(логсимвол(x14))") предпринимается сдвиг вправо либо влево указателя текущего перехода. Корректируются значения x15,x13 и последний элемент первой тройки набора x12 (воспроизводящий значение x13).

г) x16 = "подстановка" (клавиша "курсор вниз"). В случае текущего указателя-списка реализуется выбранный в нем переход: в начало стэка x12 заносится информация о новом текущем объекте, x13 заменяется на 1, и откат к прорисовке объекта. В случае текстового терминала (x14 = 1) происходит прорисовка этого терминала. Для логического терминала нажатие клавиши "курсор вниз" недопустимо (его содержимое просматривается другой командой). Наконец, в случае указателя-каталога происходит запрос на ввод текстовым редактором логического символа x24. Затем реализуется переход из него по метке x24 - аналогично случаю указателя-списка.

д) x16 = "внешдизъюнкция" (клавиша Insert). Здесь осуществляется ввод нового перехода из текущего указателя-списка либо указателя-каталога, либо, в случае терминала, его изменение. Если информационный блок еще не был введен (x12 = 0), то он вводится, состоящим пока из единственного файла длины 0. Инициализируется нулем переменная x17, которой будет переприсвоена метка перехода из текущего указателя. Для указателя - списка (в случае истинности "или(не(логсимвол(x14))равно(x14 пустоеслово))") реализуется ввод текстовым редактором терма x19 - метки нового перехода, которая и переприсваивается переменной x17. Для указателя-каталога (переход по "иначе 3") аналогичным образом вводится метка - логический символ x25. После доопределения x17 - переход по "ветвь 2".

Здесь инициализируется нулем переменная x18, которой затем с помощью прорисовываемого в верхней части экрана меню типов объектов присваивается логический символ - код типа нового объекта. Далее рассматриваются варианты:

д1) x18 - один из символов "верхняяоценка", "подобл", "извлечениеварианта" (соответственно, типы "Цв.текст", "Ч.-б. текст", "Термы"). Тогда расчищаются экран и буфер текстов. Если текущий объект - текстовый терминал, то оператор "или(не(равно(x14 1))...)" выполняет прорисовку содержимого терминала. В случае логического терминала прорисовка его содержимого осуществляется в той части фрагмента программы, которая расположена после оператора "равно(x18 извлечениеварианта)". Далее - переход через "ветвь 5". Здесь осуществляется ввод текстовым редактором новой версии терминала (если он уже имелся, то редактирование происходит поверх старого текста) и сохранение этой версии (изменение старой версии терминала либо ввод нового терминала с переходом к нему по метке x17).

д2) x18 = "заменагруппы" (тип "Битмэп"). Предпринимается редактирование ранее введенного битмэпа либо ввод нового битмэпа и сохранение его в логическом терминале по метке x17. Размеры битмэпа определяются последними операндами оператора "битмэп(...)". Это - редко используемая возможность ввода битмэпов для специальных целей (например, для прорисовки шахматных фигур), и при применении ее может понадобиться предварительно изменить оператор "битмэп(...)"в программе, указав в нем нужные число строк и столбцов битмэпа.

д3) x18 = "легковидеть" (тип "Вставка"). Происходит ввод нового логического терминала, содержимое которого было извлечено ранее из другого логического терминала (см. ниже команду, сохраняющую в буфере содержимое текущего логического терминала для последующего копирования).

д4) x18 = "внутрвывод" (тип "Указатель"). Вводится новый указатель - список, достижимый из текущего указателя по метке x17. Если информационный блок только что инициализирован (x12 =0), то этот указатель-список становится корневым. Данная возможность в принципе может понадобиться лишь для 1-го и 4-го блоков, так как лишь у них корневой указатель может быть списком. Однако, здесь она тоже вряд ли нужна. Остальные информационные блоки заведомо должны инициализироваться вне данного интерфейса.

После обработки команды x18 происходит переход через "ветвь 1" в фрагменте, начинающемся с операторов "ветвь 1 не(равно(x18 внутрвывод))". К этому моменту переменная x18 содержит уже не указатель на тип нового объекта, а ссылку на него в файле. Здесь выполняется коррекция стэка x12, соответствующая переходу к объекту редактирования, и происходит откат к запросу очередной команды.

е) x16 = "узелраздела" (клавиша F3) - уплотнение текущего информационного блока;

ж) x16 = "модули" (клавиша Ctr-Del) - исключение текущего объекта информационного блока. Следует быть осторожным в случае текущего указателя-списка - удалена будет не его текущая ветвь, а весь указатель целиком.

з) x16 = "верхняяоценка" либо x16 = "минимум" (клавиши "ц","л") - соответственно, просмотр текстового цветного терминала и логического терминала.

и) x16 = "заменагруппы" (клавиша "б") - просмотр и изменение битмэпа.

к) x16 = "Стандплюс" (клавиша "к") - регистрация содержимого S текущего логического терминала в буфере для последующего копирования. Роль буфера играет комментарий (Вставка S) к посылкам исходной задачи.

л) x16 = "группировки" (клавиша End) - внутренний перезапуск и возвращение в главное меню.

м) x16 = "утверждение" (клавиша Ctr-м). Происходит изменение метки перехода к выделенному подпункту текущего указателя-списка. Текстовым редактором вводится новая версия метки перехода (логический символ либо терм), после чего все

объекты, к которым от текущего указателя имелся переход по старой метке, оказываются достижимы из него по новой метке.

13.1.10 Вход в просмотр и редактирование вспомогательных меню

Используемые в системе меню операционной системы Windows, прорисовываемые в верхней части экрана, представляют собой разновидность ресурсов, которые можно создавать либо редактировать с помощью специального интерфейса. Вход в этот интерфейс реализуется по x9 = "фильтрквадратов" (клавиша "м"). В начале фрагмента программы активизируется первый информационный блок и определяется ссылка на корень оглавления, с помощью которого и задаются вспомогательные меню (к этому корню имеется переход по метке "Меню" от корневого указателя-списка первого информационного блока). При обращении к оператору "оглавление(начало(x12)x13)" осуществляется (обычными средствами редактирования оглавления) ввод нового меню либо изменение старого. Начало ветви отдельного меню выбирается в оглавлении произвольно, а при его использовании нужно лишь сослаться на путь к этой ветви от корня оглавления. После выбора концевого пункта оглавления - переход к части программы, с помощью которой вводится тот код клавиатуры, который будет относиться к соответствующему концевому пункту меню. В верхней части экрана при этом перерисовывается текст выбранного концевого пункта оглавления, под ним проводится горизонтальная черта, и под ней - прорисовывается ранее выбранный код клавиатуры (если его не было, то под чертой пусто). Далее - переход через "ветвь 3" к оператору "автоклавиатура(x21)", вводящему команды набор кода. Если x21 = "тринадцать" (клавиша Enter), то происходит обращение к текстовому редактору для набора кода; если x21 = "цикливариантов" либо "Плюс" (клавиши "курсор влево" и Esc), то - возвращение в оглавление.

13.1.11 Операции со словарем текстового анализатора

Для работы с текстовым анализатором необходим специальный словарь, отличный от словаря логических символов. Этот словарь называется в логической системе внешним словарем. Он хранит фрагменты слов, на которые разбивается считываемое слово текста. Каждый такой фрагмент кодируется некоторым логическим символом. Для ввода новых словарных фрагментов, сопоставления им логических символов и простейшей характеризации, используемой процедурами текстового анализатора, создан специальный интерфейс. Он запускается нажатием клавиши "в" в главном меню (либо при входе в подпункт "Ресурсы и установки" главного меню). Возникает диалоговый блок, в котором выделены две основные рамки - "Слово" (для ввода словарного фрагмента) и "Логический символ" (после ввода фрагмента прорисовывается ранее введенный кодирующий его логический символ либо происходит ввод такого символа). После ввода словарного фрагмента в окне "Фрагмент словаря" прорисовываются ближайшие к нему в лексикографическом порядке уже имеющиеся в словаре фрагменты. Для активизации текстового редактора в рамках "Слово", "Логический символ" достаточно нажать левую кнопку мыши, подведя ее курсор в соответствующую рамку. Для сохранения во внешнем словаре введенного словарного фрагмента нажимается "Insert". После этого можно задать простейшие характеристики фрагмента, нажав клавишу "о". Это нажатие переводит в другой диалоговый блок. Сверху воспроизведено характеризуемое слово, а под ним расположены рамки,

перечисляющие названия частей речи. Нужное выбирается нажатием левой кнопки мыши. В случае, например, имени существительного появляется новый диалоговый блок. Его рамки соответствуют характеристикам слова (род, одушевленный-неодушевленный, единица измерения, процесс, сокращение и т.п.). Для выбора нужного используется все та же левая кнопка мыши, причем ее нажатие включает (появляется плюс) либо выключает (плюс пропадает) выбранный пункт. Две специальные рамки выделены для указания условного номера склонения слова - в единственном и множественном числе. После выбора такой рамки появляется соответствующий диалоговый блок, рамки которого соответствуют отдельным вариантам склонения. Фактически под склонением здесь понимается просто набор окончаний слова в различных падежах. Нужное выбирается левой кнопкой мыши, затем происходит возвращение (нажатие правой кнопки мыши) к тому диалоговому блоку, в котором перечислялись части речи, и здесь нажимается "Insert". Результатом является появление оттранслированных приемов ряда простейших справочников (прежде всего справочника "слово") для нового слова, причем эти приемы зарегистрированы в разделе "Словарь" базы приемов.

Заметим, что вход в интерфейс редактирования внешнего словаря может происходить автоматически - если при чтении текста встретилось незнакомое слово, то чтение обрывается, происходит переключение на указанный интерфейс, и новое слово прорисовывается в окне "Слово".

Войти в начальный фрагмент программы интерфейса работы с внешним словарем можно через пункт "Общий интерфейс - Операции с внешним словарем - Исходная точка" оглавления программ. Здесь x9 = "легковидеть". Переменной x9 переприсваивается в качестве значения переменная с номером 1 - впоследствии значением x9 будет тот логический символ, который используется в качестве кода словарного фрагмента. Затем идет оператор "повторение", к которому будут происходить откаты при повторных запросах команд блоком диалога. Оператор "началодиалога(надфрагмент 1 x10)"выполняет фоновую прорисовку блока диалога и создает структуру данных x10 для дальнейшего диалога. Если x9 - не переменная, то далее выполняется прорисовка логического символа x9 в окне "Логический символ" (оператор "или(переменная(x9)рис(x10 ...))"). После оператора "ветвь 2" расположен фрагмент, используемый в тех случаях, когда при чтении текстовой задачи встретилось незнакомое слово. Комментарий (анализслова N_1 N_2) к посылкам исходной задачи хранит смещения N_1, N_2 в массиве текстов начала и конца отрезка, содержащего такое слово. Происходит извлечение слова в буфер текстов начиная с позиции x13, закрепленной за рамкой "Слово" данного блока диалога, прорисовка его в этой рамке и обращение к текстовому редактору для коррекции слова (отбрасывания суффиксов и окончания). По окончании редактирования результат (пока никак не зарегистрированный) остается в буфере текстов начиная с позиции x13. Далее - переход через оператор "ветвь 2".

Здесь размещается оператор "повторение", к которому будут происходить откаты при повторных запросах обращения к блоку диалога. За ним следует оператор "альтернатива(Входит(анализслова комментариипосылок(x1))и(равно(x11 усмчисло)...) диалог(x10 x11))". Этот оператор обращается к запросу команды x11 диалога либо (если при чтении текста встретилось незнакомое слово, которое уже перенесено в буфер текстов) сразу полагает x11 равным символу "усмчисло", т.е. метке рамки "Слово". Далее идет цепочка обработки вариантов команды x11:

а) x11 = "стоп" (нажатие клавиш "Esc" либо "End" либо "курсор влево"). Тогда происходит внутренний перезапуск.

б) x11 = "усмчисло" (обращение к рамке "Слово"). В структуре данных диалога x10 находится элемент (текстредактор усмчисло N), определяющий позицию N буфера текстов, начиная с которой размещается словарный фрагмент. Находится позиция x14 конца этого фрагмента в буфере текстов. Если словарный фрагмент F, расположенный между N и x14, является началом некоторого ранее сохраненного во внешнем словаре фрагмента, то накопитель x15 заполняется не более чем 10 первыми логическими символами, кодирующими словарные фрагменты, следующие в лексикографическом порядке за F. Затем - переход через "ветвь 2".

Переменной x16 присваивается 0 - она будет хранить номер текущей строки в рамке "Фрагмент словаря". Из x15 последовательно извлекаются символы x17; на очередной строке прорисовывается сначала кодируемый этим символом словарный фрагмент, а затем в скобках (символы "соединение","учеткоманды" суть коды открывающей и закрывающей скобок) - сам логический символ. Заметим, что оператор "видео(терм ...)"имеет совпадающие указатели цвета и фона символов, т.е. он лишь заносит буквы названия логического символа в буфер текстов, не выполняя на экране никакой прорисовки. Эту прорисовку будет выполнять оператор "рис(x10 1 набор(текстредактор 0 x21 x16))"; здесь 1 - метка рамки "Фрагмент словаря". По завершении прорисовки данных о символах набора x15 - переход через "иначе 1".

Выполняется поиск логического символа x17, кодирующего словарный фрагмент F. Если такой символ найден, причем рамка "Логический символ" пока пуста, то он прорисовывается в данной рамке.

в) x11 = "минимум" (обращение к рамке "Логический символ"). Содержимое данной рамки отображается элементом (новыйсимвол минимум S) структуры данных x10. Вначале S - переменная, а при обращении к рамке (обрабатываемом внутри оператора "диалог(x10 x11)") происходит замена S на введенный текстовым редактором логический символ. Более того, при вводе нового названия внутри рамки, снабженной элементом (новыйсимвол ...), оператор "диалог(...)"создает новый логический символ с таким названием. Поэтому рассматриваемая ветвь программы лишь комментирует уже введенный символ S (т.е. x13). Переменной x9 переприсваивается значение x13, после чего расчищается прямоугольная рамка с меткой 2, расположенная непосредственно под рамкой "Логический символ", но скрытая (ее границы имеют цвет, совпадающий с цветом фона). В этой рамке прорисовывается информация об использовании символа S в качестве кода словарного фрагмента. Если удается найти такой фрагмент F (оператор "внешсловарь(значение x13 ...)" истинен, то в рамке "2" прорисовывается текст: "За логическим символом закреплено слово: F". Иначе прорисовывается текст "С логическим символом не связано слово".

г) x11 = "внешдизъюнкция" (Обращение к рамке "Ввести"). Определяется логический символ x14, введенный в рамке "Логический символ". Если он еще не имел программы, то создается простейшая его программа, состоящая из единственного оператора "продолжение". Это делается для того, чтобы предотвратить сообщения интерпретатора ЛОСа об ошибках, которые появятся, например, при обращениях различных справочников к несуществующей программе символа x14. Затем (после перехода по "ветвь 2") находятся начало x15 и конец x19 словарного фрагмента в буфере текстов, и оператор "внешсловарь(запись x14 x15 x19)" регистрирует x14 в качестве кода данного фрагмента.

д) x11 = "модули" (Клавиша Ctr-Del). Происходит удаление словарного фрагмента, закрепленного за логическим символом, введенным в рамке "Логический символ". Если отсутствовала программа этого символа, то удаляется и он сам.

е) x11 = "обл" (клавиша пробела). Расчистка рамок при переходе ко вводу очеред-

ного словарного фрагмента. Указатель логического символа (новыйсимвол минимум S) тоже расчищается - S заменяется на символ переменной.

ж) x11 = "частичныйответ" (Обращение к рамке "Сведения о слове"). Здесь реализуется создание программ справочников "слово", "симв", "падеж", "Падеж", обслуживающих логический символ - код словарного фрагмента и отображение ранее созданных таких программ. Вообще говоря, эти программы можно создавать непосредственно через редактор ГЕНОЛОГа, однако данный интерфейс упрощает и ускоряет их создание. Прежде всего определяется символ x13 и контролируется наличие его программы (при отсутствии - вводится простейшая программа "продолжение"). Далее (переход через "ветвь 1") накопитель x14 заполняется парами (слово R), (симв R) для ненулевых результатов R обращения к справочникам "слово", "симв" на символе x13. После этого - переход через "ветвь 1".

Здесь инициируется диалог для указания части речи, к которой относится рассматриваемое слово. После прорисовки оператором "рис(x15 слово набор(логсимвол x13))" в верхней рамке логического символа, кодирующего слово, определяется метка x16 той рамки, в которой должен быть прорисован знак "плюс". Такие скрытые рамки располагаются справа от рамок, в которых прорисованы названия частей речи. После прорисовки плюса (если на основании информации, имеющейся в накопителе x14, не удается определить часть речи, то плюс не прорисовывается) - переход через "ветвь 1", где оператор "диалог(x15 x16)" запрашивает очередную команду x16. Типы выполняемых команд таковы:

1) x16 = "стоп" - откат к внешнему диалогу.

2) x16 = "1" (выбор рамки "Имя существительное"). Инициализируется диалог для характеризации имени существительного. Сначала на основе информации из x14 расставляются плюсы справа от тех рамок, которые относятся к данному случаю. Кроме того, справа от рамок "Склонение в единств. числе" и "Склонение во множеств. числе" указываются номера соответствующих склонений. После этого - обращение к оператору "диалог(x17 x18)", запрашивающему команду внутри текущего диалога. Большинство этих команд сводятся к выбору либо исключению характеристики слова, указанной в соответствующей рамке; при выборе справа от рамки возникает "плюс", при исключении (повторный выбор рамки) - этот "плюс" исчезает. При этом корректируется список характеризующих слово логических символов из накопителя x14. Лишь в двух случаях - x18 = "4" и x18 = "двенадцать" - происходит дальнейшее углубление в блоки диалога; на этот раз для определения номера склонения в единственном либо во множественном числе (элементы (падеж N) и (Падеж N) из x14). Здесь принцип обработки команды тот же - прорисовка либо исключение "плюса" и коррекция накопителя x14.

3) x16 = "2" (рамка "Глагол"). Здесь характеризация выполняется также отдельным диалогом, который пока сводится к выбору одного из двух пунктов - "действие" либо "отношение".

4) x16 = "внешдизъюнкция" (рамка "Ввести", клавиша "Insert"). Из накопителя x14 извлекается сначала набор x18 характеризующих текущее слово логических символов, который должен выдаваться по кодирующему данное слово логическому символу x13 справочником "слово". Если приема этого справочника еще не было, то он создается, иначе - корректируется. Аналогичным образом создаются либо корректируются приемы справочников "падеж", "Падеж"(единств. и множеств. числа).

13.1.12 Обращение к оглавлению базы теорем

Обращение к базе теорем соответствует случаю x9 = "заменагруппы"; войти в данный фрагмент программы можно через пункт "База теорем - Обращение к базе теорем из главного меню" оглавления программ. Прежде всего, проверяется наличие шестого информационного блока (в нем хранится база теорем); если его не было, то он вводится и инициализируется его корневой указатель - каталог. Далее - переход через "ветвь 2". Здесь шестой информационный блок активизируется и определяется ссылка x10 на корень оглавления базы теорем. Сначала переменной x10 присваивается 0, затем проверяется наличие переходов "оглавление", "теорема" от корневого указателя-каталога к корню оглавления; если таких переходов не было, то они создаются. После определения x10 - переход через "ветвь 1".

В начале нового фрагмента располагается оператор "повторение", к которому будут происходить откаты при возвращении в оглавление базы теорем из просмотра очередной теоремы. Оператор "альтернатива(... оглавление(x10 x12))" реализует обращение к оглавлению базы теоремы и получает пару x12 ссылок на текстовый и логический терминалы выбранного концевого пункта оглавления; при наличии комментария "автоклавиатура ..." к посылкам исходной задачи, указывающего на эмуляцию нажатия клавиши "курсор вправо", вместо обращения к оглавлению сразу выдается ссылка x12 на сохраненный либо специально установленный его концевой пункт. Заметим, что оглавление обладает "памятью", хранящей некоторый путь от его корня, и эта память регулярно используется при переключениях между оглавлениями системы.

Далее переменной x12 переприсваивается второй элемент пары, т.е. ссылка на логический терминал концевого пункта оглавления базы теорем, и происходит обращение к оператору "блоктеорем(x12)" - редактору базы теорем. Его устройство будет описано ниже в отдельном разделе.

13.1.13 Прочие функции главного меню

При x9 = "заменазнака" происходит обращение к просмотру списка всех логических символов. Хотя это делается по нажатии клавиши "c" в главном меню, процедура "главноеменю" заменяет код клавиши "c" на код "заменазнака" клавиши "C", который и обрабатывается данной ветвью программы. Символы прорисовываются в два столбца на каждой странице, по 23 символа в каждом столбце. Для смены страницы служат клавиши "Page Down - Page Up".

При x9 = "стандменьше" (клавиша F1) происходит обращение к оператору "Помощь(0)", реализующему просмотр справочника по системе.

При x9 = "Стандплюс" (клавиша "к", кир.) происходит обращение к оператору "контейнер", предназначенному для обмена информацией между различными версиями логической системы.

При x9 = "частичныйответ" (клавиша "о", кир.) происходит обращение к хранящемуся в 5-м информационном блоке оглавлению операторов ЛОСа. После выбора концевого пункта этого оглавления происходит прорисовка информации о соответствующем логическом символе, извлекаемой из 3-го инф.блока.

13.2 Программа интерфейса оглавлений

Оглавления представляют собой один из наиболее распространенных в логической системе видов интерфейса. Они упорядочивают данные и позволяют инициировать различные операции над ними. Интерфейс оглавлений реализуется оператором "оглавление(x1 x2)". Он получает в качестве входного значения ссылку x1 на корневой указатель-список оглавления. Выходной переменной x2 присваивается пара ссылок - на текстовый терминал и на логический терминал выбранного концевого пункта оглавления. Структура данных, используемая для хранения оглавления в информационном блоке, уже была описана выше.

13.2.1 Типы оглавлений

Каждое оглавление имеет свое имя (иногда называемое также типом оглавления). Обычно переход к оглавлению происходит от корневого каталога информационного блока по двум меткам - сначала по метке "оглавление", и далее по метке - типу оглавления. Перечислим типы T основных оглавлений логической системы:

1. T = "прием" - оглавление базы приемов. Оно расположено в 8-м информационном блоке. Концевой терминал оглавления базы приемов содержит набор термов "прием($A_1A_2A_3$)", где A_1, A_2, A_3 - стандартная ссылка на прием ГЕНОЛОГа, состоящая из логического символа A_1, номера A_2 узла статьи символа A_1, хранящего теорему приема, и заголовка приема A_3. Заголовок одного из термов "прием(...)" здесь может быть заменен на символ "текприем", для выделения текущего приема, с которого нужно начинать просмотр приемов данного терминала. Кроме термов "прием(...)", в терминале может встретиться терм "замечание($B_1 \dots B_m$)", перечисляющий замечания $B_1, \dots, B_m$ к данному концевому пункту оглавления, используемые различными работающими с базой приемов процедурами. Из неконцевого пункта оглавления по метке "замечание" может быть переход к логическому терминалу, хранящему замечания $B_1, \dots, B_m$ аналогичных типов.

2. T = "программа" - оглавление программ ЛОСа. Оно находится в 9-м информационном блоке. Концевой терминал оглавления программ хранит терм "программа(A_1A_2)", где A_1 - логический символ, A_2 - номер узла U статьи символа A_1 в 9-м информационном блоке, связанного с некоторым пунктом в программе символа A_1. Этот пункт выделяется в данной программе при помощи вспомогательного оператора "прием(A_2)", где A_2 - последовательность цифр (вместо десятичной записи числа). По метке "оглавление" из узла U - переход к логическому терминалу, хранящему путь в оглавлении программ ЛОСа к рассматриваемому его концевому терминалу.

3. T = "заголовок" - справочное оглавление заголовков приемов. Это оглавление находится в 5-м информационном блоке, где расположены также многие другие справочные оглавления. Обычно концевой терминал справочного оглавления содержит единственный терм "прием($A_1A_2A_3$)". Здесь A_1 - лог.символ, A_2 - номер узла статьи символа A_1, от которого по метке "терм" имеется переход к логическому терминалу, хранящему тот терм S, который является вспомогательным обозначением, определяемым данным концевым пунктом справочного оглавления. A_3 - имя справочного оглавления. По метке "команды" от узла

статьи лог.символа в 5-м инф.блоке - переход к указателю - списку, единственная метка которого - имя рассматриваемого оглавления - ведет к указателю - списку, от которого по метке "оглавление" - переход к лог.терминалу, хранящему путь к соответствующему концевому пункту оглавления.

4. T = "условие" - справочное оглавление фильтров приемов (5-й инф.блок).

5. T = "идентификатор" - справочное оглавление указателей приемов (5-й инф. блок).

6. T = "прогрблок" - справочное оглавление информационных элементов прогр. блока, используемых компилятором ГЕНОЛОГа (5-й инф.блок).

7. T = "Меню" - оглавление меню, прорисовываемых в верхней части экрана (1-й инф.блок). Переход к корневому указателю этого оглавления - от корневого указателя списка 1-го инф. блока по метке "Меню".

8. T = "оператор" - справочное оглавление операторов ЛОСа (5-й инф.блок).

9. T = "нормистина" - оглавление блоков диалога, используемых в логической системе. Оно размещено в 5-м инф.блоке. Концевой терминал оглавления содержит терм "прием(A_1 A_2 нормистина)", где A_1 - логический символ, A_2 - номер узла статьи этого символа. Этот узел хранит описание формата блока диалога, как это было описано выше.

10. T = "теорема" - оглавление базы теорем. Оно хранится в 6-м информационном блоке. Концевой терминал оглавления содержит набор термов "теорема(A_1A_2)", где A_1 - логический символ, A_2 - номер узла статьи этого символа, из которого по метке "терм" - переход к логическому терминалу, содержащему теорему. Этот узел называем также узлом данной теоремы. Заголовок одного из термов "теорема(...)"может быть заменен на "нормуравн" (ранее здесь использовался символ "тектеорема", который затем получил другое название) - для указания на теорему, с которой начинается просмотр при обращении к терминалу. Терм "замечание($B_1 \dots B_m$)" в концевом терминале оглавления указывает набор замечаний $B_1, \dots, B_m$ к концевому пункту оглавления. Замечания к неконцевому пункту оглавления хранятся в логическом терминале, к которому от указателя этого пункта имеется переход по метке "замечание".

11. T = "задачи" - оглавление задачника. Оно размещено во 2-м информационном блоке. Концевой логический терминал оглавления задачника содержит терм "задача(A_1A_2)", где A_1 - логический символ, A_2 - номер узла статьи символа A_1, хранящего задачу. Как и в предыдущих случаях, для замечаний к концевому пункту оглавления могут быть использованы термы "замечание(...)" в концевом логическом терминале либо переходы по метке "замечание" от промежуточного указателя оглавления. Описание структур данных, используемых для хранения задач во 2-м информационном блоке, будет дано ниже в отдельной главе, посвященной программам интерфейса задачника.

12. T = "раздел" - справочное оглавление типов замечаний к разделу базы теорем (5-й инф.блок).

13. T = "блоктеорем" - справочное оглавление типов комментариев к теореме в базе теорем (5-й инф.блок).

14. T = "слово" - справочное оглавление типов комментариев к представлению слова либо терма фразы (5-й инф.блок).

15. T = "текстформ" - оглавление справочника по логической системе. Хранится в 7-м инф.блоке.

16. T = "контейнер" - оглавление контейнера (12-й инф.блок).

17. T = "цели" - оглавление типов целевых установок задач (2-й инф.блок).

18. T = "приемы" - оглавление типов установок на синтез приема (5-й инф.блок).

13.2.2 Инициализация обращения к оглавлению

В этом и следующих подразделах будет рассматриваться программа оператора "оглавление(x1 x2)". Основные ее точки выделены в оглавлении программ, в подразделе "Интерфейс оглавлений". Еще раз напомним, что при обращении к программе определено значение переменной x1 - ссылка на корневой указатель-список оглавления. При чтении целесообразно находить редактором программ ЛОСа соответствующие фрагменты программы "оглавление".

Оператор "равно(x3 пустоеслово)" инициализирует стэк x3 ссылок на узлы текущего пути в оглавлении. В нем будут сохраняться пары (ссылка на указатель оглавления - метка выхода из этого указателя вдоль текущего пути). Начало пути соответствует концу списка x3. Инициализируется также ссылка x4 на текущее меню оглавления. Следующий оператор "или(... сборкаменю(набор(7)))" проверяет, нужно ли будет прорисовывать оглавление, и если нужно, то прорисовывает в верхней части экрана вспомогательное меню операций оглавления.

Далее идут различные процедуры, проверяющие наличие специальных комментариев к посылкам исходной задачи, указывающих на необходимость переустановки пути в оглавлении для выхода в просмотр нужных объектов. Прежде всего (после оператора "ветвь 2") проверяется наличие комментария (теорема $A_1A_2A_3$), используемого для перехода от просмотра приема к просмотру той теоремы в базе теорем, которая является его источником. Если он есть, и A_3 не равно 0 (что означает наличие теоремы - источника приема), то A_3 есть одноэлементый набор, состоящий из терма "теорема(B_1B_2)", ссылающегося на теорему. Оператор "адресузла(... x8)" определяет ссылку x8 на узел теоремы в 6-м информационном блоке (где хранится база теорем). Переходя от x8 по метке "оглавление" к логическому терминалу и определяя его содержимое x10, получаем путь в оглавлении базы теорем к нужному концевому пункту. Используя оператор "путьвоглавлении(...)", переустанавливаем текущий путь согласно x10. Теперь при входе в оглавление сразу же будем попадать в данный концевой пункт. Однако, он может относиться сразу к нескольким различным теоремам. Поэтому нужно еще уточнить, какая из них является "текущей". Для этого считываем содержимое x12 выбранного концевого логического терминала оглавления. Если в нем уже была выбрана текущая теорема - ссылка вида "нормуравн(...)", - то отменяем выбор, заменяя заголовок ссылки на "теорема". Затем находим ссылку на нужную теорему (то есть указанный выше терм "теорема(B_1B_2)"), и заменяем ее заголовок на "нормуравн". В заключение, разумеется, изменяем содержимое концевого логического терминала оглавления в соответствии с указанными выше изменениями в его копии, считанной в зону задач. Переходя далее по "ветвь 3", вводим комментарий (автоклавиатура ...), эмулирующий

нажатие клавиши "курсор вправо". Дальнейшие действия программы "оглавление" теперь обеспечат выбор нужного концевого пункта и немедленную выдачу ссылки на него, а следующая за программой "оглавление" программа редактора базы теорем прорисует нужную теорему из списка теорем, отнесенных к данному концевому пункту оглавления. Описанная процедура переключения ссылки на заданный текущий объект заданного концевого пункта оглавления является стандартной и применяется в самых различных ситуациях.

Возвращаясь в корневой фрагмент программы "оглавление", переходим далее через "ветвь 2" и попадаем в фрагмент, начинающийся с операторов "ветвь 1 исходнаязадача(x5)". Здесь размещаются ветви программы, которые обеспечивают смену текущего концевого пункта оглавления в различных циклических процедурах (серийный запуск решения задач, тестирование вывода теорем, и т.п.).

Если исходная задача x5 имеет комментарий "задачи", указывающий на наличие цикла решения задач из некоторого раздела задачника, то оператор "Текзадача(x7 x8 x9)" определяет набор x7 ссылок на указатели оглавления задачника, относящиеся к текущему пути в этом оглавлении, набор x8 меток перехода из этих указателей и ссылку x9 на корневой указатель того раздела задачника, в котором предпринимается цикл решения. Метка перехода от указателя - конца набора x7 - к концевому логическому терминалу оглавления, хранящему ссылку на текущую задачу, расположена в конце набора x8. Используя эту информацию, далее ищем по принципу "сначала вглубь" очередной концевой логический терминал в дереве оглавления. Если это удается сделать, то после перехода через оператор "иначе 5" выдаем ссылку x11 на найденный терминал. Если же в процессе поиска доходим до корневого указателя раздела, в котором реализуется цикл решений (см. оператор "равно(x9 конец(x7))" в фрагменте, к которому переходим через оператор "иначе 4"), то отменяем режим серийного решения (удаляем комментарий "задачи" и заменяем содержимое логического терминала "метка" 2-го информационного блока на "задачи(0)"). Заметим, что комментарий "задачи" не обязательно указывает на серийный режим решения. Если он сопровождается комментарием "элементызадачи", то происходит откат к просмотру исходного условия задачи по окончании ее решения в обычном режиме. В этом случае - переход через оператор "иначе 3", удаление комментария "задачи", ввод комментария (автоклавиатура ...), эмулирующего нажатие клавиши "курсор вправо", и выдача ссылки на тот концевой терминал оглавления, к которому относилась решавшаяся задача.

Аналогичным образом осуществляется смена текущего приема либо теоремы в циклах тестирования процедур генератора приемов (комментарии "подраздел" и "команды").

Переходя через оператор "ветвь 1", расположенный перед оператором "исходнаязадача(x5)", попадаем в начинающийся с оператора "повторение" фрагмент, который осуществляет продвижение вглубь оглавления в соответствии с указателями на выделенный в нем текущий путь. Напомним, что изначально переменной x4 присвоена ссылка на корень оглавления. Логический терминал, достижимый из x4 по метке "позиция", содержит терм "число(ij)", где i - символьный номер перехода для текущего пути. Переходим из x4 по метке i (оператор "указатель(список x4 x8 x9)"). Если переместились в указатель-список, то ссылку на него переприсваиваем переменной x4, пополняем стэк x3 парой для перехода от старого указателя x4 к новому, и откатываемся к оператору "повторение". Если переместились в логический терминал, хранящий терм "логсимвол(pq)", то имеет место переход к отрезанной ветви оглавления. Ее корень достижим из корня информационного блока по меткам p, q.

Переприсваиваем переменной x4 ссылку на данный корень, пополняем стэк x3 и тоже окатываемся к "повторение". Наконец, по достижении концевого пункта оглавления переходим через оператор "ветвь 1".

Здесь начинается прорисовка текущего меню оглавления. После оператора "повторение", к которому будут выполняться откаты для перерисовки этого меню, расположена контрольная точка "прием(4 5)", к которой можно перейти непосредственно из пункта "Интерфейс оглавлений - Начало цикла прорисовки текущего меню" оглавления программ.

Прежде всего, инициализируется пустым набором накопитель x5 структур данных, необходимых для прорисовки меню. Этот накопитель будет представлять собой набор $(A_1, \ldots, A_n)$ элементов, характеризующих отображенные на экране и предшествующие им пункты меню. Каждое A_i имеет вид $(B_1B_2B_3B_4B_5)$, где B_1 - метка перехода к объекту i - го пункта меню; B_2 - ссылка на текстовый терминал этого пункта; B_3 - ссылка на объект, переход к которому осуществляется через данный пункт меню; B_4 - переменная, если пункт предшествует группе пунктов, отображенных на экране, либо пара (символьный номер строки, с которой выдается текст данного пункта - номер строки, на которой этот текст заканчивается); B_5 - 0, если i - й пункт меню ссылается на подменю, и 1, если он является концевым.

Здесь же инициализируются единицами номер x6 текущего пункта отображаемого меню и номер x7 пункта, с которого начинается прорисовка на экране. Далее расчищается экран и уточняются значения x6,x7. Для этого определяется терм "число(ij)", хранящийся в логическом терминале "позиция" указателя x4. Проверяется, что (i, j)
$\neq (0, 0)$, и x6 переприсваивается значение i при $i \neq 0$ и j в противном случае. Аналогично, если $j \neq 0$, то x7 переприсваивается значение j, иначе - i. После уточнения x6,x7 содержимое логического терминала "позиция" заменяется на "число(0 0)", т.е. ссылка на текущий пункт меню сбрасывается. Далее - переход через "ветвь 1".

В начале цикла прорисовки пунктов меню значение x8, которое будет представлять собой номер текущего обрабатываемого пункта меню, инициализируется единицей, а x9 (номер первой свободной строки на экране - нулем). Далее идет оператор "повторение", к которому будут происходить откаты при дорисовке пунктов в нижней части экрана, высвободившейся из-за прокрутки. Проверяется отсутствие комментария, эмулирующего нажатие клавиши "ш" - в этом случае прорисовка оглавления не нужна. Собственно цикл прорисовки начинается после оператора "повторение". Находится переход из указателя x4 текущего меню по метке x8. Если он ведет к концевому пункту оглавления, то x12 присваивается 1, иначе - 0. Далее проверяется наличие комментария "заголовокприема" - такой комментарий вводится в специальных случаях просмотра ветви оглавления программ, обслуживающей генератор приемов. Здесь блокируются некоторые переходы, и x12 изменяется на 1 там, где эти переходы нужно сделать "невидимыми". Затем - переход через "ветвь 4".

Формируется набор x13 для последующего занесения в накопитель x5. Если пункт с номером x8 не должен быть прорисован на экране (x8 меньше номера x7, с которого начинается прорисовка), то x13 заносится в x5, и переход к очередному пункту. Иначе - переход через "иначе 1". Здесь предпринимается считывание из файла текста текущего пункта меню (он размещается в буфере текстов с позиции 0 по x14); при считывании устанавливается нужный цвет символов. Для обычного пункта он черный, для текущего - голубой, для занесенного в буфер - красный (если одновременно пункт - текущий) либо малиновый. Далее прорисовывается номер пункта; в случае концевого пункта после него ставится точка, иначе закрывающая скобка.

Предпринимается обращение к оператору "видео(прямоугольник обл x9 прогртерм разныеточки 0)", который определяет рамку с заданным отступлением от левого края для прорисовки текста пункта, и оператор "видео(слово обл x9 0 x14 x17)" выполняет эту прорисовку. Затем находится текущая позиция (x18,x19) курсора после прорисовки текста, и вычисляется последняя занятая строка x20. Если она достаточно близка к нижнему краю экрана, то происходит откат через "ветвь 1" к оператору, закрашивающему белым фоном рамку пункта (считается, что пункт не поместился на экране), и цикл прорисовки меню завершается. Иначе - в наборе x13 указываются первая и последняя строки, занятые пунктом, этот набор регистрируется в накопителе x5, корректируется номер x9 первой свободной строки (x20 увеличивается на 19), и переход к очередному пункту.

По завершении цикла прорисовки - переход через "иначе 3", идущее после оператора "указатель(список x4 x8 x10)". Если пункт с номером x6, который изначально был определен как текущий, не поместился на экране, то здесь x6 переприсваивается номер последнего прорисованного пункта. Затем - выход во внешний фрагмент, и из него - переход через оператор "ветвь 1".

Здесь, после контрольной точки "прием(4 6)", происходит изменение цвета того пункта текущего меню, который был выбран в качестве текущего. Его выполняет большой оператор "или(...)". К этой точке будут происходить откаты при смене текущего пункта меню. Далее расположен оператор "повторение", и за ним - оператор "автоменю(x10)", запрашивающий команду для работы с оглавлением.

13.2.3 Обработка команд интерфейса оглавления

Перечислим действия обработчика команд в зависимости от текущей команды x10. Начальные точки соответствующих фрагментов программы легко находятся при перемещении вдоль "главного ствола" вглубь от оператора "автоменю(x10)". К большинству из них можно выйти непосредственно из оглавления программ (ветвь пункта "Интерфейс оглавлений - обработчик команд").

Перемещение по оглавлению

Если x10 - "циклвариантов" либо "Плюс" (клавиши "курсор влево" либо Esc), то происходит возвращение в надменю оглавления, а для корневого меню (что соответствует пустому накопителю x3) - выход из оглавления. При возвращении в надменю x4 присваивается первый элемент первого набора накопителя x3, отбрасывается начало накопителя x3, и откат к перерисовке надменю. Особо учтен случай комментария "заголовокприема" к посылкам исходной задачи: здесь блокируется выход из корневого меню той ветви оглавления программ, которая соответствует классификации типов приемов (используется в генераторе приемов).

Для перехода к новому текущему пункту оглавления предусмотрены следующие возможности: x10 = "внешсумма" (клавиша "курсор вверх") - переход к предыдущему пункту; x10 = "подстановка" (клавиша "курсор вниз") - переход к следующему пункту; x10 = "символтеоремы" (клавиша "Ctr - курсор вниз") - переход к нижнему из прорисованных на экране пунктов; x10 - символ от "префиксныйфильтр" до "мнимаяединица" (клавиши - цифры от 0 до 9) - переход к первому из прорисованных на экране пунктов, номер которого заканчивается на данную цифру. При нажатии любой из перечисленных клавиш прежде всего проверяется существование искомого пункта оглавления: если нужно сдвинуться к предыдущему пункту, то про-

веряется, что x6 больше 1, если нужно перейти к следующему, то проверяется, что из текущего указателя x4 имеется переход по метке x6+1, и т.п. Эти проверки реализуются оператором "альтернатива(равно(x10 внушсумма) ...)". Если результат проверки отрицательный (переход через "иначе 2"), то в случае клавиши "курсор вниз" происходит сдвиг вверх прорисованных на экране пунктов оглавления (верхний пункт пропадает, а следующий за ним становится верхним). Если же результат проверки положительный, то переход по "ветвь 3".

Здесь прежде всего происходит перекраска текущего пункта, который получает цвет обычного пункта (оператор "длялюбого(x11 x12 x13 x14 ...)"). Далее рассматриваются отдельные случаи для команды x10:

а) x10 соответствует нажатию клавиш "курсор вверх" либо "курсор вниз". Определяется номер x11 нового пункта. Если нажата клавиша "курсор вверх" и экран пустой (4-й элемент последнего набора накопителя x5 является переменной, так как все предыдущие пункты при прокрутке убраны с экрана), то x11 воспроизводит номер x6 последнего из сдвинутых вверх пунктов. При непустом экране x11 получается из x6 изменением на единицу. Если пункт с номером x11 прорисован на экране (находится набор x12 накопителя x5, соответствующий этому пункту, и проверяется, что его четвертый элемент отличен от переменной "x1"), то переменной x6 переприсваивается значение x11 и реализуется откат к изменению цвета нового текущего пункта. Если новый пункт еще не прорисован на экране, то переход через "иначе 2".

Здесь происходит разветвление: при обработке команды "курсор вниз" - переход через "иначе 1"; в оставшейся части данного фрагмента обрабатывается случай "курсор вверх". В накопителе x5 находится набор x13 с номером x11. Соответствующий пункт не прорисован на экране, но должен появиться в верхней части экрана при выполнении данной команды. Оператор "файл(набор ...)" считывает текст пункта в буфер текстов. Длина (x14+1) этого текста делится на ширину экрана, и таким образом находится число x17 строк, которые будет занимать текст. Высота x21 этих строк в пикселях получается умножением x17 на 19. Если для прорисовки нового текста в верхней части экрана не нужно отбрасывать нижние прорисованные пункты (под ними достаточно пусто места, либо экран вообще пуст), то переход через "иначе 2". В противном случае определяется вхождение x19 в набор x5 последнего остающегося на экране пункта; набор x5 заменяется на свой начальный отрезок вплоть до вхождения x19 и предпринимается коррекция в нем номеров строк (к ним прибавляется x21). В x5 указываются номера первой и последней строк для верхнего пункта (оператор "изменение(окрестность(x13 3)...)"). Затем изображение сдвигается вниз на x21 пикселей; x6 и x7 устанавливаются на номер x11 верхнего пункта, и откат к перерисовке текущего пункта. Аналогичные действия (но без отбрасывания конца набора x5) выполняются в случае перехода через "иначе 2".

Обработка случая "курсор вниз" начинается с того, что находится переход из текущего указателя x4 по метке x11 к новому пункту оглавления. Переменной x14 присваивается указатель типа этого пункта (1 - концевой, 0 - промежуточный). Текст пункта должен быть прорисован в нижней части экрана. Создается набор x15, соответствующий этому пункту, который впоследствии будет занесен в накопитель x5. Так как пункт пока не прорисован, в x15 вместо номеров его первой и последней строк указана переменная "x1". Текст пункта считывается в буфер текстов и определяются число x19 его строк, а также высота x20 в пикселях. Просматриваются вхождения x21 в набор x5, вплоть до обнаружения такого пункта x22, который кончается строкой, не меньшей x20. В нем и предшествующих пунктах вместо указателей строк проставляются переменные (эти пункты выносятся за рамки экрана). Пред-

принимается сдвиг изображения вверх, при котором в верхней части экрана будет прорисован первый пункт, следующий за x22 (оператор "видео(спуск ...)"). Корректируются номера строк в наборе x5 для оставшихся на экране пунктов. В наборе x15 проставляются номера строк для нового пункта. Номер x6 текущего пункта устанавливается на x11, корректируется номер x7 пункта в верхней части экрана. К концу набора x5 присоединяется набор x15. Номер x8 очередного считываемого из файла пункта и номер x9 первой свободной строки устанавливаются на дорисовку пунктов в нижней части экрана (если после нового пункта остается достаточно места для нескольких других пунктов), и откат к дорисовке.

б) x10 соответствует нажатию клавиши "Ctr-курсор вниз". Происходит просмотр набора x5 и передача переменной x6 номеров тех пунктов, которые прорисованы на экране. В результате x6 становится равно номеру последнего такого пункта. Далее - откат к перекраске текущего пункта.

в) x10 соответствует нажатию цифровой клавиши. Набор x5 просматривается до обнаружения первого прорисованного на экране пункта, ппоследняя цифра номера x12 которого равна номеру нажатой клавиши. Переменной x6 присваивается номер такого пункта, и откат к его перекраске.

Вход в подменю оглавления либо выбор концевого пункта осуществляются при x10 = "внешконъюнкция" (клавиша "курсор вправо") либо x10 = "тринадцать" (клавиша Enter). Прежде всего, здесь происходит сохранение в логическом терминале "позиция" указателя-списка x4 текущего меню терма "число(x6 x7)". Таким образом оглавление запоминает свой текущий путь. Далее в накопителе x5 находится список x13, соответствующий текущему пункту. Если пункт неконцевой, то переход по "иначе 2". В случае концевого пункта вводятся, в зависимости от контекста, те или иные комментарии к посылкам исходной задачи (в случае оглавления приемов создается комментарий "простыеделители P", где P - последовательность меток текущего пути в оглавлении; для оглавления операторов ЛОСа вводится комментарий, эмулирующий нажатие клавиши "курсор вправо", и т.п.), Затем выдается результат - пара ссылок на текстовый и логический терминал выбранного концевого пункта (они извлекаются из набора x13).

В случае неконцевого пункта - в начало стэка x3 заносится пара (x4 x6), ссылающаяся на пройденный указатель x4 и метку x6 выхода из него; определяется ссылка на новый указатель-список текущего меню (с учетом возможного разреза оглавления по данному переходу), и значение x4 заменяется на эту ссылку. Затем - откат к перерисовке меню.

Для немедленного выхода из оглавления служит клавиша "End" (x10 = "группировки"). В этом случае оглавление запоминает текущие значения x6,x7, так что при повторном попадании в оглавление будет снова прорисован текущий пункт с номером x6, а первым на экране пунктом будет пункт с номером x7. Далее - выход из процедуры "оглавление" по истинностному значению "ложь".

Перелистывание "страниц" текущего меню оглавления выполняется клавишами "Page Down" и "Page Up" (x10 = "точкапривязки" и x10 = "контрольунификации"). Фрагменты программы для этих команд располагаются в "главном стволе" обработчика команд не сразу после перечисленных выше команд, так что для их поиска можно использовать оглавление программ (пункты "Интерфейс оглавлений - Обработчик команд - PgDown либо PgUp").

Начнем с команды "Page Down". Прежде всего, здесь проверяется, что накопитель x5 непуст и что из указателя текущего меню x4 имеется переход по метке, на единицу большей последней метки в списке x5. Тогда указатели строк всех наборов

накопителя x5 заменяются на переменную "x1" (соответствующие пункты уходят вверх за рамки экрана). Экран расчищается, переменным x6 и x7 присваивается увеличенный на единицу номер последнего пункта из x5. Инициализируется номер x8 текущего прорисовываемого на экране пункта, а также номер x9 первой свободной строки экрана. Затем размещается оператор "повторение", к которому будут происходить откаты после прорисовки очередного пункта. Определяется переход по метке x8 от указателя x4; если этот переход ведет к промежуточному пункту, то переменной x13 присваивается 0, иначе 1. Создается заготовка x14 набора для последующего занесения в накопитель x5. Дальнейшие действия совершенно аналогичны тем, которые выполнялись при исходной прорисовке меню. По завершении прорисовки - откат к перекраске текущего (первого сверху) пункта.

В случае команды "Page Up" проверяется, что x5 непусто и что первый пункт накопителя x5 не прорисован на экране. Находится вхождение x12 в x5 последнего не прорисованного на экране пункта (у него вместо пары номеров первой и последней строк стоит переменная). Инициализируется нулем переменная x13. Она будет указывать вхождение в x5 того набора, с которого начнется прорисовка. Однако, вначале она же используется в качестве счетчика числа строк, занятых прорисовываемыми пунктами. Происходит просмотр набора x5 в направлении от x12 к началу, и для каждого просматриваемого пункта определяется число занимаемых им строк. Для этого пункт сначала считывается в буфер текстов, а затем длина текста делится на ширину экрана. Как только число строк превышает допустимое, просмотр прекращается и переменной x13 переприсваивается найденное вхождение в x5. Затем - переход через "ветвь 2".

Здесь происходит расчистка экрана и переменным x6, x7 переприсваивается номер пункта по вхождению x13. Инициализируется нулем номер x9 первой свободной строки на экране. Затем начинается последовательный просмотр вхождений x14 в набор x5, начиная с x13. Переменной x8 переприсваивается номер пункта, соответствующего x14. Прорисовка пункта происходит так же, как и в случае "Page Down". Если в процессе прорисовки размеры экрана оказались исчерпаны раньше, чем закончился список x5, происходит отбрасывание в конце этого списка элементов, не поместившихся на экране. В обоих случаях далее - откат к перекраске текущего пункта.

Редактирование оглавления

Оглавление в системе совмещает функции меню и "записной книжки", что делает необходимыми команды изменения оглавления в процессе его использования.

Для ввода нового пункта оглавления служат команды x10 = "Стандплюс" (клавиша "к" кириллицы; происходит ввод нового концевого пункта оглавления в конце текущего меню); x10 = "высотаполосы" (клавиша "К" кириллицы; происходит ввод нового концевого пункта перед текущим выделенным пунктом меню); x10 = "фильтрквадратов" (клавиша "м" кириллицы; происходит ввод нового пункта подменю в конце текущего меню); x10 = "значение" (клавиша "М" кириллицы; происходит ввод нового пункта подменю перед текущим выделенным пунктом меню). В начало ветви программы, обрабатывающей эти команды, можно попасть из пункта "Интерфейс оглавлений - Обработчик команд - Ввод нового концевого пункта либо нового подраздела" оглавления программ.

Прежде всего определяются номер x11 вставляемого пункта и номер x12 той строки, начиная с которой будет размещаться его текст (оператор "альтернатива(

...)"). В случае оглавления задачника проверяется, что тип нового пункта (концевой либо неконцевой) - тот же, что у ранее введенных пунктов данного меню. При несовпадении этих типов выполнение операции блокируется. Это вызвано тем, что у оглавления задачника концевые пункты собраны в специальные концевые меню и соответствующие им задачи отображаются на экране в виде сплошной "ленты", с разделяющими задачи горизонтальными линиями. После выполнения проверки - переход через "ветвь 2".

Проверяется, что для редактирования текста нового пункта на экране имеется хотя бы одна свободная строка, и выполняется прорисовка номера нового пункта, после которого ставится точка (для концевого пункта) либо скобка. Если вводится концевой пункт оглавления задачника, то в буфер текстов заносятся подряд три тире, иначе - предпринимается обращение к текстовому редактору для ввода текста пункта (оператор "альтернатива(и(или(равно(x10 Стандплюс)...)))"). При отказе от редактирования (переход через "иначе 1") - расчищается поле редактирования концевого пункта и откат к запросу команды, а для неконцевого пункта - откат к перерисовке меню. В случае вставки нового пункта в середине меню происходит увеличение на 1 номеров пунктов, начиная с номера x11 (оператор "указатель(плюссимв(x4 x11 1))"). Далее - переход через "ветвь 2".

Инициализируется нулем переменная x16, которой будет впоследствии переприсвоена ссылка на указатель-список либо логический терминал оглавления, вводимый для нового пункта. Если вводится новое подменю, то переход через "иначе 2", где создается пустой указатель-список x17 для этого подменю. В противном случае - рассматриваются следующие возможности:

а) Имеется комментарий (вставка $A_1A_2A_3A_4$) к посылкам исходной задачи. Этот комментарий используется при переходе от просмотра программы к оглавлению программ для передачи информации о вставке контрольной точки "прием(N)". A_1 - считанный в зону задач фрагмент программы; A_2 - позиция в нем, перед которой вставляется "прием(N)"; A_3 - набор ссылок на надфрагменты фрагмента A_1; A_4 - логический символ, в программу которого входит A_1. Создается новый узел x21 статьи символа A_4, имеющий номер x22. Переменной x19 присваивается терм "прием(N)", где N = x22. Вводится логический терминал, достижимый из указателя x4 текущего меню по метке x11, содержащий терм "программа(A_4 x22)". Это - ссылка на на узел x21 и на контрольную точку "прием(N)" из оглавления программ. Определяется последовательность меток пути в оглавлении программ от корня к новому пункту, и эта последовательность сохраняется в логическом терминале, достижимом из x21 по метке "оглавление". Это - возвратная ссылка на пункт оглавления из узла x21, которая будет нужна для поиска текста пункта оглавления, если при трассировке программы встретится оператор "прием(N)". Далее - переход через "ветвь 4", где происходит вставка в фрагмент программы A_1 оператора "прием(N)"и внесение соответствующих изменений в файлы блока программы.

б) Происходит ввод нового концевого пункта оглавления задачника. Тогда находится номер x19 текущего шага работы интерпретатора ЛОСа, который используется здесь в качестве случайного числа. Это число делится на 1000, к остатку прибавляется 300, и результат рассматривается как номер того логического символа x22, за которым будет закреплена новая задача, соответствующая вводимому пункту. Вводится узел x24 этой задачи, имеющий номер x25. В логическом терминале "оглавление" узла x24 регистрируется возвратная ссылка на новый пункт оглавления и вводится концевой логический терминал "задача(x22 x25)" этого пункта.

в) Вводится концевой логический терминал нового пункта, содержащий терм

"фикс(0)". Этот терм - стандартный указатель на пустой концевой пункт оглавлений.

После рассмотрения перечисленных выше вариантов для ввода указателя-списка либо логического терминала, соответствующего новому пункту, переход через оператор "ветвь 1", расположенный после оператора "равно(x16 0)".

После проверки того, что x16 не равно 0, т.е является ссылкой на уже введенный новый объект оглавления, вводится текстовый терминал, сопровождающий этот объект. Для этого используется текст нового пункта, ранее сформированный в буфере текстов. Дальнейшие действия зависят от того, вводится ли новый пункт в конце текущего меню или в середине. В последнем случае - переход через "иначе 1". В первом - определяется номер x20 последней строки, занятой текстом введенного пункта, и к накопителю x5 присоединяется набор x21, описывающий этот пункт. Если введен новый концевой пункт оглавления задачника, то он запоминается в оглавлении как последний пункт текущего пути, и сразу же выдается результат - ссылка на новый пункт. Это приводит к автоматическому входу в редактор задач для набора условия новой задачи. В остальных случаях происходит перекраска ранее имевшегося текущего пункта в "обычный" цвет, выбор нового пункта в качестве текущего, и откат к его перекраске.

Если новый пункт введен в середине меню (см. указанный выше переход через "иначе 1"), то он вызовет сдвиг в нумерации последующих пунктов меню, а при этом изменятся и последовательности меток, определяющие пути в оглавлении к различным его концевым вершинам, проходящие через данное меню по метке с номером, большим или равным x6. Однако, эти последовательности меток хранятся в качестве возвратных ссылок на оглавление из различных структур данных (описаний приемов, задач, и т.д.), на которые ссылается оглавление. Поэтому необходим просмотр таких возвратных ссылок и их коррекция. Для этого и служит ветвь программы, начинающаяся с оператора "ветвьоглавления(x4 ...)". Данный оператор просматривает все концевые пункты оглавления, достижимые из текущего меню x4 по путям, начинающимся с меток, больших или равных x6. Для текущего пункта он выдает тройку x18: (содержимое логического терминала концевого пункта - пара ссылок на терминалы данного пункта (текстовый и указанный выше логический) - набор лог. символов, определяющих путь к терминалу от корня оглавления). Просматриваются элементы $F(S\ N)$ x19 логического терминала текущего концевого пункта, ссылающегося на некоторую внешнюю по отношению к оглавлению структуру данных - их заголовки F перечисляются в списке "задача", "текприем", "теквхожд", и т.д. Каждый такой элемент ссылается на N-й узел статьи символа S в том же информационном блоке, что и оглавление. Находится ссылка x21 на этот узел. Далее происходит разбор случаев по F, определяющему, где именно нужно искать возвратную ссылку на оглавление.

После коррекции возвратных ссылок проверяется, не был ли введен концевой пункт оглавления задачника - тогда, как и выше, происходит выход из процедуры "оглавление". Иначе - откат к перерисовке меню.

Следующая операция по редактированию оглавления - изменение текста текущего пункта (x10 = "числитель"; клавиша "p" кир.). Здесь определяется набор x11 накопителя x5, соответствующий текущему пункту и проверяется, что этот пункт изображен на экране. Находится номер x13 строки, с которой начинается текст пункта. Затем расчищается буфер текстов; текст пункта считывается в буфер текстов (на позиции с 0 по x14), и происходит рачистка экрана вниз начиная со строки x13. После этого порисовывается номер редактируемого пункта, устанавливается рамка, необ-

ходимая для повторной прорисовки старого текста пункта, он прорисовывается, и предпринимается вход в текстовый редактор с сохранением старого текста. После редактирования происходит регистрация нового текста в файле. Далее (а также при отмене редактирования) - откат к перерисовке меню.

Чтобы перености выбранные пункты оглавления на новое место, они сначала заносятся в буфер оглавления - комментарий (оглавление A) к посылкам исходной задачи. Здесь A - набор троек (ссылка на меню оглавления, содержащее отобранный пункт - набор меток пути к этому меню от корня оглавления - метка перехода к выбранному пункту меню). Для занесения в буфер оглавления либо исключения из него текущего пункта служит команда x10 = "заменагруппы"(клавиша "б"). Если уже имелся комментарий (оглавление A), то формируется тройка x13 в формате набора A, соответствующая текущему пункту. Если x13 входит в A, то она исключается из A, иначе - заносится в A. При отсутствии буфера - он вводится с зарегистрированной в нем тройкой x13. Во всех перечисленных случаях далее - откат к перекраске текущего пункта (цвет его меняется в зависимости от принадлежности буферу).

Для перенесения отобранных в буфер пунктов на новое место используются команды x10 = "новыесвязки" (клавиша "и") и x10 = "верхняягрань" (клавиша "И"). В первом случае пункты присоединяются к концу текущего меню, во втором - вставляются перед текущим пунктом этого меню. Порядок их - тот же, что при отборе в буфер. Прежде всего, находится комментарий x12 (оглавление A) и проверяется, что A непусто. В случае x10 = "новыесвязки" выполняется цикл просмотра меток указателя x4 и присвоения переменной x14 номера, на единицу большего последней из этих меток. Затем - переход через "ветвь 2" и переприсвоение переменной x13 номера того пункта, начиная с которого будут регистрироваться извлекаемые из буфера пункты.

Далее начинается просмотр троек x15 из буфера A. Если вставка пунктов происходит в середине меню, то увеличиваются на 1 все метки указателя-списка x4, начиная с x6. Одновременно просматривается буфер A, и если в нем имелась ссылка на пункт меню x4 с номером, большим или равным x6, то в этой ссылке также происходит увеличение номера пункта на 1. Находятся "старые" указатель x16 и метка x17, по которым зарегистрирован пункт тройки x15, и этот пункт переключается на указатель-список x4, из которого он становится достижим по метке x14 (случай занесения в конец меню) либо x6 (занесение в середину меню). Такое переключение осуществляется двумя операторами "указатель(перестановка ...)" - сначала переносится основной объект пункта, а затем его текстовый терминал. Далее предпринимается уменьшение на 1 номеров перехода из указателя x16, больших или равных x17 (чтобы устранить пробел в нумерации, возникший после удаления перехода по метке x17). Пути в оглавлении, упоминаемые в накопителе x3 и в буфере A, корректируются в соответствии с этим уменьшением номеров. Значение x14 (в случае занесения пунктов в конец меню) либо значение x6 (при занесении в середину) увеличивается на единицу, если только не имело место перенесение пункта "вперед" из текущего меню в него же. Если через указатель-список x16 проходил текущий путь в оглавлении, то его продолжение после x16 сбрасывается. Затем - откат к очередному элементу x15 накопителя A. После просмотра всех таких элементов - переход через "иначе 1".

После того, как пункты оглавления из буфера A перенесены на новое место, нужно провести коррекцию обратных ссылок для оглавления. Для этого A преобразуется в список ссылок на начала тех ветвей оглавления, которые требуют контроля обратных ссылок. Прежде всего, в начало набора A заносится ссылка на переход из

текущего меню x4 по метке x13 (с этой метки начинаются перенесенные в x4 пункты). Далее просматриваются имеющиеся в A переходы из одного и того же меню. Сохраняется тот из них, для которого номер метки перехода меньше, а другой заменяется в A на 0 (рассмотрение ветви первого включает в себя рассмотрение ветви второго). Затем - переход через "ветвь 2", где реализуется последовательный просмотр ненулевых элементов из A и коррекция обратных ссылок в их ветвях. Это делается точно так же, как и в случае ввода нового пункта оглавления (см. выше) - с помощью оператора "ветвьоглавления(...)", перечисляющего концевые пункты оглавления, достижимые из данного подменю по путям, начинающимся с метки, большей или равной заданной. После коррекции обратных ссылок - откат к перерисовке текущего меню.

Для удаления текущего пункта оглавления служит команда x10 = "модули" (клавиша Ctr-Del). Прежде всего, в накопителе x5 находится набор x11, описывающий текущий пункт, и проверяется, что этот пункт прорисован на экране. Затем находится объект, к которому от указателя-списка x4 имеется переход по текущей метке x6. Если этот объект - указатель-список, из которого имеется переход по метке 1 (т.е. его подменю непусто), то удаление пункта блокируется (откат через оператор "ветвь 2", продолжающий действия по удалению пункта). Аналогичная блокировка производится, если найденный объект - логический терминал, по которому проходит разрез оглавления, а продолжение оглавления после разреза непусто. Если же найденный объект - концевой пункт оглавления, то предпринимаются действия по удалению прямых и обратных ссылок, связанных с данным пунктом. Это выполняется начиная с фрагмента, к которому от исходного фрагмента (содержащего оператор "равно(x10 модули)") имеются переходы через операторы "иначе 3" и "иначе 1".

Прежде всего, находится комментарий x17 (вид T) к исходной задаче, определяющий тип T рассматриваемого оглавления. Здесь рассматриваются следующие случаи:

а) Оглавление базы теорем либо базы приемов. Проверяется, что концевой пункт не ссылается на какую-либо теорему либо на прием, не исключенный из базы приемов. Если это не так, то удаление отменяется, иначе - продолжение действий по удалению пункта (в этом и последующих случаях дальнейшие действия по удалению происходят в ветви программы, к которой от исходного фрагмента обработки данной команды имеется переход через "ветвь 2").

б) Оглавление справочного характера (типы оглавлений перечислены в операторе "входит(конец(x17)набор(...))"). Здесь предпринимается исключение узла статьи логического символа, на который ссылается концевой пункт оглавления.

в) Оглавление задачника. Исключается узел задачи; в случае задачи на анализ рисунка из тринадцатого инф.блока, хранящего битмэпы рисунков, исключается соответствующий битмэп.

г) Оглавление программ. Удаляется узел, на который ссылается концевой пункт, после чего в программе находится контрольная точка "прием(N)", которая была связана с удаленным концевым пунктом оглавления, и эта точка исключается из программы.

Далее действия продолжаются в ветви программы, достижимой по оператору "ветвь 2" из исходного фрагмента обработки команды удаления пункта. Здесь исключаются ссылки по метке x6 из текущего указателя-списка x4 на удаленные объект и его текстовый терминал. Номера оставшихся ссылок из x4, превосходящих x6, уменьшаются на 1. Предпринимается коррекция обратных ссылок для ветви оглавления, в которой после указанного вычитания единицы изменились пути (последовательности проходимых меток) к концевым пунктам. Затем - переход к перерисовке

меню. Текущим пунктом становится пункт, следующий после удаленного (т.е. номер x6 сохраняется).

Дополнительные операции оглавления

Кроме перечисленных выше простейших общих операций, процедура "оглавление" выполняет большое число специализированных операций, связанных с теми блоками логической системы, которые обслуживаются оглавлениями. Более подробное описание этих операций естественно отложить до описания соответствующих блоков (например, базы приемов или базы теорем). Здесь мы ограничимся лишь кратким перечислением таких операций.

Команда x10 = "внешдизъюнкция" (клавиша Insert) служит для регистрации в текущем концевом пункте оглавления базы приемов либо теорем тех приемов (либо теорем), которые предварительно были занесена в буфер при просмотре этой базы. Регистрируемый прием (теорема) исключается из того концевого пункта, в котором он до этого находился. В случае базы приемов роль буфера играет комментарий (ссылканаприем A), где A - набор пар (ссылка на концевой терминал оглавления, где находился прием - терм "прием($B_1B_2B_3$)", адресующий прием). В случае базы теорем аналогичным образом используется комментарий (блоктеорем A); A - набор пар (ссылка на концевой терминал оглавления базы теорем - терм "теорема(B_1B_2)").

Команды x10 = "минимум", x10 = "арккосинус" (клавиши "л" и Ctr-л) служат для просмотра, соответственно, использованных либо неиспользованных в текущем вспомогательном оглавлении логических символов. Такой просмотр помогает найти подходящий новый логический символ при пополнении типов обозначений, перечисляемых в оглавлении.

Команды x10 = "элементы", x10 = "нормвариант" (клавиши F8, Ctr-F8) служат для запуска цикла просмотра всех приемов либо всех теорем базы приемов (соответственно, теорем). Вторая из них используется только в случае базы приемов и находит все неоттранслированные приемы. Первая в процессе просмотра обращается к вспомогательной процедуре ("текприем" для базы приемов и "тектеорема" для базы теорем). Этой вспомогательной процедуре передается вся информация о текущем просматриваемом объекте, так что перед просмотром можно перепрограммировать ее на заданную серийную обработку приемов либо теорем. Наиболее часто вспомогательная процедура используется для просмотра приемов либо теорем, удовлетворяющих заданному условию. Чтобы просмотр текущего объекта состоялся, процедура должна выдать ответ - набор, включающий в себя логический символ "См".

Команда x10 = "оператор" (клавиша Ctr-р; кир.) служит для перехода в справочное оглавление, перечисляющее типы комментариев, которыми могут сопровождаться разделы оглавления базы теорем. Эти комментарии (лог,символы и термы) хранятся в логических терминалах "замечание" указателей-списков оглавления либо внутри термов "замечание(...)" из логических терминалов концевых пунктов. Они используются различными процедурами обработки базы теорем. Если тексты пунктов оглавления предназначены для оператора, работающего с логической системой, то комментарии к пунктам представляют собой "теневые" версии названий пунктов, предназначенные для самой логической системы.

Команда x10 = "обобщслагаемое" (клавиша "?") служит для поиска приема в базе приемов, если известны логический символ и номер узла теоремы приема в статье этого символа. Сначала здесь реализуется интерфейс для ввода символа и номера,

затем выбирается один из приемов, основанных на данной теореме, и происходит переключение текущего пути в оглавлении на концевой пункт, содержащий выбранный прием. Сам прием делается текущим в списке приемов указанного концевого пункта.

Команды x10 = "элементызадачи", "развертка", "стрелкапирса", "сжатиефильтра"(клавиши Shift - 1,2,3,4) служат для переключения между оглавлениями: программ (Shift - 1), базы приемов (Shift - 2), задачника (Shift - 3) и базы теорем (Shift - 4). Нажатие соответствующей клавиши приводит к переходу на текущий пункт соответствующего оглавления. Для перехода вводится комментарий к посылкам исходной задачи, указывающий то оглавление, куда нужно перейти, и предпринимается откат к обработке команд главного меню (введенный комментарий блокирует перерисовку главного меню, и внешне наличие отката никак не проявляется). При переходе в оглавление базы приемов из буфера задачника текущий путь в оглавлении базы приемов переустанавливается на тот прием, для слежения за срабатываниями которого был введен данный пункт буфера задачника.

Группа команд предназначена для запуска цикла решения задач в заданном разделе оглавления задачника, либо для серийной обработки разделов оглавлений базы приемов и базы теорем (в основном для тестирования процедур автоматического синтеза приемов и вывода теорем). Это - команды x10 = "новыйузел" (клавиша Ctr-з, кир.), x10 = "префикснаярекурсия" (клавиша Ctr-э), x10 = "элемент" (клавиша "З"), x10 = "заменазнака" (клавиша "С"), x10 = "записьзадачи" (клавиша Ctr-х), x10 = "сдвигпеременных" (клавиша Ctr-г), x10 = "дробнаявеличина" (клавиша Ctr-ч). Чтобы запомнить корень того раздела задачника, внутри которого происходит серийная обработка, используется логический терминал "позиция" корневого указателя-списка этого раздела. Он должен был хранить терм "число(ij)", где i - метка перехода из указателя вдоль текущего пути; j - номер пункта меню, с которого начинается прорисовка на экране. Для выделения корневой точки, на период серийной обработки в ней вместо такого j помещается 0.

Команда x10 = "список" (клавиша F9) позволяет быстро перейти из просмотра оглавления базы приемов в редактирование программы символа "текприем", используемой для поиска в базе приемов либо для автоматического преобразования приемов.

Команда x10 = "стандменьше" (клавиша F1) переводит в просмотр справочника по решателю.

Команда x10 = "новаяветвь" (клавиша Ctr-F6) служит для разрезания оглавления. Эта операция выполняется, если оглавление стало чрезмерно большим и не помещается в одном файле информационного блока. Она уже была описана выше.

Команда x10 = "обл" (клавиша "пробел") служит, чтобы сбросить список выделенных для просмотра приемов либо задач. Такой список сохраняется в комментарии (подборнеизвестных A_1 A_2 A_3) к посылкам исходной задачи.

Команда x10 = "кортеж" (клавиша Ctr-т) служит для коррекции обратных ссылок в оглавлениях базы приемов, базы теорем, оглавлении программ и оглавлении операторов ЛОСа, если эти ссылки по каким-либо причинам оказались испорченными. Просматривается вся ветвь текущего меню оглавления, начиная с текущего пункта этого меню. Для концевых пунктов оглавления определяются точки обратных ссылок, и в них заносится фактический путь к концевому пункту.

Команда x10 = "выражение" (клавиша Ctr-и) служит для просмотра и редактирования комментариев к текущему пункту оглавления. В случае неконцевого пункта комментарии $A_1, \ldots, A_n$ (логические символы либо термы) хранятся в логическом

терминале "замечание", в случае концевого - в терме "замечание($A_1 \dots A_n$)" из логического терминала этого пункта.

Команда x10 = "соответвхожд" (клавиша Delete) позволяет сбросить буфер приемов либо теорем, созданный для перенесения этих приемов или теорем на новое место в оглавлении.

Команда x10 = "подборнеизвестных" (клавиша "з") инициирует просмотр информации о решении задач раздела. Это делается с помощью диалоговых блоков решателя. При просмотре возможно создание списков задач, для которых произошло ускорение, замедление, утеря либо изменение ответа. Для этих списков вводится упомянутый выше комментарий (подборнеизвестных ...). Команда x10 = "схемаидентификации" (клавиша "ш") позволяет использовать данный комментарий для перехода к очередному элементу списка.

Команда x10 = "взаимнооднозначно" (клавиша "О") служит для сброса буфера базы приемов, теорем либо задачника. Такой буфер представляет собой раздел оглавления, создаваемый автоматически для вспомогательных целей. В случае базы приемов в него заносятся удаленные приемы, старые версии измененных приемов и новые приемы - чтобы на локальном отрезке обучения решателя легче было находить причины обнаруженных при тестовом решении задач ухудшений в работе. В случае задачника в буфер заносятся те задачи, для которых необходимо первоочередное тестирование (например, все задачи, имеющие срабатывание заданного приема). Размещается ветвь буфера в конце корневого меню оглавления.

Команда x10 = "нормпроизведениевсех" (клавиша "ф") позволяет перейти от произвольной точки оглавления к буферу этого оглавления; если буфера не было, то она игнорируется.

Команда x10 = "усмцелое" (клавиша "й") служит для анализа избыточности приема ГЕНОЛОГа по разделу буфера задачника, созданному для рассмотрения задач, в которых этот прием применяется. Происходит последовательный просмотр задач раздела, и для каждой предпринимается попытка ее решения с отключенным приемом. Чтобы отключить прием, используется комментарий (вычерк (прием A_1 A_2 A_3)0) к посылкам исходной задачи, который воспринимается процедурами "вывод", "заменавхождения" и пр., осуществляющими преобразования при решении задачи. Для решения отводится время, чуть большее того, которое было нужно без отключения приема. Если за это время задача не решается либо ответ изменяется, то переход к следующей задаче. Если же ответ сохраняется, а время решения уменьшается, то после текста, сопровождающего задачу в ветви буфера, ставится знак вопроса - возможно, следует ввести дополнительные фильтры в прием для блокировки его срабатываний в ситуациях, аналогичных встретившимся в этой задаче.

Команда x10 = "нормдроби" (клавиша Ctr-ь) используется для коррекции величин вхождений лог.символов в теоремы базы приемов. Эти величины хранятся в логических терминалах "мощность" корневых указателей-списков статей логических символов базы приемов. Они используются при автоматическом выборе логического символа привязки в приемах. Если в силу каких-то ошибок величина в терминале "мощность" становится отрицательной, то компилятор ГЕНОЛОГа перестает реагировать на явное указание выбора другого логического символа привязки и выбирает символ с отрицательной величиной. В таких случаях и рекомендуется применять данную команду. Сначала все величины в терминалах "мощность" заменяются на нули, а затем просматривается вся база приемов и определяются суммарные количества вхождений логических символов в теоремы приемов.

Команда x10 = "извлечениеварианта" (клавиша "т") позволяет просмотреть все

теоремы ветви текущего пункта текущего меню оглавления базы приемов, упорядоченные в виде линейного списка. По любому элементу этого списка можно перейти к полному просмотру соответствующего приема нажатием клавиши "курсор вправо", а затем вернуться в список нажатием "курсор влево". Эта же команда позволяет просматривать все теоремы ветви текущего пункта текущего меню базы теорем. Клавиши "курсор вправо - влево" здесь используются аналогичным образом.

Команда x10 = "частичныйответ" (клавиша "о", кир.) позволяет вернуться в оглавлениях базы приемов, теорем и задачника из просмотра буфера к той точки оглавления, от которой до этого (через нажатие "ф") был осуществлен переход к буферу.

Команда x10 = "переобозначение" (клавиша Ctr-п) используется для входа в интерфейс создания оболочки нового пакетного оператора ГЕНОЛОГа. Этот интерфейс реализуется с помощью диалоговых блоков решателя. Он обеспечивает синтез приемов справочников, определяющих формат пакетного оператора, а также простейших приемов самого оператора, которые становятся основой для последующего заполнения пакета реальным содержанием.

Команда x10 = "анализатор" (клавиша "п") позволяет просматривать и редактировать текстовые примечания к текущему меню оглавления. Такие примечания сохраняются в текстовом терминале, достижимом из указателя-списка x4 по метке "примечание". Следует заметить, что эта возможность пока практически не используется, так как обычно тексты пунктов оглавления уже несут всю необходимую сопровождающую информацию.

Команды x10 = "область" (клавиша "Л") и x10 = "минимум" (клавиша "л") осуществляют запуск в разделе оглавления базы теорем серийного логического вывода с целью тестирования приемов вывода. Первая из них ограничивает вывод только теми случаями, когда применяется заданный, заранее выбранный прием.

Команда x10 = "внутрвывод" (клавиша "у") позволяет просмотреть все теоремы в заданном разделе базы теорем, для которых на последнем цикле тестирования приемов вывода не удалось получить их вывод из связанных с ними теорем-источников. Для просмотра используется стандартная конструкция - создается комментарий (подборнеизвестных ...), и далее каждое нажатие клавиши "ш" переводит к просмотру очередной теоремы.

Команда x10 = "нижняяоценка" (клавиша "х") позволяет последовательно просмотреть все приемы в ветви текущего меню базы приемов, имеющие заданный тип. Тип приема определяется ветвью оглавления приемов, связанной со справочником "заголовокприема" генератора приемов. После нажатия клавиши "х" возникает эта ветвь оглавления, и при выходе на ее концевой пункт (каждый такой пункт задает тип приема) составляется список просматриваемых приемов. Здесь снова применяется комментарий (подборнеизвестных ...); для перехода к очередному приему списка нажимается "ш". Та же команда x10 = "нижняяоценка" в случае базы теорем позволяет (аналогичным образом) просмотреть все теоремы в ветви текущего меню, имеющие комментарий заданного типа.

Команда x10 = "подобл" (клавиша "ч") позволяет просмотреть информацию об автоматическом построении чертежей в задачах текущего раздела задачника.

Приведенный список дополнительных операций, выполняемых через интерфейс оглавлений, будет пополняться по мере развития логической системы. Фактически, оглавления превратились здесь из средства древовидной организации данных, необходимого для быстрого поиска, в мощное средство интерактивного обучения, позво-

ляющее запускать различные циклы групповой обработки этих данных и анализировать полученные результаты.

13.3 Программа формульного редактора

Необходимость пополнять формульный редактор при обучении решателя возникает постоянно. Для упрощения этого процесса большинство логических символов прорисовываются формульным редактором в виде тех же словообразований, которыми они задаются в словаре. Чтобы такая прорисовка стала возможной, достаточно ввести (в виде псевдотеоремы приема) на ГЕНОЛОГе формат прорисовки символа и создать по нему серию приемов справочников, к которым обращается формульный редактор. В этом формате, в частности, можно определить кодовые комбинации клавиш (обычно - две последовательно нажимаемые клавиши) для ускоренного ввода символа. Процедура создания приемов справочников формульного редактора автоматизирована. Тем не менее, в особых случаях такая простейшая форма прорисовки может оказаться нежелательной, и тогда для расширения формульного редактора понадобится представление о его внутреннем устройстве.

Как уже говорилось выше, действия формульного редактора реализуются оператором "формредактор(x1 x2 ... x10)". У него x1-x4 задают координаты рамки редактирования (столбец-строка левого верхнего угла и солбец-строка правого нижнего угла); x5 - цвет фона. Редактор работает в двух режимах: ручной набор формулы (терма) и создание по заданному терму такой структуры данных, которая позволяет прорисовать его на экране в стандартной математической записи (точнее говоря, почти-стандартной). В первом режиме при обращении к оператору переменной x6 присваивается цвет символов, во втором режиме - тот терм, для которого нужно выполнить прорисовку. Переменные x7-x10 являются выходными переменными оператора. Переменной x7 присваивается корень так называемого дерева указателей - структуры данных для прорисовки терма в стандартной записи. Переменной x8 присваивается терм, полученный при ручном вводе. Переменные x9,x10 используются для уточнения размеров области прорисовки: x9 - правый столбец собственной рамки набранной формулы, x10 - пара (верхняя строка рамки - нижняя строка рамки).

Программа формульного редактора использует ряд вспомогательных процедур, реализованных, ради ускорения вычислений, непосредственно на СИ. Эти процедуры собраны в оператор ЛОСа "формблок(N...)", где N - символьный номер реализуемой процедуры. В первых версиях формульного редактора они программировались непосредственно на ЛОСе.

13.3.1 Структура данных, используемая для прорисовки формул в стандартной математической записи

Для прорисовки формулы на экране создается древовидная структура данных, воспроизводящая (быть может, с некоторыми отклонениями) древовидное строение формулы. Эта структура данных называется деревом указателей, а элементы ее (вершины) - указателями формульного редактора. Для ссылки на дерево указателей используется его корень. Каждый указатель определяет прорисовку части формулы (как правило, отдельной операции, предиката, квантора и т.п.) и хранит необходимую для этого геометрическую информацию. Он представляет собой набор ($A_1 \ldots A_7$), элементы которого несут следующую нагрузку:

1. A_1 - символьный номер, называемый типом указателя. При инициализации указателя он иногда получает тип 0, который впоследствии заменяется на ненулевое значение. Тип 1 имеют указатели большинства операций и отношений, прорисовываемых "префиксным образом" как $f(t_1 \dots t_m)$. Тип 2 имеют указатели "инфиксных записей" - $(t_1 f t_2 f \dots f t_n)$. Тип 3 - указатели одноместных отношений, прорисовываемые с помощью тире (t – число, t – функция, и т.п.). Тип 4 получили указатели для прорисовки выражений "величина(...)" (десятичные числа, отличные от цифр), "степень(...)", "связприставка(...)". Последняя конструкция вводится самим формульным редактором - при анализе кванторов и описателей переменные связывающей приставки группируются под символом "связприставка". Тип 5 имеют указатели, используемые при прорисовке модуля. Тип 6 - указатели для прорисовки переменных; тип 7 - указатели функциональных переменных, и тип 8 - указатели матриц.

2. A_2 - логический символ, являющийся заголовком указателя и обычно совпадающий с названием соответствующей операции (отношения, и т.п.) в прорисовываемой формуле. Если $A_1 = 6$, то A_2 есть код клавиатуры, соответствующий переменной.

3. A_3, A_4 - координаты "начала" подтерма, определяемого ветвью указателя. A_3 - столбец; A_4 - строка. Предполагается, что у любого подтерма выделена его "главная" текстовая полоса (высотой в 19 пикселей), на которой (кроме ряда исключений, таких, как матрицы) прорисована корневая операция. A_4 есть верхняя строка этой полосы. Этот принцип ссылки на верхнюю строку текстовой полосы является в формульном редакторе стандартным.

4. A_5 - информация о внешней скобке подтерма, определяемого ветвью указателя. Если $A_5 = 0$, то такой скобки нет. При наличии скобки A_5 представляет собой пару (h, l), у которой h - высота части скобки, выступающей вверх над 19-пиксельным вертикальным отрезком главной полосы подтерма. Верхняя и нижняя части скобки строго симметричны по отношению к этой 19-пиксельной полосе. Если скобка еще не закрыта (в процессе набора фрмулы), то $l = 0$. Иначе l есть пара (столбец - строка), определяющая координаты закрывающей скобки. Под строкой здесь понимается номер верхней строки той текстовой полосы, на уровне которой должна быть размещена закрывающая скобка.

5. A_6 - специальная информация для прорисовки(см. ниже).

6. A_7 - набор ссылок на подуказатели данного указателя. Обычно они соответствуют операндам операции (отношения, и т.п.) данного указателя и размещены в том же порядке.

Информационный элемент A_6 используется следующим образом:

1. Если $A_1 = 1$, то A_6 есть пара $(P_1\ P_2)$, где P_1 - координата (столбец - строка) текста заголовка указателя без учета внешней скобки; P_2 - либо 0, либо содержит дополнительную информацию для прорисовки сложных операций (дроби, радикалы, интегралы и т.п.).

   При A_2 = "дробь" значением P_2 служит пара (длина дробной горизонтальной черты в пикселях - указатель U на завершение набора дроби). По завершении набора дроби U становится равно символу "пустоеслово".

При A_2 = "корень" (что соответствует случаю радикала степени от 3 до 9) либо A_2 = "квадркорень" (для квадратного радикала) значением P_2 служит четверка (B_1 B_2 B_3 B_4). Здесь B_1 - смещение вверх от текущей строки (т.е. от верхней строки текущей 19-пиксельной полосы) к верхней строке радикала; B_2 - смещение от текущей строки до нижней строки радикала; B_3 - указатель на завершение набора радикала (1 - набор завершен, 0 - не завершен); B_4 - длина верхней горизонтальной линии радикала. Под завершением набора в случае дробей и радикалов понимается ситуация, когда цвет прорисовки их линий становится черным.

При A_2 = "суммавсех" либо "произведениевсех" P_2 представляет собой тройку (смещение вверх от текущей строки до верхней линии знака суммы или произведения - длина верхней горизонтальной линии знака - указатель завершения набора). Знак прорисовывается симметричным относительно текущей 19-пиксельной полосы образом. Указателем завершения набора служит 1 (до завершения - 0).

При A_2 = "интеграл" либо "факториал" либо "числосочетаний" элемент P_2 представляет собой пару (смещение вверх от текущей строки до верхнего края знака - указатель завершения набора, такой же, как для "суммавсех").

2. Если $A_1 = 2$ (инфиксная операция), то значением A_6 служит набор $(0, (B_{21}, B_{22}), \ldots, (B_{n1}B_{n2}))$, где n - число операндов; (B_{i1}, B_{i2}) - координата позиции символа операции A_2 перед i - м операндом.

3. Если $A_1 = 3$, то A_6 есть координата (столбец-строка) начала текста, идущего после тире.

4. Если $A_1 = 4$ и A_2 - символ "степень", то A_6 есть индикатор завершения набора степени (он становится равен символу "пустоеслово", когда набор завершен).

5. Если $A_1 = 6$ и вводится переменная с индексом, то A_6 служит для указания на завершение набора индекса (равно "пустоеслово", когда индекс набран).

6. Если $A_1 = 8$ и A_2 есть логический символ "элементы", то значением A_6 служит набор $(B_1 \ldots B_{n-1}C)$. Здесь $B_1, \ldots B_{n-1}$ - смещения по горизонтали от левого края матрицы до начала прорисовки элементов 2-го,3-го, ... , n - го столбцов; C - смещение до закрывающей скобки матрицы. Данный указатель соответствует некоторой строке матрицы; сама матрица представлена указателем типа 8 с заголовком A_2 = "Набор", у которого $A_6 = 0$. Указатели для строк - подуказатели последнего указателя.

13.3.2 Начало программы формульного редактора

Для выхода в программу формульного редактора можно использовать пункт "Вспомогательные процедуры интерфейсов - Формульный редактор - Основной блок формульного редактора" оглавления программ.

Начинается программа формульного редактора с рассмотрения того специального случая, когда нужно прорисовать уже имеющуюся формулу, содержащую матрицы. В этом случае сначала предпринимается попытка получить (в размерах отведенного прямоугольника) изображение формулы с обычной прорисовкой матрицы, а

если это не удается, то предпринимается дополнительная попытка получить изображение матрицы в виде "строки($(a_{11} \dots a_{1n}) \dots (a_{m1} \dots a_{mn})$)".

В отсутствии указанного выше особого случая - переход через "ветвь 2". Оператор "русшрифт(0)"переключает здесь драйвер клавиатуры на режим ввода латинских букв (по выходе из формульного редактора будет восстановлен режим кирилицы). Далее блокируется попытка прорисовать однобуквенный терм служебного характера (например, отдельно взятый заголовок какой-либо обычной математической операции, квантора и т.п.). Такие попытки, возможные при работе отладчика, могут привести к выдаче сообщений об ошибках в работе программы. Далее - переход через "ветвь 1".

Если имеет место режим ручного ввода формулы (x6 - логический символ), то предпринимается расчистка экрана. Инициализируются переменные x11,x12, которые указывают координаты текущей позиции курсора формульного редактора (столбец - строка). Инициализируется стэк x14 указателей формульного редактора. Последний элемент набора x14 соответствует указателю для всей формулы, предпоследний - его подуказатель, и т.д. вплоть до первого элемента, который соответствует текущему вводимому подтерму. При инициализации в x14 заносится заготовка корневого указателя, в котором все позиции, кроме позициий текущих столбца и строки, заполнены нулями и символами "пустоеслово". Затем идет оператор "повторение", к которому будут происходить откаты при вводе очередной команды (только для ручного набора). При ручном режиме ввода прорисовывается курсор формульного редактора и происходит обращение к процедуре "формклавиатура(x15)" для запроса очередной команды x15. В режиме формирования дерева указателей для уже известной формулы - полагается $x15 = 0$.

Процедура "формклавиатура" видоизменяет последовательность сигналов, поступающих с клавиатуры, в соответствии с некоторыми правилами. Она имеет "память", роль которой выполняют комментарии (формклавиатура ...) и (кортеж ...) к посылкам исходной задачи. Используя эту память, процедура в определенных случаях выдает результат вообще без нажатия клавиш; в то же время, в ряде случаев для выдачи результата может понадобиться несколько нажатий клавиш. Преобразования сигналов с клавиатуры происходят в соответствии со следующими соглашениями:

а) Если имеется комментарий (кортеж A_1 A_2) к посылкам исходной задачи, то перед каждой вводимой буквой, кроме первой, вставляется символ умножения (кодируемый логическим символом "спуск"). A_1 - указатель первой буквы, вначале равный 0, а после ввода первой буквы заменяемый на 1. A_2 - буфер, в котором сохраняется очередной буквенный символ, в то время как вместо него выдается "умножение". При следующем обращении результат будет извлечен сразу из этого буфера, а в буфер будет занесен ноль. Такое преобразование используется при вводе обозначений типа $\triangle(ABC)$, когда несколько буквенных символов A, B, C должны вводиться подряд без символов разделителей (оператор "формклавиатура" автоматически вводит здесь в качестве разделителей символы умножения, которые при определении результирующего терма будут отброшены). В прочих ситуациях формульный редактор блокирует ввод двух букв подряд либо (см.ниже) рассматривает эти буквы как кодовую комбинацию, преобразуемую в отдельный символ.

б) Вводимые с клавиатуры буквенные символы, а также специальные сигналы - пробел, слэш, звездочка, сохраняются в буфере A_1 комментария (формклавиатура A_1 A_2). Буфер A_2 этого же комментария используется для немедленной выдачи хранящихся в нем символов. Чтобы распознавать кодовые комбинации буквенных клавиш, служит справочник "префикс", которому передается содержимое буфера

A_1, причем символом обращения к справочнику служит первый элемент буфера A_1. Если он выдает переменную, то заполнение буфера продолжается; если 0 - буфер сбрасывается; если выдается другой логический символ, то он рассматривается как результат распознавания кодовой комбинации клавиш. Таким образом, могут выдаваться не только коды клавиш, но также множество дополнительных кодов для последующей обработки формульным редактором.

в) Если идет процесс накопления информации о кодовой комбинации клавиш, то при поступлении очередной буквы оператор "формклавиатура" выдает сигнал об откате (логический символ "8"), занося поступившую букву в буфер A_2. При следующем обращении она будет оттуда извлечена в качестве результата.

г) Для обработки символа, поступившего после нажатия клавиши "слэш", используется справочник "суффикс", определяющий фактически выдаваемый результат.

д) Если кодовая комбинация клавиш начинается с символа "пробел", то для ее распознавания применяется не справочник "префикс", а справочник "формклавиатура".

е) Двойное нажатие клавиши "звездочка" воспринимается как кодовая комбинация, приводящая к прорисовке умножения в виде точки (при однократном нажатии применяется обычное слитное расположение сомножителей).

После получения команды x15 прежде всего проверяется, не является ли она символом закрывающей скобки (символ "учеткоманды"), и если является, то сбрасывается упомянутый выше комментарий (кортеж ...).

13.3.3 Обработчик команд формульного редактора

Перечислим процедуры обработки специальных команд x15 формульного редактора; обработка общего случая - команд ввода символов формулы - рассматривается в следующем подразделе.

1. x15 = "Плюс" (клавиша Esc). В этом случае убирается курсор (закрашивается цветом фона); драйвер клавиатуры переключается на режим кириллицы, удаляются комментарии (кортеж ...) и (формклавиатура ...), используемые оператором "формклавиатура", и выход из формульного редактора по истинностному значению "ложь".

2. x15 = "8" (клавиша Backspace). В этом случае реализуется откат для последнего введенного символа набираемой формулы. Прежде всего здесь убирается курсор. Если начало стэка указателей x14 (т.е. текущий заполняемый указатель) уже имеет определенный тип, то переход через "иначе 2". В противном случае проверяется, что длина стэка больше 1 (если она равна единице, то набор формулы еще не начат и откат не выполняется) и находится второй элемент x16 стэка указателей. Текущий указатель является его подуказателем. Далее предпринимается разбор случаев по типу указателя x16:

   а) x16 - указатель типа 1. Здесь рассматриваются два подслучая:

   а1) Число подуказателей указателя x16 равно 1 . В этом случае единственный подуказатель - стоящая перед x16 "пустая" заготовка, так что при откате нужно удалить саму операцию, определяемую указателем x16. Сначала предпринимается обращение к оператору "блокредактора(десять ...)", который стирает данную операцию на экране, предпринимая при необходимости также перерисовку ряда внешних операций. Например, если при откате укорачивается

числитель или знаменатель дроби, так что дробная черта становится избыточно длинной, то она тоже укорачивается; аналогичным образом корректируется прорисовка знаков радикалов, суммирования, и т.п. Затем курсор прорисовывается на новой позиции, соответствующей указателю x16. В первую ячейку указателя x16 заносится 0 (его тип сбрасывается); в последнюю (список подуказателей) - "пустоеслово". Корректируются координаты x11, x12 текущей позиции курсора. После этого особо рассматривается случай, когда исключенная операция являлась минусом перед очередным слагаемым (переход по "иначе 5"). Если это не имело места, то информационный (6-й) элемент указателя x16 заменяется на "пустоеслово"; удаляется, если он был, комментарий (кортеж ...); отбрасывается первый элемент стэка указателей (расположенный перед x16), и откат завершен. В противном случае - проверяется, равнялось ли число слагаемых двум. Если равнялось, то находится надуказатель суммы, и ссылка из него на сумму заменяется ссылкой на первое (оставшееся после отката) слагаемое, а указатель суммы отбрасывается. Если число слагаемых больше двух, то сумма остается, хотя ссылка из нее на подуказатель x16 исключается. В обоих случаях в начале стэка указателей располагается указатель последнего из оставшихся слагаемых суммы. Если этот указатель имеет подуказатели, то к началу стэка может быть добавлен последний из них, и т.д. до получения неудлиняемой далее цепочки указателей. Такое удлинение стэка указателей осуществляется оператором "блокредактора(одиннадцать ...)".

а2) Число подуказателей указателя x16 больше одного. Определяется заголовок x17 указателя x16. Отбрасывается последний подуказатель этого указателя (т.е. расположенный перед ним в стэке x14 недоопределенный указатель) и в начало стэка помещается последний из оставшихся подуказателей указателя x16. В зависимости от типа x17 (дробь, радикал с указателем степени, интеграл, и т.п.) находится новая позиция курсора при откате (например, в случае дроби происходит возвращение курсора из знаменателя в числитель). Чтобы вернуться к доопределению "последнего" фрагмента в подтерме, определяемом новым началом U стэка x14, используются обращения к оператору "блокредактора(двенадцать ...)", добавляющему в начале стэка x14 цепочку подуказателей указателя U.

б) Либо тип указателя x16 равен 5, либо этот указатель имеет заголовок "перечень", а число подуказателей равно 1. Эти ситуации возникают, если была введена первая вертикальная черта модуля либо левая фигурная скобка конечного списка. Тогда выполняются стирание с экрана символа указателя x16, прорисовка вместо него курсора, замена типа указателя x16 на 0, замена на "пустоеслово" списка подуказателей и информационного элемента, удаление комментария (кортеж ...) и отбрасывание в стэке указателей элемента, идущего перед x16.

в) Тип указателя x16 равен 2. Тогда должна быть стерта последняя введенная инфиксная операция. Находится символ x17 этой операции. В случае заголовка указателя "набор" либо "перечень" он равен запятой, иначе - совпадает с заголовком. Оператор "блокредактора(десять ...)" стирает символ x17. Координаты этого символа извлекаются из информационного элемента указателя (они расположены в конце данного элемента). Затем курсор устанавливается на позицию стертого символа. Если число подуказателей указателя x16 было равно 2 (первый из них соответствует полностью введенному операнду, второй

- расположенной перед x16 заготовке указателя), то вместо x16 помещается его первый подуказатель; если же число подуказателей было более двух, то последний из них удаляется и одновременно удаляется последняя координатная пара из информационного элемента указателя x16. В обоих случаях отбрасывается указатель, располагавшийся перед x16, и применяется процедура "блокредактора(
одиннадцать ...)" для удлинения стэка указателей x14.

г) Тип указателя равен 4 либо 6. Эти ситуации возникают в двух случаях: либо набиралась переменная с индексом, и имеет место откат к уровню переменной после отказа от индекса (тип 6), либо набиралась степень и имеет место возвращение к уровню главной строки от уровня показателя степени. Соответственно, в первом случае номер x12 текущей строки уменьшается, а во втором случае - увеличивается на 9. По скорректированной текущей позиции прорисовывается курсор. Если x16 не был корневым указателем набираемого терма, то он заменяется на свой первый подуказатель; соответственно корректируется стэк x14.

д) Тип указателя равен 8. В этом случае происходит откат в строке набираемой матрицы к предыдущему элементу (если текущий не введен либо стерт предыдущим откатом). Если список x17 подуказателей указателя x16 более чем одноэлементный, то текущая строка непуста. В этом случае курсор переводится на позицию, непосредственно следующую после последнего элемента строки. Если же x17 одноэлементый, то текущая строка пуста, и тогда нужно вернуться к предыдущей строке. Случай одноэлементного x19 означает, что рассматриваемая строка вообще была первой. Если это не так, то корректируются размеры левой скобки матрицы (правая пока не прорисована), и курсор возвращается в конец предыдущей строки.

е) Тип указателя x16 равен 0, причем в нем указана ссылка на открывающую скобку. Это означает, что после скобки нет каких- либо записей, и при откате нужно ее удалить. Особо рассматриваются случай, когда такая скобка стояла после квантора либо описателя (осуществляется откат к продолжению прорисовки связывающей приставки - см. действия после "заголовок(x17 1)"), и случай, когда открывающая скобка стояла после буквенного символа, в результате чего появилась заготовка указателя для терма вида "значение(f ...)". Эта заготовка удаляется, и происходит возвращение к обычному указателю переменной f.

Перейдем теперь к рассмотрению случая, когда первый указатель стэка x14 имел тип, отличный от нуля. Переход к соответствующему фрагменту программы - через оператор "иначе 2" в начале ветви обработки команды отката. Отличие от нуля типа первого указателя стэка x14 означает, что он определяет некоторый полностью набранный терм T. При откате следует вернуться к указателю того подтерма терма T, который обрабатывался на последнем шаге набора T. Чтобы найти данный указатель, реализуется цикл, начинающийся с оператора "повторение" - к нему будут происходить возвращения для выполнения очередного перехода вглубь терма T.

Прежде всего, обрабатывается общий случай - первый указатель x16 стэка x14 имеет тип 1, 2 либо 7, с ним не связана скобка, и заголовок его не является одним из символов, требующих перекраски при завершении набора (т.е. не является дробью, радикалом, символом суммирования или интегрирования и т.п.). Если

этот указатель имеет подуказатели, то последний из них заносится в начало стэка x14, и откат к "повторению".

Если указанный выше общий случай не имеет места, то переход через оператор "ветвь 1". Прежде всего, здесь проверяется, не связаны ли с указателем внешние скобки. Если они есть, то переход через оператор "иначе 2". Здесь происходит стирание закрывающей скобки (в специальных случаях - при заголовках "модуль" и "перечень" указателя x16 - вместо закрывающей круглой скобки берутся вертикальная черта либо закрывающая фигурная скобка). Курсор переносится на место стертой скобки. Если была стерта круглая скобка, то между x16 и его надуказателем вводится промежуточный указатель x19 типа 0, в который из x16 переносятся данные о внешних скобках (в самом x16 они исключаются). Указатель x19 - временный; после закрытия скобки он будет устранен, и данные о внешних скобках снова окажутся в x16. Во всех случаях после стирания скобки предпринимается стандартное удлинение стэка указателей с помощью "блокредактора(двенадцать ...)" (в начало стэка выносится указатель наименьшего подтерма, который будет доопределяться при наборе), и выполнение команды отката завершается.

Если с указателем x16 не были связаны внешние скобки, то прежде всего рассматривается случай указателя типа 3. Здесь стирается текст указателя, располагавшийся после тире, курсор переносится на начало этого текста, указатель x16 заменяется на свой подуказатель, оператор "блокредактора(одиннадцать ...)"выполняет удлинение стэка x14, и выполнение команды отката завершается.

Далее рассматривается случай указателя x16 типа 1, заголовок которого требует перекраски по завершении набора (дробь, радикал, интеграл, и т.п.). В этом случае восстанавливается малиновый цвет заголовка, в случае дроби отменяется центровка числителя и знаменателя, курсор переносится на позицию, с которой будет происходить доопределение терма, и выполняются необходимые коррекции стэка x14.

Оставшиеся случаи, рассматриваемые в той же цепи фрагментов программы, таковы: сначала - откат после набора степени либо переменной с индексом (корректируется высота курсора); затем - откат к набору более чем односимвольного числа либо связывающей приставки; затем - откат после набора переменной, цифры либо символа константы.

3. x15 = "стандменьше" (клавиша F1). Здесь выполняется обращение к справочнику по логической системе. Предварительно осуществляется перевод драйвера клавиатуры на режим кириллицы (чтобы правильно воспринимались обычные команды интерфейсов). Текущее содержимое экрана сохраняется в буфере (оператор "видео(минус)"). После обращения к справочнику - восстановление исходного экрана и переход в режим латиницы.

4. x15 = "символтеоремы" (клавиша "Ctr-курсор вниз"). При наборе формулы вручную происходит переход на новую строку. Для этого применяется процедура "выводсимвола", которой передается информация о фиктивном символе "развязка". Длина этого символа указана достаточно большой, чтобы процедура "выводсимвола" начала свою работу с перехода к новой строке. Фактический вывод этого символа блокируется.

5. x15 = "фигуры" (клавиша Ctr-Enter). Происходит набор логического символа с помощью текстового редактора; после этого x15 заменяется на набранный символ, и продолжается обработка скорректированного значения x15 как очередного символа набираемой формулы. Этот способ применяется для редко встречающихся символов, когда ввод специальной кодовой комбинации клавиш нецелесообразен. Тем не менее, здесь все же нужно, чтобы с символом заранее были связаны необходимые программы справочников формульного редактора (справочники "формредактор", "блокредактора", и др.).

6. x15 = "внешдизъюнкция" (клавиша Insert). Здесь происходит вставка с текущей позиции набираемой формулы некоторого подтерма, который был заранее занесен в буфер. Роль такого буфера играют комментарии (смполоса ...), (Смполоса ...) к посылкам исходной задачи. Первый из них вводится процедурой просмотра задачника, второй - процедурой просмотра цепи задач при пошаговой трассировке решения. В этих комментариях регистрируются выделенные элементы задач. Предпринимается обращение к клавиатуре для ввода цифры от 1 до 9, находится выделенный подтерм T с соответствующим номером, и переменная x13 заменяется на терм "набор(T)". Эта переменная была инициализирована нулем (одновременно с инициализацией стэка указателей x14),и ее ненулевое значение определяет временное переключение (см. ниже) с режима ручного ввода на режим автоматического ввода терма T; по окончании снова восстановится ручной режим.

13.3.4 Обработка очередного символа набираемой формулы

Если полученная формульным редактором команда x15 не обрабатывается одним из перечисленных выше фрагментов, то она либо есть 0 (в случае создания формульным редактором дерева указателей для уже известного терма), либо представляет собой логический символ, кодирующий очередной элемент набираемой вручную формулы (операцию, переменную, скобку и т.п.). Обработка такого x15 начинается в фрагменте, соответствующем пункту "Вспомогательные процедуры интерфейсов - Формульный редактор - Основной блок формульного редактора - Обработка очередного входного набора" оглавления программ. Первый оператор "альтернатива(логсимвол(x6)...)" этого фрагмента в случае ручного набора (x6 - логический символ) проверяет, равно ли x13 нулю. Если равно, то находится результат x16 обращения к справочнику "формредактор" на символе x15. Этот результат - либо 0 (в случае ввода переменной), либо четверка (i, r, p, s) характеристик графического изображения кодируемого посредством x15 символа s. i - тип символа, равный 1 для префиксных операций, 2 - для инфиксных (включая записи типа "a-число"); 3 - для степени; 4 - для "Enter" и "курсор вниз"; 5 - для скобок; 6 - для цифр и точки; 7 - для модуля, целой части, перечня и набираемой в стандартном виде матрицы; 8 - для константных символов ("e","пи","пусто" и т.п.). r - длина (символьный номер) записи символа s в 8-пиксельных единицах; p - зависящая от символа вспомогательная информация (для инфиксных операций - указатель их относительной "силы" при расстановке скобок; равенство p символу "пустоеслово" означает, что s имеет нестандартные размеры по вертикали). Если x13 не равно 0, то оно имеет вид "набор(T)", и тогда включается режим автоматического набора терма T. В этом режиме перечисляются четверки x16, эмулирующие ручной набор T. Начинается перечисление с набора (5 1 0 операнд), обозначающего открывающую скобку, затем идут собственно наборы для T,

и в конце - набор (5 1 0 последнийоперанд) для закрывающей скобки. По окончании перечисления - переход через "иначе 1", где значение x13 изменяется на 0, и откат ко вводу очередной команды для ручного набора формулы. Наконец, в случае режима создания дерева указателей для известного терма x6, происходит обращение к оператору "модульредактора(левыйкрай(x6) x16)", перечисляющему четверки x16, эмулирующие ручной ввод терма x6.

Таким образом, как в случае ручного набора терма, так и в случае создания дерева указателей для известного терма, дальнейшие действия по обработке наборов x16 почти совпадают. Отличие возникает лишь на этапе завершающей обработки построенного дерева указателей - в первом случае по нему будет определяться терм, а во втором случае терм уже известен.

После получения значения x16 прежде всего проверяется, не определяет ли оно операцию, между операндами которой при наборе должны вставляться фиктивные символы умножения; если определяет, то вводится уже упоминавшийся выше комментарий (кортеж 0 0) к посылкам исходной задачи, используемый оператором "формклавиатура" в качестве буфера. Если значение x16 равнялось 1 (при эмуляции набора терма стало ясно, что его не следует изображать формульным редактором), то попытка построить дерево указателей обрывается, и выход из формульного редактора по значению "ложь".

Далее идет оператор "или(не(заголовок(начало(x14)1))...)", который усматривает завершенный набор числа, укорачивая при этом стэк указателей x14 таким образом, чтобы в начале был размещен указатель для максимального терма, набор которого уже завершен. После этого оператора начинается разбор случаев для x16.

Прежде всего, рассматривается случай, когда x16 соответствует префиксной операции либо символу "минус", не расположенному под знаком суммы - см. оператор "или(заголовок(x16 1)...)". Здесь выполняется обращение к процедуре "выводсимвола(...)", которая осуществляет фактическую прорисовку введенного символа операции, корректируя при необходимости размеры ранее введенных элементов изображения - как в дереве указателей, так и на экране. Если для размещения символа не хватило места (переход через "иначе 5"), то в случае обработки уже известного терма предпринимается попытка переформулировать его так, чтобы исключить не помещающиеся на одной строке числители и знаменатели дробей. Подозрительные на такое переполнение подтермы "дробь(...)" заменяются на подтермы "деление(...)", при прорисовке которых вместо горизонтальной черты используется двоеточие. После замены - откат к повторной обработке терма.

Если места для прорисовки символа оказалось достаточно, то предпринимаются обращения к справочникам "формблок" (определение вертикального смещения курсора после прорисовки символа), "терм" (получение дополнительной информации для прорисовки специальных символов), "пусто" (определение случаев, не требующих сдвига курсора по горизонтали после прорисовки символа). Далее происходит обращение к оператору "формблок(5 ...)", реализация которого вынесена в программу интерпретатора ЛОСа. Он доопределяет первый указатель стэка указателей x14 и вводит его подуказатель, присоединяя его к началу стэка. Новая версия стэка - значение выходной переменной x22 данного оператора. Затем корректируется позиция курсора, и откат к получению очередной четверки x16.

Для рассмотрения следующего подслучая перейдем через "иначе 4" к фрагменту, начинающемуся с оператора "заголовок(x16 2)". Это - случай инфиксной операции. Ветвь, расположенная после оператора "повторение", обеспечивает усмотрение завершенного ввода операнда, если новая (внешняя) инфиксная операция оказалась

"слабее", чем старая (внутренняя) инфиксная операция, являющаяся заголовком этого операнда. Соответственно укорачивается стэк x14. Для сравнения инфиксных операций по их "силе" используется информационный элемент p набора x16. Чтобы определить соответствующее значение для внутренней инфиксной операции, используется справочник "блокредактора", аналогичный справочнику "формредактор", но выдающий не четверку (i, r, p, s), а тройку (i, r, p) и получающий в качестве входного символа не код клавиатуры, а символ s. Этот же справочник используется и при эмуляции оператором "модульредактора" набора терма, так как здесь изначально известным оказывается не код клавиатуры, а логический символ s.

После коррекции стэка x14 - переход через оператор "ветвь 3", где прежде всего проверяется, не была ли введена операция умножения после ввода выражения, начинающегося с минуса. Если это так, то указатель операнда при минусе выносится в начало стэка x14, чтобы в итоге минус оказался отнесенным ко всему произведению. Далее используется оператор "формблок(4 ...)", который определяет скорректированный стэк указателей (в начало его заносится бланк указателя для ввода очередного операнда). Заметим, что программа оператор "формблок(4 ...)" содержит список инфиксных операций f, прорисовываемых в виде "$a-f$". Этот список должен пополняться при вводе новых таких операций, с перекомпиляцией интерпретатора ЛОСа.

При обработке случая "заголовок(x16 2)" особо рассматривается подслучай ввода символа "точка с запятой" после ввода фигурной скобки для конечного списка. Это - условное обозначение для конечного списка, определяемого по набору элементов, заданному неявно.

Если первый элемент четверки x16 равен 3, то была нажата клавиша "курсор вверх" для перехода к набору показателя степени. В этом случае проверяется, что основание степени набрано полностью (либо тип начального указателя стэка x14 отличен от 0, либо этот указатель соответствует паре внешних скобок, в которые заключено основание степени). Затем - обращение к оператору "выводсимвола", которое приводит к перенесению курсора на уровень показателя степени, быть может, сопровождаемому коррекцией прочих параметров изображения формулы. Если основание степени начиналось с минуса и не было заключено в скобки, то в начало стэка x14 добавляется указатель, соответствующий выражению под минусом. Этим достигается то, что минус будет отнесен ко всей степени. Затем вводятся указатели x19 (бланк для набора показателя степени) и x20 (собственно указатель для степени, имеющий своим подуказателем первый указатель стэка). x19 и x20 заносятся в начало стэка x14, из которого при этом исключается первый элемент (учтенный в x20). Затем - откат к получению очередного набора x16.

Если x16 начинается с символа 4, то была нажата "управляющая клавиша" формульного редактора. Здесь рассматриваются следующие случаи:

а) набору x16 соответствует логический символ "спуск" (клавиша "курсор вниз"). Для ввода индекса переменной курсор переводится вниз; создается бланк x19 подуказателя текущего указателя переменной, в котором будет формироваться индекс; x19 заносится в начало стэка x14, и переход к очередному набору x16.

б) набору x16 соответствует логический символ "сброс" (клавиша Enter). Здесь сгруппированы различные процедуры, обслуживающие завершение набора специального фрагмента терма, такие, как перенесение курсора из числителя в знаменатель, перенесение курсора при вводе пределов суммирования или интегрирования, перекраска в черный цвет линий полностью набранного фрагмента (в процессе набора цвет незавершенных дроби, радикала, интеграла и т.п. - малиновый), и др. Кроме того, здесь обрабатывается случай завершения набора всей формулы с выдачей ре-

зультата. Переходы к соответствующему фрагменту - через операторы "иначе 3", "ветвь 1". Для получения терма по дереву указателей служит оператор "блокредактора(8 конец(x14)x18)". Если этот оператор оказывается не в состоянии определить терм (ввод заведомо не завершен либо происходил с ошибками), то выход из формульного редактора блокируется, и происходит откат к вводу очередной команды. Иначе предпринимаются обращения к операторам "блокредактора(9 ...)" и "блокредактора(1 ...)", которые находят верхнюю и нижнюю строки области, занятой формулой, после чего выдается результат.

в) набору x16 соответствует логический символ "конец" (клавиша End) либо "следузел" (клавиша Page Down). Соответствующие процедуры связаны со вводом матрицы: в первом случае ее набор завершается и прорисовывается правая скобка матрицы; во втором случае - выполняется переход к следующей строке матрицы.

Если набор x16 начинается с символа 5, то была введена скобка. Находится последний элемент x17 набора x16 - символ, обозначающий тип скобки. Для открывающей скобки (x17 = "операнд") рассматриваются два случая: в скобки заключается новая операция (случай нулевого типа первого указателя x20 стэка x14), либо имеет место функциональная запись $F(\ldots)$, где выражение F определяется первым указателем стэка x14 (тогда тип этого указателя ненулевой). В первом случае наличие скобки регистрируется в указателе x20, и в начало стэка добавляется его подуказатель x21 - для ввода заключенного в скобки выражения. Во втором случае (переход через "иначе 3") - сначала находится минимальное осмысленное F в конце набранной части, которое и будет рассматриваться как обозначение функции. Затем создается указатель x23 с заголовком "значение", первый подуказатель которого определяет F, а второй есть новый указатель x22, содержащий информацию о скобке. Кроме того, для ввода аргумента функции в начало стэка x14 помещается "пустой" подуказатель x21 указателя x22.

Если набор x16 начинается с символа 6, то вводится цифра либо точка. Здесь различаются следующие случаи:

а) Вводится отдельно стоящая цифра. Тогда тип первого указателя x17 стэка x14 равен 0. Если цифра вводится в показателе радикала, то переход через "иначе 3", где указатель радикала x19 с заголовком "корень" получает, в дополнение к подуказателю x17, сохраняющему введенную степень радикала, также новый подуказатель x20 для ввода выражения под радикалом. Иначе - x17 доопределяется в соответствии со введенной цифрой; это делает оператор "формблок(десять ...)".

б) Продолжается перечисление цифр ранее инициализированного числа. Если ранее имелась единственная цифра, то для накопления числа вводится указатель x22 с заголовком "величина". Иначе - добавляется новый подуказатель ранее созданного указателя "величина", хранящий данные о введенной цифре либо точке.

Если набор x16 начинается с символа 7, то вводится модуль, либо целая часть, либо инициализируется матрица. Переход к инициализации матрицы - через "иначе 1"(в этом случае набор x16 заканчивается логическим символом "Набор"). Левая черта модуля либо левая скобка целой части вводится в ситуации, когда тип первого указателя стэка x14 равен 0; правая черта либо правая скобка - если этот тип не равен 0 (см. переход через "иначе 2").

Переход к ветви программы, в которой рассматриваются оставшиеся случаи для набора x16 - через оператор "иначе 3" исходного фрагмента обработчика очередного символа (т.е. фрагмента, в котором было определено значение x16). Здесь обрабатываются случаи, когда x16 равно 0, имеет заголовок 8 или заголовок 9. Случаи

x16 = 0 и заголовка 9 аналогичны и означают ввод переменной (при ручном наборе ввод переменной дает нулевое значение x16, при эмуляции набора известного терма оператор "модульредактора" создает для переменных четверки, начинающиеся с 9). Случай заголовка 8 означает ввод символа константы.

13.3.5 Прорисовка очередного символа и коррекция ранее введенной части изображения

Прорисовка очередного символа формулы выполняется оператором "выводсимвола". Он не только обеспечивает вывод символа на экран, но и осуществляет необходимую коррекцию параметров уже имеющейся части изображения, чтобы для нового символа хватило места. Эта коррекция предпринимается в дереве указателей и сопровождается перерисовкой той части формулы, которую она затрагивает. Таким образом обеспечивается автоматическое удлинение линий дробей и радикалов, смещение вверх или вниз формульной строки, на которой появляется "многоэтажная" конструкция (например, дробь),и т.п. Так как оператор "выводсимвола" необходим для создания дерева указателей, он применяется не только в режиме ручного ввода формулы, но и при автоматической обработке уже известной формулы.

Выход в начальный фрагмент программы оператора "выводсимвола" возможен либо через оглавление программ (см. соответствующий подпункт в разделе формульного редактора), либо непосредственно через набор заголовка этой программы в главном меню. Заметим, что стэк указателей оператора "формредактор" здесь является значением переменной x9. Напомним, что начало набора x9 есть текущий указатель, каждый следующий указатель является надуказателем предыдущего, а конец набора x9 - указатель, определяющий всю формулу. Прежде всего, программа определяет длину x14 выводимого символа в 8-пиксельных единицах. Находится число x16 8-пиксельных позиций, имеющихся в текущей строке после позиции курсора.

Далее рассматривается случай, когда на текущей строке места для прорисовки символа не хватает (см. участок после контрольной точки "прием(2)"). Инициализируется нулем переменная x18, и предпринимается просмотр стэка указателей x9 для поиска такого самого большого выражения, содержащего вводимый символ, которое вместе с этим символом должно помещаться на одной строке. Такие не допускающие переноса на новую строку выражения суть: дроби, радикалы, матрицы, логарифмы в случае набора первого операнда, а также целый ряд других - см. большой оператор "или(входит(x20 ...)". Переменной x18 присваивается вхождение в стэк x9 указателя для найденного выражения. Если любое надвыражение вводимого символа может быть перенесено через строку, то x18 остается равно 0. Затем - переход через "иначе 3".

Предпринимается обращение к оператору "блокредактора(1 ...)", определяющему нижнюю строку x19 той части формулы, которая получается при отбрасывании подформулы x18 (она будет перенесена на следующую строку). Затем с помощью оператора "блокредактора(9 ...)" находится верхняя строка подформулы x18, определяется смещение к этой строке от главной строки формулы x18, и x19 увеличивается на это смещение. Таким образом находится та строка x19, с небольшим отсуплением вниз от которой будет происходить прорисовка перенесенной части.

Если x18 равно 0, то достаточно перенести курсор на новую строку. Так как найденная строка x19 определяет лишь верхний край расположенных на ней букв, то для определения строки курсора к ней прибавляется 22 (19 пикселей - буква и еще 3 пикселя для промежутка между частями формулы). Столбец курсора полагается

равным левому краю рамки x1. Если указатель, расположенный в начале стэка x9, имел тип 0 и еще не имел подуказателей, то координаты его подтерма устанавливаются по новым координатам курсора.

Если x18 определяло позицию указателя для перенесения части формулы на новую строку, то проверяется, что для прорисовки нового символа после такого перенесения останется достаточно места. Если это не так, то работа процедуры "выводсимвола" обрывается. В случае обработки фиктивного символа "развязка" (ручной переход на новую строку) длина этого символа x14 заменяется на 0 для продолжения действий по перерисовке переносимой на новую строку части формулы. Далее используется дважды процедура "выводформулы", которая сначала стирает ранее прорисованное изображение по указателю x18, а затем прорисовывает его с начала новой строки (как и в предыдущем случае, отступая от x19 вниз на 22 пикселя). Эта же процедура обращается к "блокредактора(2 ...)" для коррекции параметров поддерева указателя x18, чтобы они соответствовали новым позициям символов на экране.

В обоих случаях факт имевшего место перехода к новой строке отмечается вводом комментария "выводформулы" к посылкам исходной задачи.

После перехода к новой строке - откат к переходу через "ветвь 2" в корневом фрагменте программы. Здесь перехватывается случай обработки фиктивного символа "развязка", и сразу выдается результат (новые координаты курсора). В прочих случаях - переход через "ветвь 1".

Здесь начинается учет вертикальных размеров прорисовываемого символа и коррекция параметров дерева указателей, если места по вертикали недостаточно. Прежде всего, проверяется, что информационный элемент p четверки (i, r, p, s) равен логическому символу "пустоеслово". Это означает, что символ, вообще говоря, не укладывается в вертикальные размеры одной строки (дроби, радикалы, степени, и пр.). В случае символа "степень" сразу же происходит определение необходимых для прорисовки степенного выражения смещений вверх (x16) и вниз (x17) от верхнего и нижнего краев стандартной 19-пиксельной полосы. Смещения берутся до геометрических границ области, в которой будет прорисована вся степень. Они определяются вертикальными размерами основания степени, причем в случае смещения вверх вводятся дополнительные 9 пикселей (на эту величину будет смещен вверх показатель степени). Следующий большой оператор "равно(x19 ...)" определяет пару x19 смещений вверх и вниз от верхнего и нижнего краев стандартной полосы уже в общем случае; для степени берутся найденные перед этим значения x16, x17. Так, в случае дроби пара "(двенадцать десять)" означает, что полоса числителя смещается вверх по отношению к главной полосе на 12 пикселей, а полоса знаменателя - вниз на 10. Таким образом, между ними остается зазор в 3 пикселя.

Далее определяется величина x20 смещений вверх и вниз от стандартной 19-пиксельной полосы, необходимых для прорисовки символа операции. Эта величина равна 0 для обычных символов, а для специальных символов, вертикальные размеры которых зависят от размеров их операндов (интегралы, сигмы, и т.п.), она берется равной своему наименьшему значению - 10 пикселям. После этого предпринимается попытка усмотреть достаточность пространства по вертикали для прорисовки символа: это заведомо так, если нет надоперации, накладывающей ограничения на вертикальные размеры операндов (проверяется с помощью справочника "модульредактора"), не было переноса формулы на новую строку (проверяется по отсутствию комментария "выводформулы"), не было внешней скобки, и расстояния от текущей

строки до верхней и нижней сторон рамки не меньше, чем этого требуют значения в наборе x19.

Если контроль вертикальных размеров все же необходим, то инициализируется пустым словом набор x21, в котором будет накапливаться план коррекций параметров изображения для высвобождения достаточного пространства по вертикали. Типы элементов этого плана таковы:

а) набор (указатель - направление вертикального сдвига (0 - вверх, 1 либо 2 - вниз) - величина сдвига);

б) пара (указатель - новая высота его скобки);

в) пара (указатель - скорректированный набор из информационного элемента A_6 этого указателя, в котором указаны новые параметры изображения соответствующей операции).

Если строка курсора совпадает с главной строкой указателя в начале стэка x9, то для составления плана сдвига происходит просмотр стэка указателей слева направо (от подуказателя к надуказателю), с выделением тех точек, где операция имеет нестандартные размеры по вертикали. Для каждой такой точки корректируются параметры прорисовки операции и переопределяются новые величины сдвига вверх и вниз от главной строки операции (номер этой строки передается переменной x17; для хранения таких величин сдвига по-прежнему используется пара x19). Переход к рассмотрению очередной операции с нестандартными размерами по вертикали - через оператор "иначе 5". Часть программы, расположенная после этого оператора, связана со следующими двумя преобразованиями:

а) Если текущая операция - инфиксная, то просматриваются ее операнды, отличные от текущего набираемого операнда, и определяются необходимые для их прорисовки смещения вверх и вниз от текущей полосы операции. Если они превосходят величины, необходимые для прорисовки набираемого операнда, то соответственно корректируется пара x19.

б) Если встречается операция, которая была перенесена на новую строку, и расстояние между ее последней и предпоследней строками недостаточно (с учетом первого элемента пары x19), то в план сдвига x21 заносятся элементы, определяющие необходимый сдвиг вниз для указателей последней строки. Те элементы последней строки, которые относятся не к ее операндам, а ко всей операции в целом (например, скобки), при этом перерисовываются на экране на новых позициях.

Для операции x25 с нестандартными размерами по вертикали рассматриваются следующие случаи (в них x24 - указатель операции, x23 - его текущий подуказатель, x17 - главная строка для x23, x26 - главная строка для x24):

а) x25 = "дробь". Если x23 - указатель для числителя дроби, то проверяется, достаточно ли расстояние от его главной строки x17 до дробной черты (с учетом последнего элемнта пары x19), и если недостаточно, то в плане сдвига указывается необходимость сдвига числителя вверх на соответствующее число пикселей. Аналогично для знаменателя. В каждом случае вводятся новые значения для указанных в x19 смещений вверх и вниз от главной полосы дроби.

б) x25 = "степень". В этом случае x23 - указатель для показателя степени. Если места вниз от показателя недостаточно, чтобы набираемые символы находились выше главной строки основания степени, то в плане сдвига указывается на сдвиг показателя вверх.

в) x25 = "корень" либо "квадркорень". В этом случае в наборе B_3 информационного элемента $A_6 = ((B_1, B_2), B_3)$ указателя радикала содержится информация о

смещениях вверх и вниз от главной строки, определяющих вертикальные размеры радикала. Если места под радикалом оказывается недостаточно, то в копию набора B_3 вносятся необходимые изменения, и в план сдвига вносится элемент, указывающий на необходимость соответствующей коррекции указателя радикала и на его перерисовку.

г) x25 - один из символов списка "суммавсех", "произведениевсех", "объединениевсех", "пересечениевсех", "дизъюнкциявсех", "конъюнкциявсех", "интеграл", "двойнойинтеграл", "тройнойинтеграл". В этом случае коррекция зависит от номера подуказателя x23 указателя x24. Если это - первый подуказатель, то набирается выражение снизу от знака операции. Тогда проверяется достаточность места сверху от x23, и если его мало, то в x21 указывается необходимость сдвинуть x23 вниз. Аналогично, в случае второго подуказателя (сверху от операции), проверяется достаточность места снизу. Наконец, в случае третьего подуказателя (основной операнд операции) находятся верхняя и нижняя строки всей прорисованной части операции и определяется достаточность места в полосе между ними. Если места мало, то в копии набора B_3 информационного элемента A_6, определяющего размеры символа, вводятся необходимые изменения. В x21 включаются после этого указания на изменение размеров символа, а также указания на вертикальные смещения выражений под и над символом, связанные с увеличением размеров символа.

При просмотре стэка указателей анализируются все встречающиеся указатели со скобкой (см. переход через "ветвь 4" из фрагмента программы, содержащего цикл просмотра стэка), и если запрос на смещения вверх и вниз, определяемый набором x19, превосходит высоту скобки, то в x21 включается задание на увеличение этой высоты. Если же встречается скобка, вертикальные размеры которой поглощают запросы на смещения вверх и вниз, то просмотр стэка обрывается и осуществляется реализация накопленных в наборе x21 команд на изменение ветви дерева указателей. Это делается в фрагменте программы, переход к которому - через оператор "ветвь 3"; собственно реализацию плана изменений x21 выполняет оператор "блокредактора(4 ...)".

Если произошел полный просмотр стэка x9, то по окончании его - переход через "иначе 2". Здесь сначала проверяется, что запрос на смещение вверх не выводит за верхнюю границу рамки формульного редактора; если это не так, то в x21 включается задание на соответствующий сдвиг изображения вниз. Далее проверяется, что запрос на смещение вниз не выводит за нижнюю рамку формульного редактора. Если это не так, то работа формульного редактора обрывается с истинностным значением "ложь". Наконец, далее происходит реализация плана x21.

Если строка курсора располагалась ниже главной строки первого указателя стэка x9, то вместо просмотра стэка - переход через "ветвь 1". Этот случай соответствует формуле, для которой имел место перенос на новую строку. Соответственно, коррекции по вертикали будут затрагивать только последнюю строку, которую можно независимо от расположенной над ней части смещать на необходимое расстояние вниз. Сначала определяется самая нижняя линия x22 изображения, предшествующая перенесенному на новую строку концу формулы, затем находится указатель x24, для которого имел место перенос, и оператор "блокредактора(6 ...)" корректирует этот указатель для сдвига вниз той части формулы, которая лежит на последней строке. На экране подтерм по x24 перерисовывается с помощью операторов "блокредактора(3 ...)" целиком.

После коррекции вертикальных размеров изображения происходит возвращение к

фрагменту программы, начинающемуся с операторов "ветвь 1 равно(левпозиция(x10 2)пустоеслово)" и переход через оператор "ветвь 1". Переменной x18 присваивается: в случае выдачи переменной - логический символ "переменная", иначе - прорисовываемый логический символ. Далее справочник "терм" определяет по значению x18 пару (дополнительная информация для справочника "выводсимвола", который будет выдавать на экран изображение символа - величина сдвига строки к новой позиции курсора после выдачи символа). Сразу же определяется новая строка курсора x17.

Если результат x19 обращения к справочнику "терм" ненулевой (т.е. символ - специального типа), то переменной x19 присваивается первый элемент пары x19 (информация для прорисовки символа), и переход через "ветвь 1". Иначе - рассматриваются следующие случаи:

а) x18 - логический символ "конец" (прорисовка закрывающей простой, фигурной либо квадратной скобки, либо второе вхождение вертикальной черты модуля). Тогда x19 присваивается информация о левой скобке (либо черте модуля) из первого указателя стэка x9, а x18 уточняется - заменяется на тот логический символ, для которого обращение к справочнику "выводсимвола" даст скобку нужного типа.

б) x18 - логический символ "сброс" (нажатие клавиши Enter для завершения набора всей формулы либо ее фрагмента). Стирается курсор, и далее анализируется заголовок x21 первого указателя x20 стэка x9. Если из него видно, что завершается набор фрагмента, после которого следует вернуться к главной строке (например, после набора показателя степени, кванторной приставки, индекса переменной и т.п.), то корректируется строка курсора x8. В случае основания степени, заключенного в скобки, учитывается возможность имевшего место переноса его через строку - тогда главная строка определяется по позиции закрывающей скобки. В случае радикала предпринимается перенесение курсора из позиции для степени радикала в первую позицию под радикалом. В случае дроби, у которой набран только числитель, происходит перенесение курсора в знаменатель; если же завершается набор знаменателя, то выполняется центровка числителя либо знаменателя (в зависимости от того, что из них короче) и возвращение курсора в главную строку. Аналогичные действия выполняются для операций типа "суммавсех", "интеграл", "числосочетаний", и т.п. Во всех перечисленных случаях на новой позиции прорисовывается курсор, и выполнение процедуры "выводсимвола" завершается.

в) x18 - логический символ "последнийоперанд", причем заголовок первого указателя стэка x9 - "Набор". Этот случай означает прорисовку закрывающей скобки матрицы. Переменной x19 передается информация об открывающей скобке.

После рассмотрения перечисленных случаев - переход через "ветвь 1" того фрагмента, в котором происходило обращение к справочнику "терм". Здесь убирается курсор, после чего выполняется обращение к справочнику "выводсимвола" для фактической прорисовки символа на экране. В случае виртуальной прорисовки (если создается дерево указателей для уже известного терма) это обращение тоже нужно, так как оно определяет первый свободный после прорисовки столбец x20. Прежде, чем продолжать действия в общем случае (переход через оператор "ветвь 1"), проверяется, не имеет ли места переход к новой строке при наборе матрицы (соответствующий нажатию клавиши "курсор вниз" символ "спуск" и заголовок "элементы" первого указателя стэка). В этом случае корректируется ордината курсора x17. Затем - переход через "ветвь 1".

После прорисовки символа может понадобиться увеличение горизонтальных размеров надопераций (горизонтальная черта дроби, верхняя черта радикала, горизон-

тальные размеры операций типа "суммавсех", "произведениевсех", объединениевсех" при удлинении текста над или под ними, и т.д.). Сначала предпринимается обращение к оператору "формблок(7 ...)", который просматривает слева направо стэк указателей и создает заготовку x21 списка отрезков, которые должны быть дорисованы. В этот список включаются лишь простейшие случаи - дроби и радикалы. Если встречается более сложный случай операции типа "суммавсех", то для него в x21 ничего не заносится, но выдается значение x22 = 1, указывающее на необходимость специального дополнительного просмотра стэка x9. Такой просмотр и реализуется после оператора "равно(x22 1)". Заметим, что для сложных операций бывает необходима не просто дорисовка отрезков, но частичная перерисовка отрезков. В зависимости от типа операции, для этого сначала стирается старая версия отрезка, а в план x21 заносится новая его версия. По окончании просмотра стэка x9 - переход через оператор "ветвь 1" к новому фрагменту, где все накопленные в списке x21 отрезки выдаются на экран. Далее располагаются операторы, уменьшающие абсциссу курсора x20 для тех случаев, когда он должен оказаться не правее прорисованной операции (под радикалом, над или под дробной чертой, и т.п.). Наконец, происходит прорисовка курсора и выдаются его координаты.

13.3.6 Блок вспомогательных процедур формульного редактора

Ряд вспомогательных процедур формульного редактора вынесен в оператор "блокредактора(x1 ...)". Первая входная переменная x1 указывает номер процедуры. Дадим краткое описание этих процедур.

1. x1 = 1. Процедура определяет номер самой нижней строки изображения, за вычетом некоторого подтерма этого изображения (например, при перенесении формулы на новую строку бывает нужно найти нижнюю границу той части, которая остается связана со старой строкой). Вся эта процедура включена в реализуемый интерпретатором ЛОСа оператор "формблок(1 ...)".

2. x1 = 2. Процедура выполняет коррекцию ветви дерева указателей, соответствующую сдвигу по горизонтали и (или) по вертикали. Входная переменная x2 имеет своим значением пару (ij), где $i = 0$ означает сдвиг влево, иначе - вправо; j - величина сдвига в пикселях. Аналогичное значение (ij) переменной x3 определяет сдвиг по вертикали ($i = 0$ - сдвиг вверх, иначе - вниз). x4 - ограничение сверху для номера строки при сдвиге (если оно нарушено, оператор ложен); x5 - указатель, определяющий обрабатываемую ветвь. Данная процедура бывает нужна и после применения формульного редактора, если происходит перемещение на экране формулы с уже известным деревом указателей. Как и в предыдущем случае, она включена в оператор "формблок(2 ...)".

3. x1 = 3. Процедура осуществляет прорисовку на экране формулы по дереву указателей. x2,x3 - цвет фона и символов; x4 - корень дерева указателей; x5 - указатель режима прорисовки: 0 - прорисовка не происходит, а вычисляются лишь выходные значения x6,x7; 1 - прорисовка выполняется; символьный номер n, больший единицы - на экране прорисовываются лишь строки с номерами, большими n. Выходным переменным x6,x7 присваиваются координаты курсора после прорисовки формулы.

4. x1 = 4. Процедура осуществляет изменение дерева указателей в соответствии с уже упоминавшимся выше планом сдвига по вертикали при выводе очередного символа. x2,x3 - цвет фона и символов; x4 - максимально допустимый номер строки изображения; x5,x6 - текущая позиция курсора; x7 - план сдвига; x8 - указатель режим прорисовки: 0 - нет прорисовки, 1 - есть прорисовка. Выходным переменным x9,x10 присваиваются координаты новой позиции курсора.

5. x1 = 5. Процедура определяет наибольшую ординату точек той части изображения, которая расположена до последней строки x3. x2 - корень ветви дерева указателей. Так как программа процедуры имеет рекурсивный характер, то используется также входная переменная x4, которой присваивается текущее значение наибольшей ориднаты, найденное до просмотра указателя x2. Выходной переменной x5 передается скорректированное после просмотра x2 значение x4. Выходной переменной x6 присваивается значение 0, если внутри x2 еще нет строки x3, иначе ей присваивается 1.

6. x1 = 6. Процедура выполняет коррекцию ветви дерева указателей, соответствующую сдвигу по вертикали вниз части изображения, расположенной после заданной строки. x2 - величина сдвига в пикселях; x3 - максимально допустимый номер строки; x4 - корень дерева указателей; x5 - номер строки, после которой происходит сдвиг.

7. x1 = 7. Процедура выполняет отбрасывание начала стэка указателей x2 вплоть до максимального завершенного операнда. Выходной переменной x3 передается получаемый укороченный стэк.

8. x1 = 8. Это - наиболее важная из процедур блока, которая осуществляет переход к терму от дерева указателей. x2 - корень дерева указателей, выходной переменной x3 присваивается терм. Если терм определить не удалось, то оператор ложен.

   При определении терма прежде всего, проверяется, равен ли нулю тип указателя x2. Если равен, то переход через "иначе 2", где проверяется, что этот указатель соответствует взятому в скобки подуказателю. Находится терм, определяемый подуказателем, который и выдается в качестве результата. Если тип указателя x2 не равен 0, то находится его заголовок x4 = f, и составляется список x5 термов $t_1 \dots t_n$, определяемых подуказателями указателя x2. Если этот список короче списка подуказателей, то выход по значению "ложь". В общем случае результат имеет вид $f(t_1 \dots t_n)$. Однако, имеется большое число особых случаев, в которых результат получается из f и $t_1, \dots, t_n$ более сложным образом. Не указывая здесь подробный список особых случаев (принципы их обработки достаточно легко вычитываются из программы), приведем лишь два первых примера из этого списка - если x4 = "квадркорень", то результат имеет вид "степень(t_1 дробь(1 2))"; если x4 = "корень", то результат имеет вид "степень(t_2 дробь(1 t_1))", и т.п.

9. x1 = 9. Процедура определяет номер x3 самой верхней строки изображения, определяемого деревом указателей с корнем x2.

10. x1 = 10. Процедура выполняет стирание изображения символа на экране и соответствующую коррекцию горизонтальных размеров символов надопераций (например, при стирании последнего символа числителя дроби выполняется

укорочение дробной черты). x2,x3 - координаты позиции, с которой начинается символ; x4 - символ; x5 - необходимая для прорисовки символа дополнительная информация; x6 - цвет фона; x7 - стэк указателей.

11. x1 = 11. Процедура выполняет добавление в начале стэка указателей цепи подуказателей при откате (определяется, к набору какой наименьшей операции следует вернуться), выполняемом для текущей инфиксной операции x3 (либо для минуса). x2 - цепь указателей.

12. x1 = 12. Аналог случая x1 = 11, но откат происходит для операций другого (не инфиксного) типа.

13. x1 = 13. Процедура определяет план сдвига вниз элементов последней строки, если имел место перенос формулы на новую строку. x2 - нижняя линия той части изображения, которая расположена до перехода к последней строке; x3 - последняя строка; x4 - корень дерева указателей. x5 присваивается скорректированное значение x2; x6 - набор указателей в последней строке, каждый из которых независимо от других должен быть сдвинут вниз; x7 - набор дополнительных указаний на сдвиг. Каждый такое указание имеет либо вид (1 A),где A - указатель с правой скобкой, перенесенной через строку (при опускании последней строки размеры правой скобки увеличиваются, так что одновременно должны быть увеличены и размеры левой скобки), либо вид(2 a (b, c)), где a - символ, который должен быть сдвинут, но не находится внутри указателей списка x6 (например, перенесенный через строку "плюс"); (b, c) - извлеченная из некоторого указателя пара, определяющая координаты символа.

13.3.7 Блок эмуляции набора формулы с клавиатуры

Для автоматического формирования дерева указателей по известному терму используется специальный режим формульного редактора. В этом режиме определяется такая последовательность ввода символов с клавиатуры, при которой будет получено дерево указателей данного терма; переход от дерева указателей к терму здесь не нужен и программой не ыполняется. Эмуляция набора терма с клавиатуры осуществляется процедурой "модульредактора(x1 x2)". Значением ее входной переменной x1 служит вхождение первого символа набираемого терма. Выходная переменная x2 при этом перечисляет четверки (i r p s), представляющие собой результаты обращения к справочнику "формредактор" на тех символах, которые должны были бы вводиться с клавиатуры. Оператор "модульредактора" реализован рекурсивным образом; на очередном уровне рекурсии анализируется вхождение некоторой операции, и в зависимости от ее особенностей предпринимаются обращения к обработке с его же помощью тех или иных вхождений, расположенных внутри этой операции.

Выход в корневой фрагмент программы "модульредактора" - либо обычным образом через главное меню, либо через раздел "Вспомогательные процедуры интерфейсов - Формульный редактор - Блок эмуляции набора формулы с клавиатуры" оглавления программ.

Прежде всего, переменной x3 присваивается тот символ, который расположен по вхождению x1. Если этот символ представляет собой переменную, то процедура "видпеременной" находит пару (код буквы s для данной переменной - индекс переменной); при отсутствии индекса второй элемент пары равен 0. Сначала выдается четверка (9 1 0 s), соответствующая вводу буквы переменной. При запросе очередной

четверки - переход через "ветвь 4", где анализируется наличие индекса. Если индекс имеется, то сначала выдается четверка (4 0 пустоеслово спуск), соответствующая нажатию клавиши "курсор вниз" для перехода к уровню индекса. Затем перечисляются четверки, соответствующие цифрам индекса, и в конце добавляется четверка (4 0 0 сброс), соответствующая нажатию клавиши Enter. Заметим, что во всех случаях по окончании обработки текущего символа x3 происходит переход через "ветвь 2" к фрагменту, проверяющему, не является ли текущее обрабатываемое вхождение x1 корневым (это будет выполнено для первого обращения к процедуре и не выполнено при рекурсивных обращениях). Если оно корневое, то выдается завершающая набор всего терма четверка (4 0 0 сброс). Иначе - перечисление обрывается без выдачи такой четверки.

Если символ x3 не является символом переменной, то из начального фрагмента через "иначе 3" переходим к фрагменту, проверяющему, не происходит ли при прорисовке этого символа использование обозначения некоторого другого символа. Например, при прорисовке сложения векторов используется тот же знак "плюс", что и для сложения чисел. Здесь используется справочник "Замена"; если он выдает ненулевой результат x4, то первоначальный символ x3 заменяется на новую его версию x4 (заметим, что с этого момента x3 уже не будет равен символу по вхождению x1).

Далее, переходя через "ветвь 1", попадаем в фрагмент, первый оператор которого обращается к справочнику "блокредактора" для символа x3 и получает от него значение x4 - первые три элемента (i r p) той четверки (i r p x3), которая соответствует вводу символа x3 с клавиатуры (прямому либо с помощью кодовых комбинаций клавиш). Предпринимается проверка, что x4 не равно 0, причем вхождение x1 не является бесскобочным вхождением логического символа, отличного от специальных случаев таких вхождений, допустимых для формульного редактора (символы цифр, запятая, константы "е", "пи", вспомогательные символы "максимум", минимум", используемые для указания типа экстремума, и т.п.). Если это не так, то проверяется, не выдается ли на экран терм, определяющий вид некоторого другого терма и использующий операцию "запись" (как это принято в ЛОСе). Если последний случай распознается, то для него генерируется соответствующая последовательность четверок, иначе - выдается значение 1, указывающее на невозможность отображения терма формульным редактором. Если же первое условие (x4 не равно 0 и т.д.) выполнено, то переход через "иначе 1" для дальнейшей обработки символа x3. Здесь начинается разбор случаев для типов i символа x3:

1. Прежде всего, рассматривается случай, когда $i = 2$, причем x3 не есть символ "минус". Это - случай инфиксных операций. Состаавляется список x5 вхождений операндов операции x3 (до этого особо рассматривается случай символа "вариант"), и последовательно просматриваются позиции x6 этого списка. Для текущей позиции прежде всего определяется, будет ли соответствующий операнд заключаться в скобки. Индикатор наличия скобки - переменная x7 (x7 =1 означает наличие скобки). Это делается исходя из сравнения числовых характеристик "силы" связи между операндами для различных инфиксных операций, получаемых с помощью справочника "блокредактора". Если скобка нужна, то перечисление для текущего операнда начинается с четверки (5 1 0 операнд) для открывающей скобки, а завершается четверкой (5 1 0 последнийоперанд) для закрывающей. После набора операнда выдается четверка для очередного вхождения инфиксной операции (кроме случая последнего операн-

да), причем в случае x3 = "умножение" анализируется необходимость прорисовки умножения в виде точки. Точке соответствует четверка (2 1 1 умножение), отсутствию точки - (2 0 1 умножение).

2. Либо $i = 1$, либо x3 = "минус". Это - случай префиксных операций либо символьных констант. Программа содержит большое число независимых ветвей для конкретных символов x3. В общем, они не содержат сколь-нибудь интересных моментов, кроме следующих двух: отделение символами умножения операндов для выражений типа $\triangle(ABC)$ (они распознаются при помощи справочника "кортеж") и определение случаев, когда операнд одноместной операции заключается в скобки. В частности, для этого применяется справочник "прямоепроизведение".

3. $i = 3$. Набирается степень. Прежде, чем набирать ее "в общем виде", предпринимаются попытки усмотреть квадратный корень, радикал степени от 3 до 9 и экспоненту.

4. $i = 6$. Набирается десятичное число.

5. $i = 7$. Набирается выражение для числового промежутка, модуля, целой части, конечного перечня, и т.п.

6. $i = 8$. Набор символьных констант либо конечных отрезков целых чисел.

7. $i = 10$. Набор производной либо частной производной.

8. $i = 11$. Некоторые специальные логические обозначения.

13.4 Программа текстового редактора

Хотя программа текстового редактора - явление достаточно тривиальное, все же в нее включены фрагменты "чужих" интерфейсов, обращающихся к данной программе. Поэтому ее также приходится время от времени корректировать и пополнять. Соответственно, несколько более подробное описание ее устройства может оказаться весьма полезным.

Текстовый редактор реализован оператором "текстредактор(x1 x2 x3 x4 x5 x6 x7 x8)". Входные данные суть: x1 - номер позиции буфера текстов, начиная с которой будут размещаться символы, вводимые в рамке редактирования (начинающийся с x1 отрезок буфера текстов взаимно-однозначно сопоставлен позициям рамки, согласно обычному принципу переноса на новую строку); x2,x3,x4,x5 - координаты рамки редактирования в пикселях; x6 - цвет фона; x7 - цвет символов. Выходной переменной x8 присваивается по завершении редактирования номер последней занятой введенными символами позиции буфера текстов. Если при обращении положить значение x7 равным логическому символу 0 либо символу "метка", то перед редактированием сохраняется старое изображение, иначе рамка редактирования расчищается. Расчистка буфера текстов в начале редактирования не выполняется, и при необходимости ее надо выполнить отдельно.

Интерфейс текстового редактора уже был описан выше, поэтому сразу перейдем к рассмотрению его программы. Вход в начальный ее фрагмент - либо через главное меню, либо через пункт "Вспомогательные процедуры интерфейсов - Текстовый редактор" оглавления программ.

Прежде всего, драйвер клавиатуры переводится в режим кириллицы. Затем предпринимается расчистка рамки редактирования (кроме указанных выше случаев специальных значений x7). Координаты x4,x5 правого нижнего угла рамки корректируются так, чтобы по ее размеры по вертикали были кратны 19, а по горизонтали - 8. Переменной x7, если она до этого не указывала цвет символов, присваивается значение 1, соответствующее черным символам. Инициализируются координаты x9, x10 текущей позиции курсора (изначально - по левому верхнему углу рамки). Инициализируется указатель x11 позиции буфера текстов, соответствующей текущей позиции курсора. Если исходная задача имела комментарий к посылкам (теквхожд A_1 A_2), указывающий на ту позицию курсора, с которой нужно войти в редактирование, то значения x9,x10,x11 корректируются согласно данному комментарию, а сам он удаляется. Затем - переход через "ветвь 2".

Здесь инициализируется переменная x12, сохраняющая информацию о текущей выполняемой операции, требующей нескольких нажатий клавиш (удаление фрагмента текста, режим вставки, перенесение фрагмента текста либо сохранение его в буфере). Ее значением служит набор, первый элемент которого указывает на операцию либо на ее этап. Если он равен 0, то специальная операция отсутствует; 1 - режим вставки; 2 - удаление; 3 либо 4 - перенесение фрагмента текста; "десять" - регистрация фрагмента текста в буфере. Последний элемент набора x12 определяет величину шага курсора: если он равен 0, то курсор сдвигается по горизонтали либо вертикали на размеры одной буквы, иначе - на размеры 8-буквенной цепочки. После инициализации x12 расположен оператор "повторение", к которому будут происходить откаты при запросе очередной команды.

Прежде всего, считывается символ x13 с текущей позиции буфера текстов, и этот символ прорисовывается на экране на позиции курсора. Цвет фона здесь является цветом курсора и выбирается в зависимости от того, какая операция выполняется, цвет символа равен x7. Далее происходит обращение к оператору "клавиатура" для ввода очередной команды x16. Если текущая позиция буфера текстов была пустая, то цветовые ее атрибуты переустанавливаются согласно x6,x7. Затем восстанавливается цвет фона x6 на текущей позиции курсора - кроме случаев, когда на этой позиции следует удерживать особый цвет фона до завершения начатой операции.

Если заголовок x12 равен 0 (отсутствует режим специальной операции) и исходная задача имеет комментарий к посылкам "слово", указывающий на режим перевода нажатия клавиатуры в логический символы, кодирующий нажатую клавишу, то данный комментарий удаляется, на экране прорисовывается с текущей позиции название символа, корректируются x9,x10,x11, и откат к запросу очередной команды. Заметим, что данный режим бывает нужен в редакторе программ при определении символа, соответствующего нажатию той или иной клавиши. Если такой режим не имел места, то переход через "ветвь 1". Здесь начинается цепь подслучаев для различных команд x16:

1. x16 = "циклвариантов". Нажата клавиша "курсор влево". В режиме вставки команда игнорируется. Иначе происходит коррекция значений x9,x10,x11, соответствующая одному шагу влево. Здесь возможны два режима: если конец набора x12 равен 0, то шаг делается на 8 пикселей (горизонтальный размер одной буквы). При этом из крайней левой позиции строки делается переход на крайнюю правую позицию предыдущей строки. Если же конец набора x12 не равен 0, то имеет место режим увеличенного шага, равного сразу 8 размерам буквы. Здесь переход на предыдущую строку не выполняется.

2. x16 = "внешконъюнкция". Нажата клавиша "курсор вправо". Обрабатывается аналогично предыдущему.

3. x16 = "внешсумма". Клавиша "курсор вверх"; аналогично предыдущему.

4. x16 = "подстановка". Клавиша "курсор вниз"; аналогично предыдущему.

5. x16 = "тринадцать". Клавиша Enter, завершающая редактирование. В режиме вставки символов игнорируется. Обращение к оператору "видео(текстредактор ...)" позволяет определить последнюю позицию x21 буфера текстов (после начальной позиции x1), содержащую символ, прорисовываемый на экране "непустым" образом. Если этот символ - искусственно введенный нажатием клавиши F9 указатель конца текста "нормсуп", то он заменяется на символ пробела. После этого выдается результат x21.

6. x16 = "копияблокаанализа". Клавиша F4, используемая в текстовом редакторе для вызова справочника по системе (F1 занято как клавиша поиска парной скобки).

7. x16 = "элементы". Клавиша F8, используемая для перехода в режим трансляции нажатия очередной клавиши в логический символ - код этой клавиши. Указателем режима служит комментарий "слово" к посылкам исходной задачи.

8. x16 = "8". Нажата клавиша Backspace. Если имел место режим вставки, то восстанавливается обычный режим, а содержимое буфера текстов, расположенное после позиции вставки, перерисовывается на экране. Такая перерисовка необходима для восстановления правой части текущей строки, пропавшей с экрана при вставке. Если имел место режим удаления либо регистрации фрагмента текста в буфере, то восстанавливается обычный режим и удаляется введенный в начале операции цветной маркер. Если имел место режим перенесения фрагмента на новое место, то удаляется последний введенный до этого цветной маркер. Если цветной маркер был единственный, то восстанавливается обычный режим, иначе - режим перенесения фрагмента сохраняется, но x12 корректируется на случай одного оставшегося маркера. Наконец, в обычном режиме происходит откат курсора на предыдущую позицию и занесение нуля на соответствующую позицию буфера текстов.

9. x16 = "внешдизъюнкция". Нажата клавиша Insert, инициализирующая либо завершающая режим вставки. Если заголовок набора x12 равен 0, то определяется последняя позиция x21 буфера текстов, на которой находится "непустой" символ, и указатель режима x12 заменяется на (1 x21 0). Если же имел место режим вставки, то восстанавливается обычный режим и происходит перерисовка части текста, расположенной после точки вставки.

10. x16 = "соответвхожд". Нажата клавиша Delete. Если имел место обычный режим, то указатель режима x12 заменяется на набор (2 - столбец курсора - строка курсора - позиция буфера текстов - 0), инициализирующий режим удаления с прорисовкой красного маркера на текущей позиции. Если же режим удаления уже был включен (начало набора x12 равно 2), то находится сохраненная в x12 позиция x17 начала удаляемой части в буфере текстов. Проверяется, что она предшествует текущей позиции x11. Затем находится последняя непустая позиция x22 буфера текстов, и происходит исключение отрезка буфера текстов

с x17 по x11, причем перед x22 вставляется такое же число пустых символов с соответствующим цветом фона. Текущая позиция устанавливается по маркеру начала удаленного отрезка, происходит перерисовка с этой позиции оставшейся части текста, и восстанавливается обычный режим.

11. x16 = "Плюс". Нажата клавиша Esc, приводящая к отказу от редактирования.

12. x16 = "9". Нажата клавиша Tab, осуществляющая смену малого шага курсора на большой и обратно. Изменяется последний элемент набора x12.

13. x16 = "приведение". Нажата клавиша Home. Курсор сдвигается к левому краю строки, соответственно корректируется текущая позиция x11 буфера текстов.

14. x16 = "группировки". Нажата клавиша End. Курсорв сдвигается к правому краю строки.

15. x16 = "Текзадача". Нажата клавиша Ctr-F4. Происходит циклическая смена цвета фона x6 (цвета нумеруются от 1 до 16).

16. x16 = "текуровень". Нажата клавиша Ctr-F5. Происходит циклическая смена цвета символов x7.

17. x16 = "узелраздела". Нажата клавиша F3. Выдается краткая подсказка относительно команд текстового редактора.

18. x16 = "стандменьше". Нажата клавиша F1. Если на текущей позиции x11 буфера текстов находится скобка, то переменной x17 присваивается позиция буфера текстов для парной скобки (если такую скобку удается найти). Находятся координаты позиции x17 на экране, курсор переводится на эту позицию, и x11 присваивается значение x17.

19. x16 = "попыткапараметризации". Нажата клавиша F2. Если имел место обычный режим (заголовок набора x12 равен 0), то инициализируется режим перенесения фрагмента текста на новое место. При этом x12 заменяется на набор (3 - столбец курсора - строка курсора - текущая позиция в буфере текстов - 0). Если режим перенесения уже был инициализирован, но отмечено только начало переносимого фрагмента (заголовок набора x12 равен 3), то проверяется, что текущая позиция в буфере текстов расположена не раньше выделенного начала, заголовок набора x12 меняется на 4, и после указателя текущей позиции начала фрагмента в буфере текстов в набор x12 вставляется тройка (столбец курсора - строка курсора - текущая позиция в буфере текстов), отмечающая конец переносимого фрагмента. Наконец, если уже выделены начало и конец переносимого фрагмента (заголовок набора x12 равен 4), то проверяется, что текущая позиция x11 не расположена внутри фрагмента, после чего реализуется перестановка фрагмента в буфере текстов, установка курсора по началу изменяемой части изображения и перерисовка с этой позиции остатка текста. Далее восстанавливается обычный режим.

20. x16 = "фиктопер" либо x16 = "простоенеравенство". Соответственно, нажаты клавиши F6 либо F7. В первом случае оператор "видео(исключение ...)" исключает ровно одну строку из буфера текстов, начиная с текущей позиции, во втором - оператор "видео(сборкнабора ...)" вставляет ровно одну строку,

заполненную символами пробелов. Затем предпринимается перерисовка текста, начиная с текущей позиции.

21. x16 = "путьвхождения". Нажата клавиша Ctr-F9. Предпринимается включение либо выключение (с помощью ввода либо удаления комментария "биключ" к посылкам исходной задачи) дополнительного драйвера клавиатуры, переводящего нажатия латинских клавиш в условно соответствующие им буквы кириллицы. Эта возможность бывает полезна для клавиатуры, не имеющей кириллицы.

22. x16 = "точкапривязки". Нажата клавиша Page Down. Если имел место обычный режим (заголовок набора x12 равен 0), то инициализируется режим регистрации фрагмента текстов в буфере текстового редактора. При этом x12 заменяется на набор (десять - столбец курсора - строка курсора - текущая позиция в буфере текстов - 0), указывающий на начало заносимого в буфер фрагмента. Если же режим регистрации фрагмента уже был инициализирован, то удаляется синий цветовой маркер начала фрагмента, восстанавливается обычный режим, проверяется, что текущая позиция расположена не раньше начала фрагмента, и предпринимается регистрация фрагмента в буфере текстового редактора с помощью оператора "видео(извлекается ...)". Заметим, что роль буфера текстового редактора играет массив, доступ к которому осуществляется только на уровне интерпретатора ЛОСа.

23. x16 = "контрольунификации". Нажата клавиша Page Up. Проверяется, что имеет место обычный режим, и в буфер текстов начиная с текущей позиции x11 вставляется содержимое буфера текстового редактора. Затем предпринимается перерисовка текста начиная с текущей позиции.

24. x16 - один из символов "новыйузел", "конъюнктоперанд", "записьзадачи", "элементызадачи", "стрелкапирса". Они соответствуют специальным командам редактора приемов (нажатия клавиш Ctr-з,Ctr-у,Ctr-х (кир.), Shift-1, Shift-3). Вызов этих команд из текстового редактора служит для автоматической вставки в редактируемый текст различных кодовых фрагментов. Программа проверяет, что имеет место использование текстового редактора в процессе редактирования одного из окон описания приема (комментарий "учетпоказателя" к посылкам исходной задачи); в случае клавиш Shift-1,Shift-3 проверяется условие на номер окна, после чего подготавливается прерывание текстового редактирования: создается комментарий (теквхожд ...) к посылкам исходной задачи, содержащий информацию, необходимую для последующего продолжения редактирования; содержимое буфера текстов сохраняется в дополнительном буфере, и происходит выход из программы текстового редактора. Информация о команде x16 сохраняется в том же комментарии (теквхожд ...); эта команда будет передана через данный комментарий программе редактора приемов. Например, в случае x16 = "конъюнктоперанд" произойдет выход в оглавление фильтров ГЕНОЛОГа, а после выбора концевого пункта данного оглавления будут реализованы восстановление на экране сохраненного в дополнительном буфере текста, добавление к нему текста шаблона фильтра для выбранного концевого пункта и возвращение в текстовое редактирование.

25. x16 = "список". Нажата клавиша F9 - указатель на то, что пустой отрезок в конце текста вплоть до текущей позиции должен быть включен в область

редактирования. Здесь предпринимается замена значения x16 на логический символ "нормсуп", используемый в качестве специального указателя конца текста (прорисовывается как $\perp$), и переход через "ветвь 1" к продолжению обработки команды x16.

26. x16 - один из логических символов "транслвыражения", "конъюнктоперанд", "кортеж", "утверждение"(клавиши Ctr-a,Ctr-e, Ctr-n,Ctr-v, все - лат.). В этом случае вводится символ, соответственно, квантора общности, существования, отрицания и дизъюнкции. x16 изменяется так, как это нужно для прорисовки указанных символов, и переход через "ветвь 1".

27. Проработка общего случая - введенный символ x16 регистрируется на текущей позиции. Предварительно проверяется, не был ли включен режим дополнительного драйвера перевода латинской клавиатуры в кириллицу, и если был, то x16 корректируется в соответствии с этим драйвером, через комментарий (текстредактор ...) к посылкам исходной задачи. Затем отдельно рассматриваются два случая - наличие общего режима (заголовок x12 равен 0) и режима вставки (заголовок равен 1). В первом случае символ x16 заносится на текущую позицию буфера текстов и прорисовывается на экране вместо курсора; если позиция в рамке редактирования не последняя и x11 не выходит за рамки буфера текстов, то x11 увеличивается на 1 и предпринимается сдвиг позиции курсора для ввода следующей буквы. Во втором случае предпринимается сдвиг на одну позицию вправо содержимого буфера текстов для освобождения позиции x11 и одновременный сдвиг вправо текущей строки на экране, начиная с текущей позиции (оператор "видео(вставка ...)"). На высвободившуюся позицию в буфере текстов и на экране заносится буква x16. Далее, как и в первом случае, если позиция в рамке редактирования не последняя, происходит увеличение x11 на единицу и сдвиг позиции курсора. Если курсор переносится на новую строку, то предварительно предпринимается перерисовка всей оставшейся части текста (для отображения на экране "спрятанной" при вставке правой части текущей строки).

13.5 Программы геометрического и текстформульного редакторов

Интерфейсы и основные структуры данных геометрического и текстформульного редакторов были описаны выше; программы их в принципиальном плане достаточно просты, так что здесь мы ограничимся лишь очень кратким описанием их устройства.

Программа геометрического редактора реализуется оператором "геомредактор(x1 x2 x3 x4 x5 x6 x7 x8)", у которого x1-x4 определяют прямоугольную рамку чертежа; x5,x6 - цвет фона и линий; x7 - описание чертежа; x8 - указатель на режим изменения (1) либо только прорисовку чертежа. Описание чертежа представляет собой пару, первый элемент которой перечисляет пятерки (переменная, обозначающая точку - столбец точки - строка точки - столбец буквы, обозначающей точку - строка буквы, обозначающей точку), причем координаты в пикселях берутся не относительно рамки чертежа, а относительно всей рамки экрана. Второй элемент пары перечисляет (уже в терминах переменных, обозначающих точки) элементы чертежа, которые должны быть прорисованы - отрезки, окружности, и т.п. В этом же элементе могут сохраняться указатели на измененные размеры рамки чертежа.

Программа геометрического редактора прежде всего удаляет из описания чертежа указатели на имевшее место изменение рамки, после чего переключает драйвер клавиатуры на режим латиницы. Затем вводится копия x9 описания чертежа x7, с которой и будет происходить работа при редактировании чертежа. Инициализируется нулем указатель x10 режима работы геометрического редактора (ввод новых точек, проведение отрезков, и т.д.). Инициализируются нулями вспомогательные регистры x11,x13, используемые для сохранения информации о точках в операциях, применяемых к нескольким точкам (в частности, x13 указывает обозначение "текущей точки", которую можно сдвигать курсорами); инициализируется пустым словом накопитель x12 переменных - обозначений точек. Если этот накопитель непуст, то для новой точки обозначение берется из него, иначе - выбирается первая неиспользованная большая буква. Далее располагается оператор "повторение", к которому происходят откаты для перерисовки чертежа.

Перерисовка начинается с расчистки рамки чертежа. Затем последовательно просматриваются подлежащие прорисовке элементы x15 из описания чертежа, и справочник "геомредактор" выполняет их прорисовку. После этого просматриваются все точки из описания чертежа, и изображаются соответствующие им буквы. Если x8 равно 0, то восстанавливается режим кириллицы для драйвера клавиатуры, и работа геометрического редактор обрывается. Иначе - голубым цветом обозначается рамка чертежа, в правом верхнем углу этой рамки отделяется небольшой прямоугольник, и в нем размещается логический символ, указывающий режим редактирования. Если указатель режима x10 равен 0, то прямоугольник остается пустым.

Далее идет оператор "повторение", к которому осуществляются откаты для запроса очередной команды x15. После него - обычная цепочка обработки этой команды. Мы не будем подробно описывать реализацию этих команд, так как она в большинстве случаев весьма проста. Для операций, изменяющих координаты точек (даже для сдвига одной точки на один пиксел), предпринимается полная перерисовка чертежа. В общем, необходимости ускорять такую крайне неэкономную схему пока не возникало. Сохранение элементов, связывающих точки линиями чертежа, приводит к тому, что при перемещениях точек большая (но не вся) часть чертежа также корректируется. Разумеется, данную, по существу, очень примитивную версию геометрического редактора можно было бы довести в рамках описанной структуры данных до весьма высокого уровня, однако для той цели, для которой она была создана - для снабжения чертежами школьных планиметрических задач на вычисление - она и сейчас вполне достаточна. Развитие системы пошло не по пути усиления ручного интерфейса редактирования чертежей, а по пути их автоматического построения.

Программа текстформульного редактора реализуется оператором "текстформ(x1 x2)". Здесь x1 - текстформульная структура, определяющая форматированный текст с вставленными формулами и чертежами; x2 - набор информационных элементов, определяющих режим работы текстформульного редактора. Типы информационных элементов в x1 и x2 были определены выше, при описании основных процедур интерфейса логической системы.

Корневой фрагмент программы "текстформ" прежде всего создает в верхней части экрана меню для работы с тексформульным редактором (кроме случаев, когда при обращении не предусмотрена работа в диалоговом режиме). Если текстформульный редактор используется для просмотра результатов грамматического анализа текста, то берется специальное меню. Затем определяется номер x3 нижней строки той области, которая выделена для редактирования, и переход через "ветвь 2".

Инциализируются переменная x4, которая хранит номер текущей строки, а также

переменная x5, хранящая номер первой свободной строки. Инициализируется экранная структура x6, описывающая полосы изображения (формат ее элементов был описан там же, где и типы элементов в x1,x2). Инициализируется единицей номер x7 той полосы экранной структуры, с которой начинается прорисовка на экране. Проверяется наличие в x2 элемента (видео ...); если он есть, то x5,x6,x7 корректируются на значения, указанные в этом элементе. Далее переход через "ветвь 1".

Здесь размещен оператор "повторение", к которому происходят откаты при перерисовке изображения. Сначала предпринимается прорисовка тех элементов изображения, которые уже представлены в экранной структуре. Затем - переход через "ветвь 1" к циклу продолжения прорисовки за счет тех элементов текстформульной структуры, которые следуют после последнего элемента, учтенного в экранной структуре. После дорисовки проверяется наличие в x2 элемента (высотаполосы ...). Если он есть, то в нем регистрируется высота полосы, востребованной для прорисовки (заметим, что в данном случае фактической прорисовки на экране не происходит), и выход из редактора. Далее проверяется наличие элемента "выход", который также обрывает работу редактора. Наконец, при отсутствии указанных элементов - вход в цикл обращений к оператору "автоменю(x10)"для ввода команд. Перед циклом инициализируются нулями вспомогательные переменные x8,x9. Значением x8 будет далее служить ссылка на левый край выделенного элемента текстформульной структуры, к которому относится команда x10; x9 - используется при форматировании текста. Процедуры обработчика команд легко анализируются с учетом ранее приведенных описаний форматов данных; начальные точки обработки команд указаны в оглавлении программ (раздел "Вспомогательные процедуры интерфейсов - Текстформульный редактор - Цикл обращений к клавиатуре").

Глава 14

Программа редактора программ

Хотя вносить изменения в программу редактора программ ЛОСа приходится крайне редко, в этих особых случаях описание ее процедур может оказаться необходимым. Кроме того, такое описание будет полезно для попыток общего усовершенствования интерфейса редактирования программ на ЛОСе.

В главе "Программы общего интерфейса системы" (подраздел "Обращение к редактору ЛОСа" раздела "Главное меню") была описана начальная часть программы редактора программ. В этой части произошла прорисовка фрагмента программы и инициализированы следующие программные переменные: x13 - логический символ, программа которого редактируется; x11 - номер выделенного оператора перехода текущего фрагмента (не символьный номер, а десятичное число); x12 - набор троек, определяющий путь к текущему фрагменту программы от ее корня (первые два элемента тройки - ссылка на фрагмент в блоке программ, последний элемент - номер перехода от него к подфрагменту; корень соответствует началу набора x12); x14 - набор ссылок на отключенные при редактировании, но пока не удаленные ветви программы; (x15,x16) - ссылка на текущий фрагмент; x17 - сам фрагмент; x18 - набор определяющих прорисовку на экране операторов перехода текущего фрагмента пятерок (столбец - строка - вхождение оператора перехода в фрагмент программы - исходная позиция в буфере текстов текста этого оператора - заключительная его позиция); x19 - вхождение в x18 пятерки для текущего оператора перехода, либо 0 при отсутствии переходов. Кроме того, произошло обращение к оператору "автоменю(x21)" для ввода очередной команды x21. В данной главе мы приведем описание действий, выполняемых программой при обработке команды x21.

1. x21 - либо "внешконъюнкция", либо "циклвариантов"(клавиши "курсор вправо" и "курсор влево"). Происходит переход к следующему либо предыдущему оператору перехода (текущий оператор выделен желтым цветом фона). Проверяется, что x19 не равно 0 и имеет в наборе x18 соседа, соответственно, справа либо слева. Затем корректируется x11, цвет фона старого оператора перехода делается белым, а нового оператора - желтым, и переменной x19 присваивается вхождение x22 в x18 пятерки для нового оператора перехода.

2. x21 = "подстановка". Нажата клавиша "курсор вниз". Если x19 не равно 0, то происходит переход к подфрагменту для текущего оператора перехода. К началу набора x12 добавляется тройка (x15 x16 x11), указывающая на пройденный фрагмент программы. Переменной x22 присваивается текущий оператор перехода. Если он соответствует реальному переходу, то имеет два операнда - логических символа x23 и x24, определяющих ссылку на подфрагмент. В этом

случае значения x15 и x16 заменяются на x23,x24; переменной x11 (указателю номера выделенного оператора перехода) присваивается 1, и откат к прорисовке подфрагмента. Если же переход - недоопределенный, то оператор x22 имеет лишь один операнд - номер такого перехода (он является нелокальным и отличается от прорисованного на экране локального номера перехода). Нелокальный номер прорисовывается в левом верхнем углу экрана, x15 заменяется на вхождение левого края исходной задачи (чтобы указать на отсутствие фргамента, к прорисовке которого произойдет откат), и откат к прорисовке пустого фрагмента.

3. x21 = "внешсумма". Нажата клавиша "курсор вверх". Происходит возвращение к надфрагменту текущего фрагмента, либо, в случае корневого фрагмента (набор x12 пуст), возвращение к главному меню. В последнем случае предварительно завершается исключение из блока программ ранее отключенных ветвей, ссылки на которые хранятся в наборе x14. В первом случае (переход через "иначе 2") - переменным x15 и x16 присваивается ссылка на надфрагмент, извлеченная из первой тройки стэка x12, а переменной x11 - номер текущего оператора перехода, извлекаемый из той же тройки. Отбрасывается первая тройка стэка x12, и откат к прорисовке надфрагмента.

4. x21 = "Плюс". Клавиша Esc, при нажатии которой удаляются все отключенные фрагменты из буфера x14 и происходит внутренний перезапуск системы с дополнительной расчисткой зоны программ.

5. x21 = "контрольунификации". Клавиша PageUp, при нажатии которой происходит удаление отключенных фрагментов и возвращение к главному меню без внутреннего перезапуска системы. В отличие от Esc, здесь сохраняются комментарии к посылкам исходной задачи, что бывает необходимо для выполнения ряда операций, инициированных в просмотре текущей программы).

6. x21 = "частичныйответ". Нажата клавиша "o", переводящая в режим просмотра операторов фрагмента программы. Этот режим обладает своим собственным длинным списком команд. Поэтому ограничимся здесь описанием первых выполняемых шагов, а описание списка используемых в данном режиме команд приведем в конце, после рассмотрения списка команд основного режима редактора программ. Прежде всего, проверяется непустота текущего фрагмента x17. Если текущий фрагмент имел операторы перехода (x19 не равно 0), то восстанавливается белый фон выделенного оператора перехода. Переменной x22 присваивается набор из пяти цветов, которые будут использоваться при выделении подтермов (после того, как пятый цвет использован - переход к первому, и т.д.). Инициализируется пустым словом стэк x23, который будет сохранять наборы четверок для перекраски надтермов текущего терма; каждая такая четверка указывает: исходную позицию буфера текстов для перекраски, заключительную позицию для перекраски, а также столбец и строку для выдачи надтерма на экране. Инициализируется переменная x24, которая будет указывать вхождение текущего цвета в набор x22. Переменной x21 присваивается пара (вхождение текущего выделенного оператора в фрагмент программы - указатель типа оператора (оператор перехода - 1, иначе - 0)). Переменной x25 присваивается вхождение выделенного подтерма текущего оператора. Инициализируется переменная x26, которая будет являться указателем направления перемещения

при просмотре подтермов одного уровня (1 - влево, 0 - вправо). Инициализируются переменные x27, x30 (далее они указывают начало и конец выделенного подтерма в буфере текстов), а также переменные x28, x29 - столбец и строка начала этого подтерма на экране.

После инициализации переменных прорисовывается меню, обеспечивающее режим просмотра подтермов. Далее идет оператор "повторение", к которому будут происходить откаты при перекрасках. После оператора "ветвь 2" - действия по перекраске выделенного подтерма текущего оператора. Здесь учитывается указатель направления перемещения x26: при движении влево изначально известно значение x30 и по нему определяется соответствующее x27, при движении вправо - по известному x27 определяется x30. Затем - переход через "ветвь 2".

Здесь осуществляется учет комментария (умножсимв p) к посылкам исходной задачи. Если он имеется, то p - позиция в буфере текстов, соответствующая тому подтерму, который желательно выделить многоцветной указкой. Соответственно, через комментарий (автоклавиатура ...) передаются команды для нажатий клавиш курсора, обеспечивающих выделение подтерма. Команды выдаются поштучно, и после обработки очередной команды происходит возвращение в данную точку для продолжения работы с комментарием (умножсимв p). После выделения нужного подтерма он удаляется. Далее - переход через "ветвь 1" и обращение к оператору "автоменю(x31)", вводящему очередную команду режима просмотра подтермов операторов.

7. x21 = "точкапривязки". Нажата клавиша "PageDown". Происходит поиск в программе текущего символа x13 первого недоопределенного перехода, экран расчищается для входа в редактирование фрагмента, и в верхнем левом углу прорисовывается номер перехода. Собственно поиск недоопределенного перехода выполняется оператором "прогфайл(продолжение x13 x22 x23)", после чего предпринимается коррекция стэка x12 для пути к найденному фрагменту.

8. x21 = "числитель". Нажата клавиша "p", инициирующая редактирование фрагмента. Прежде всего, создается стэк x22 для учета новых ветвей программы, появляющихся при редактировании. Если при редактировании фрагмента возникли новые операторы перехода, то после его ввода автоматически появится предложение ввести подфрагмент для последнего из не определенных переходов. В процессе ввода подфрагмента могут возникнуть дальнейшие операторы перехода, для которых тоже будут сделаны предложения ввести их подфрагменты, и т.д. В результате к текущему вводимому фрагменту будет вести цепочка надфрагментов, часть которых указаны в стэке x12, а часть - новые надфрагменты, для которых и создан стэк x22. В него заносятся наборы, последние два элемента которых являются ссылками на фрагменты в блоке программ, а первые элементы - номера недоопределенных ссылок из этих фрагментов. Номера здесь берутся нелокальные, т.е. те, которые фактически были введены текстовым редактором.

   В правом верхнем углу экрана прорисовывается красный прямоугольник, предупреждающий о режиме изменения программы. Если фрагмент имел операторы перехода, то желтый фон текущего оператора перехода заменяется на белый. Затем идет оператор "повторение", к которому будут происходить откаты при обнаружении ошибок либо при переходах к редактированию очередно-

го фрагмента вводимой ветви программы. После него выполняется обращение к текстовому редактору для набора программы фрагмента. Заметим, что это обращение сохраняет ранее имевшийся экран - чтобы можно было исправить обнаруженную ошибку. Если имел место отказ от редактирования, то расчищаются экран и буфер текстов, и далее - откат к перерисовке исходного вида редактируемого фрагмента. Иначе - обращение к оператору "кодтекста(1 x24 x25)", который преобразует содержимое буфера текстов в набор термов. Если это сделать не удается, то есть x25 отлично от 0, то x24 и x25 содержат информацию об ошибке, которая прорисовывается в верхнем правом углу экрана. Если ошибка локализуется, то перед откатом к текстовому редактору создается комментарий (теквхожд ...), определяющий позицию курсора для указания на ошибку.

В случае успешного определения набора термов x24 выполняется оператор "разверткапрограммы(x24)". Он устраняет условные обозначения в набранной программе, заменяя их на обозначаемые ими конструкции языка ЛОС. Во-первых, вместо обозначения "логсимвол(n)", используемого для указания на символьное число n, помещается это символьное число - логический символ, имеющий номер $260 + n$. Во-вторых, вместо обозначения "замена($xi\ t$)" помещается оператор "замена($i\ t$)".

Создается список x26 всех операторов перехода в новом фрагменте программы x24. Предпринимается просмотр операторов x27 данного списка и проверка, что каждый из них имеет лишь одно вхождение в список. Если это не так, то в правом верхнем углу экрана выдается сообщение об ошибке, и откат к повторному редактированию. Иначе - переход через "иначе 4".

Просматриваются все операторы перехода, имевшиеся в текущем фрагменте до его редактирования, и создаются: список x27 номеров $1, 2, \ldots$ этих операторов, а также список x28 пар однобуквенных термов - определяемых ими ссылок на подфрагменты.

Просматриваются все операторы перехода из списка x26. Каждый такой оператор имеет вид "ветвь(n)" либо "иначе(n)", где n - натуральное. Если n входит в список x27, то находится соответствующая пара термов t_1, t_2 из x28, и оператор заменяется на "ветвь($t_1\ t_2$)" либо "иначе($t_1\ t_2$)". Это означает, что через оператор будет выполнен переход к тому же подфрагменту, к которому он имелся через n-й из старых операторов перехода до редактирования. Иначе происходит замена десятичного числа n на его символьное представление.

Если вводимый фрагмент - корневой и стэк x22 пуст, то значение x29 заменяется на 0, а значение x27 - на заголовок редактируемой программы. Иначе x29 заменяется на 1, а x27, x28 - на пару логических символов, образующих ссылку в блоке программ на тот оператор перехода, который ведет к редактируемому фрагменту. Затем оператор "прогфайл(запись ...)" выполняет запись нового фрагмента программы x24 в блоке программ и запись ссылки на него из соответствующего надфрагмента (при x29 = 1) либо из каталога программ (при x29 = 0).

Далее переменной x17 (текущий фрагмент) присваивается пустой набор. Если редактировался корневой фрагмент ветви новых фрагментов (стэк x22 пуст), то ссылка x15,x16 на редактируемый фрагмент изменяется на новое значение, указанное оператором "прогфайл(запись ...)". Создается список x32 номеров

недоопределенных переходов из нового фрагмента (они упорядочены в направлении от конца фрагмента к началу). Если этот список непуст, то к началу стэка x22 добавляется набор, соответствующий недоопределенным переходам из нового фрагмента. В этом случае в левом верхнем углу экрана прорисовывается номер первого (по списку x32) такого перехода, экран расчищается ниже первой строки, и откат к набору подфрагмента. Если список x32 пуст, то анализируется стэк x22. От начала его последовательно отбрасываются наборы длины 3 (они уже обработаны). Если встречается набор длины, большей 3, то удаляется первый его элемент, в верхнем левом углу экрана прорисовывается первый из оставшихся элементов, экран расчищается ниже первой строки, и откат к набору подфрагмента. Если же стэк x22 оказывается пустым, то в верхней части экрана оставляется только заголовок программы, и откат к перерисовке корневого фрагмента новой ветви (т.е. новой версии того фрагмента, для которого было начато редактирование).

9. x21 = "модули". Нажата клавиша Ctr-Del для удаления ветви программы, начинающейся с выделенного оператора перехода. Эта ветвь, до выхода из редактора программ, не исключается из блока программ, а лишь отключается от программы. Прежде всего, проверяется, что фрагмент имеет операторы перехода и что накопитель x14 отключенных фрагментов пуст. Затем определяются значения x22, x23, x24, указывающие внешнюю ссылку в блоке программ на текущий фрагмент: если фрагмент корневой, то x22 равно 0, а x23 равно заголовку программы. Иначе x22 равно 1, а x23, x24 - ссылка в блоке программ на оператор перехода к данному фрагменту. После определения указанных значений - переход через "ветвь 2".

   Здесь определяется вхождение x25 текущего оператора перехода в текущий фрагмент программы x17 и проверяется, что этот переход определен. В накопитель x14 заносится тройка (ветвь x27 x28), ссылающаяся на корневой фрагмент удаляемой ветви программы.

   Предпринимается коррекция фрагмента программы x17, ориентированная на запись его в блоке программ; фактически это затрагивает лишь операторы "замена(...)". Определяется ссылка x27,x28 на вхождение в блок программ оператора перехода, ведущего к удаляемому подфрагменту текущего фрагмента, и предпринимается отключение этой ветви. Затем из текущего фрагмента x17 исключается рассматриваемый оператор перехода, и скорректированная версия этого фрагмента регистрируется в блоке программ вместо старой версии. Ссылка x15,x16 на текущий фрагмент корректируется на его новую версию, и откат к перерисовке.

10. x21 = "конъюнктконтекст". Нажата клавиша Ctr-F7. Выполняется удаление всей программы текущего логического символа. Проверяется, что накопитель отключенных фрагментов x14 пуст, и в него заносится ссылка на корневой фрагмент удаляемой программы.

11. x21 = "элементы". Нажата клавиша F8. Происходит поиск фрагмента программы текущего логического символа, содержащего заданный оператор. Такой оператор может быть либо заранее определен специальным комментарием к посылкам исходной задачи, и тогда инициированное нулем значение x22 заменяется на оператор, извлекаемый из соответствующего комментария, либо

(см. переход через "ветвь 3") искомый оператор вводится вручную с помощью текстового редактора, и снова x22 получает в качестве значения этот оператор. Далее - переход через "ветвь 2" к фрагменту, в котором находится оператор "прогфайл(имя x22 x25)". Этот оператор и осуществляет поиск фрагмента. Его выходной переменной присваивается путь в программе текущего символа до фрагмента с искомым оператором; формат описания пути - тот же, что и в случае поиска первого недоопределенного перехода. Стэк x12 переопределяется на найденный путь, вводится комментарий "автоклавиатура ..." к посылкам исходной задачи, определяющий такую последовательность нажатий клавиш, при которой искомый оператор в найденном фрагменте оказывается выделен многоцветной указкой, и откат к перерисовке этого фрагмента.

12. x21 = "узелраздела" либо x21 = "попыткапараметризации" (соответственно, клавиши F3 и F2). Переход к просмотру справочной информации о логическом символе. В верхней части экрана устанавливается меню для просмотра и редактирования справочной информации о символе. Дальнейшие действия выполняются процедурой "помощь", причем в случае F3 ей передается известный символ - заголовок текущей программы, а в случае F2 она организует ввод символа текстовым редактором.

13. x21 = "новаяветвь". Нажата клавиша Ctr-F6. В накопителе x14 находится первая ссылка на отключенную ветвь программы; эта ссылка удаляется из x14, удаляется вся программа текущего символа x13, и указанная выше ветвь становится программой символа x13.

14. x21 = "схемаидентификации". Нажата клавиша "ш" для перехода к просмотру программы следующего логического символа. Прежде всего, осуществляется удаление отключенных фрагментов программы текущего логического символа, если они имеются в буфере x14. Затем идет оператор "повторение", к которому будут происходить откаты в цикле поиска первого следующего за текущим символа, имеющего программу. Номер текущего символа x13 увеличивается на 1; если достигнут максимальный символ (пока в программе редактора программ указана константа 5000), то цикл обрывается, и откат к перерисовке. Иначе - проверяется, что символ имеет программу. Если имеет, то выполняются действия, необходимые для того, чтобы эта программа стала текущей и была прорисована при откате (в частности, в верхней части экрана прорисовывается новое название символа).

15. x21 - один из символов "прообраз", "подборзначений", "внутрвывод", "анализатор". Соответствующие клавиши - "П", "У", "у", "п". В случаях "п","П"запускается процедура проверки корректности программы (для "П" - серийная, начиная с текущего логического символа; для "п" - только проверка текущей программы). В случаях "у","У" происходит поиск фрагментов программы, которые могут оказаться неудаленными остатками ранее удаленных приемов ("У" - серийный поиск, "у" - однократный). Основные действия здесь выполняются процедурой "нормпрог", к которой сразу же и происходит обращение. Она выполняет общий контроль корректности текущей ветви программы и организует обращения к оператору "смпрог" в цикле сканирования операторов программы (напомним, что оператор "спрог" программируется вручную как средство поиска в программе либо автоматического изменения программы). В

случаях "у","У" оператору "нормпрог" передается информационный элемент "продолжение", инициирующий поиск концевых вхождений оператора "продолжение", подозрительных на наличие неудаленного остатка приема. Набор x22 содержит результаты проверки программы (типы элементов этого набора перечислены в справочной информации символа "результат").

Если найдены ошибка либо то вхождение оператора, для которого предпринимался поиск, то информация о них передается в элементы (стоп ...), (титр ...) набора x22. Первый из них содержит координаты оператора, второй - указывает на сопровождающий текст, который должен быть прорисован. Соответственно, установки на текущий фрагмент корректируются для прорисовки фрагмента с найденным оператором, вводится комментарий (автоклавиатура ...) для выделения оператора многоцветной указкой, и откат к перерисовке фрагмента.

В случае серийного запуска (переход через "ветвь 4" после анализа содержимого набора x22, если это содержимое не указывает на необходимость выдачи на экране полученной информации) - проверяется, не была ли в процессе вычислений нажата какя-либо клавиша. Если была, то цикл обывается. Иначе - вводится комментарий (авоклавиатура ...), определяющий нажатия клавиши "ш" (для перехода к следующему символу) и снова - той же клавиши x21.

16. x21 = "извлечениеварианта". Команда соответствует нажатию клавиши "т", которое в действительности является фиктивным, так как запускается эта команда обычно через комментарий (автоклавиатура ...) и служит для прорисовки в верхнем правом углу экрана текста согласно комментарию (титр ...), который вводится при проверке правильности программы.

17. x21 = "группировки" либо x21 = "старшийчлен". Соответственно, нажимаются клавиши End и "э". В случае клавиши "э" происходит откат к прорисовке корневого фрагмента программы текущего символа; предварительно удаляются все отключенные фрагменты, а также комментарий (контрольпрограммы ...), обеспечивавший возвращение в базу пприемов, если переход к редактору программ произошел из нее. В случае клавиши End - реализуется возвращение из редактора программ в тот внешний интерфейс, из которого к нему происходил переход. Здесь рассматриваются следующие случаи:

а) Исходная задача имела комментарий посылок "оператор". В этом случае реализуется возвращение в оглавление операторов ЛОСа.

б) Исходная задача имела комментарий посылок "контрсерия ...". В этом комментарии накапливается последовательность логических символов, к просмотру программ которых имели место переходы при нажатии клавиш "с" из редактора программ. Реализуется откат по этой цепочке символов на один шаг - возвращение к просмотру корневого фрагмента программы первого из символов цепочки и отбрасывание начала цепочки.

в) Исходная задача имела комментарий посылок "контрольпрограммы ...". В этом случае имел место переход от просмотра приема ГЕНОЛОГа к его программе, и реализуется возвращение в просмотр приема.

В остальных случаях предпринимается откат к перерисовке корневого фрагмента программы и расчистка накопителя x14.

18. x21 = "фиктопер". Нажата клавиша F6. Реализуется операция объединения двух последовательно идущих фрагментов программы в один фрагмент. Первый из них заканчивается операторами "ветвь N", "продолжение", где N - ссылка на второй фрагмент, либо операторами P, "иначе N", "продолжение". Прежде всего, программа проверяет, что последние операторы текущего фрагмента x17 действительно имеют указанный выше вид. Затем находится набор x25 операторов второго фрагмента программы. В зависимости от вида окончания фрагмента x17, создается результат x27 объединения фрагментов x17 и x25 (в случае оператора "иначе(N)" оператор P заменяется на свое отрицание). Все непосредственные подфрагменты фрагмента x25 отключаются, но сохраняются в блоке программ. Отключение необходимо для того, чтобы при последующем удалении фрагмента x25 не оказались удаленными и его подфрагменты, на которые ссылки будут делаться также из нового фрагмента x27. После отключения находится ссылка из надфрагмента на фрагмент x17 (в случае корневого фрагмента - ссылка из каталога программ), и с помощью оператора "прогфайл(запись ...)" выполняется запись по данной ссылке нового фрагмента x27. Автоматически при этом удаляются старые фрагменты x17 и x25. Далее - откат к перерисовке текущего фрагмента.

19. x21 = "копияблокаанализа". Нажата клавиша F4. Выполняется поиск всех вхождений в текущую ветвь программы терма либо логического символа, вводимого в окне текстового редактора. Для поиска очередного вхождения каждый раз нажимается F4.

    Сначала проверяется наличие комментария (клавиатураоператора ...). Такой комментарий вводится при обнаружении в некотором фрагменте просматриваемой ветви нужного терма либо логического символа A. Он сохраняет список операторов фрагмента, вхождение в этот список того оператора, внутри которого расположено найденное вхождение A, и само это вхождение. Если данный комментарий имеется, то он используется для продолжения просмотра при поиске следующего вхождения A. Чтобы определить, какой терм или логический символ нужно найти, далее извлекается комментарий (перечислфрагментов ...) к посылкам исходной задачи. Он содержит информацию о пути к корню просматриваемой ветви от корня программы; информацию о пути от корня просматриваемой ветви к ее текущему фрагменту, а также список термов либо логических символов, которые нужно искать. В соответствии с этой информацией и предпринимается продолжение просмотра фрагмента, указанного в комментарии (клавиатураоператора ...). После обнаружения очередного искомого вхождения - вводится комментарий (автоклавиатура ...), определяющий последовательность нажатий клавиш для входа в выделение найденного вхождения многоцветной указкой, и откат к обработке очередной команды. Если же просмотр фрагмента завершился безрезультатно, то комментарий (клавиатураоператора ...) удаляется, и переход через "ветвь 2". Если комментария (клавиатураоператора ...) вообще не было, то сразу же - переход через "ветвь 2".

    Здесь переменная x22 инициализируется нулем, после чего ей переприсваивается комментарий (перечислфрагментов ...), если он уже введен к данному моменту. Если в данном комментарии еще не был указан путь от корня программы к корню просматриваемой ветви, то в качестве такого пути регистрируется x12. Далее - переход через "ветвь 1", где переменная x23 инициализируется нулем; если x22 было ненулевым, то из него извлекается список искомых термов

(в данном интерфейсе он всегда одноэлементный), и его элемент переприсваивается переменной x23.

В случае нулевого x22 реализуется интерфейс ручного ввода искомого терма; этот терм переприсваивается переменной x23. Затем - переход через "ветвь 1".

Переменной x24 присваивается путь от корня просматриваемой ветви к текущему фрагменту. Затем предпринимается обращение к оператору "перечислфрагментов(...)", выполняющему перечисление следующих за текущим фрагментов данной ветви. Набор операторов очередного такого фрагмента передается выходной переменной x26. В списке x26 предпринимается поиск очередного искомого вхождения. Операторы перехода исключаются из поиска, так как в них случайным образом может оказаться искомый логический символ, используемый в качестве кода ссылки на подфрагмент. Если вхождение найдено, то вводится соответствующий комментарий (клавиатураоператора ...). Вводится либо корректируется на найденное вхождение комментарий (перечислфрагментов ...); стэк x12 переустанавливается на путь к найденному фрагменту; переменные x15,x16,x11 изменяются для просмотра этого фрагмента; вводится комментарий (автоклавиатура ...), определяющий последовательность нажатий клавиш, необходимую для выделения найденного вхождения многоцветной указкой, и откат к перерисовке фрагмента. Если же просмотр ветви завершился безрезультатно, то переход через "иначе 1".

Экран расчищается, и предпринимается контрольное обращение к клавиатуре. После нажатия любой клавиши - проверяется, равно ли x22 нулю. Если равно, то сразу откат к перерисовке текущего фрагмента (он не изменялся и является по-прежнему корневым фрагментом просматриваемой ветви). Иначе - перед откатом к перерисовке восстанавливаются значения x12,x15,x16,x11, соответствующие корневому фрагменту ветви, и удаляется комментарий (перечислфрагментов ...).

20. x21 = "внешдизъюнкция". Нажата клавиша Insert. Удаляется, если он был, комментарий (копия ...) к посылкам исходной задачи, и вводится новый такой комментарий, ссылающийся на текущий фрагмент. Этот комментарий используется в качестве буфера для копирования ветви программы. Команда создания на его основе копии ветви программы дается через режим просмотра подтермов операторов (см. ниже).

21. x21 = "существпосылки". Нажата клавиша Ctr-a (кир.). Это - переключатель для работы с альтернативным программным блоком, расположенным в поддиректории ALT. Он вводит либо удаляет комментарий "альтернатива" к посылкам исходной задачи, указывающий на работу с альтернативным блоком программ. Затем расчищается буфер отключенных фрагментов x14, перерисовывается главное меню, и выполняется откат ко вводу заголовка программы.

22. x21 = "истинно". Нажата клавиша Ctr-ъ. Если имеет место режим работы с альтернативным блоком программ, то создается ссылка на текущий фрагмент для копирования его в основной блок программ. Роль такой ссылки играет комментарий (копияфайла A) к посылкам исходной задачи. Если имеет место режим работы с основным блоком программ, то находится комментарий (копияфайла A). Если стэк x12 пуст, то реализуется перенесение из альтернативного блока программ ветви, ссылка на корень которой есть A, в основной блок программ,

причем эта ветвь становится программой текущего логического символа. Если же x12 непусто, то находится ссылка x27 на оператор перехода, ссылающийся на текущий фрагмент, и вся ветвь текущего фрагмента заменяется на ветвь, переносимую из альтернативного блока программ по ссылке A. В обоих случаях собственно перенесение ветви осуществляется оператором "копияветви(...)".

23. x21 - один из символов "арккосинус", "переобозначение", "транслвыражения", "кортеж". Соответственно, нажаты клавиши Ctr-л, Ctr-п,Ctr-ф,Ctr-т. Здесь реализуется обращение к одному из справочных оглавлений: оглавлению операторов ЛОСа, оглавлению информационных элементов программного блока (компилятор ГЕНОЛОГа), оглавлению типов комментариев к представлению слова либо терма фразы, и оглавлению типов комментариев к теореме в базе теорем. Все эти оглавления располагаются в 5-м информационном блоке, так что прежде всего этот блок активизируется. Находится указатель x23 типа оглавления, и находится одноэлементный набор x24, состоящий из ссылки на корень оглавления (если оглавления еще не существовало, то оно при этом вводится). Далее происходит обращение к процедуре "оглавление". После выбора концевого пункта оглавления - экран расчищается, в верхней части его перерисовывается текст выбранного пункта, и под текстом - горизонтальная черта. Затем находится содержимое логического терминала выбранного концевого пункта оглавления. Этим содержимым является терм "прием(...)", первые два операнда которого суть логический символ и номер узла этого символа. Находится логический терминал, достижимый из данного узла по метке "терм" - его содержимое и прорисовывается под горизонтальной чертой в качестве информации, сопровождающей текст пункта оглавления. Затем - переход через "ветвь 6" к небольшому обработчику команд, позволяющему войти в текстовый редактор для изменения сопровождающей информации (x33 = "тринадцать"), а в случае оглавления операторов ЛОСа - также просмотреть информацию о заголовке оператора (x33 = "внешконъюнкция"). По окончании работы со справочным оглавлением - откат к оператору "ветвь 2"; далее активизируется первый информационный блок, расчищается верхняя часть экрана, в которой восстанавливается прорисовка текущего логического символа x13, и откат к перерисовке текущего фрагмента.

24. x21 = "нормвариант". Нажата клавиша Ctr-F8. Происходит контроль блока программ на уровне интерпретатора ЛОСа. Основная работа выполняется оператором "прогфайл(контроль x22)", определяющим набор x22 наборов (номер файла блока программ - тип ошибки - A_1 - A_2), где (A_1, A_2) - ссылка на фрагмент, в котором найдена ошибка. На практике эта возможность контроля практически не используется, так как контроль блока программ автоматически осуществляется при уплотнении либо сохранении программы решателя. Первоначально предполагалось создать специальный интерфейс коррекции ошибок блока программ, возникших на уровне интерпретатора, с помощью самого решателя. Однако, пока это направление не получило сколь-нибудь заметного развития, а при обнаружении поломок в файлах на указанном уровне предпринимается просто извлечение резервной копии из директории SLCOPY.

25. x21 = "правмногоугольник". Нажата клавиша "я". Предпринимается регистрация ветви текущего фрагмента в контейнере - информационном блоке номер

12, используемом для перенесения информации из одной версии логической системы в другую.

26. x21 = "путьвхождения". Нажата клавиша Ctr-F9. Дополнение к команде Ctr-F8 - удаляется найденная изолированная ветвь в блоке программ. На протяжении всего периода работы с системой эта команда практически не использовалась.

27. x21 = "стандменьше". Нажата клавиша F1. Происходит обращение к справочнику по логической системе.

28. x21 = "мышь". Нажата левая либо правая кнопка мыши. Прежде всего, переменной x22 присваивается указатель на левую (0) либо правую (2) клавишу мыши. Переменным x23,x24 присваиваются столбец и строка текущей позиции курсора. Далее проверяется, что курсор мыши находится ниже верхней текстовой строки. Вычисляется номер x30 позиции в буфере текстов, соответствующей позиции курсора мыши. Проверяется, то либо на позиции x30, либо на следующей за ней позиции расположен символ с "непустой" прорисовкой. Находится начальная позиция в буфере текстов того слова, к которому относится x30 (если x30 приходится на оператор перехода, то определяется начальная позиция этого оператора); x30 присваивается номер этой начальной позиции. Далее - переход через оператор "ветвь 3".

    Если была нажата левая кнопка мыши, то вводится комментарий (умножсимв x30) к посылкам исходной задачи - указатель той позиции в буфере текстов, которая является началом выделяемого многоцветной указкой подтерма. Затем формируется комментарий (автоклавиатура ...) для перехода в режим просмотра подтермов операторов, и откат к запросу очередной команды.

    Если была нажата правая кнопка мыши, то с помощью набора x18 определяется тот оператор перехода, начальной позицией которого в буфере текстов служит x30. Затем создается комментарий (автоклавиатура ...) для такой цепочки нажатий клавиш "курсор влево" либо "курсор вправо", завершаемой нажатием "курсор вниз", которая сделает текущим выделенный оператор перехода и обеспечит переход через него.

    Наконец, при нажатии правой кнопки мыши вне оператора перехода - переход через "ветвь 2" из исходного фрагмента обработки данной команд, где формируется комментарий (автоклавиатура ...) для нажатия клавиши "курсор вверх", что соответствует возвращению в надфрагмент.

29. x21 = "арктангенс". Нажата клавиша Ctr-д. Предпринимается вход в оглавление блоков диалога и создание шаблона нового такого блока либо редактирование ранее созданного. Сначала активизируется 5-й информационный блок, в котором находится данное оглавление. Затем находится корневой указатель оглавления блоков диалога (тип этого оглавления - логический символ "нормистина"), и происходит обращение к оглавлению. При выборе концевого пункта оглавления проверяется, что он содержит ссылку на узел ранее созданного блока диалога. Если это не так - переход через "иначе 6", где с помощью простейшего датчика случайных чисел (остаток от деления на 1000 номера текущего шага работы интерпретатора) выбирается тот логический символ, для которого будет введен узел нового (пока пустого) шалона блока диалога. Такой узел вводится, и откат к повторному поиску ссылки на узел выбранного концевого

пункта оглавления. Затем - обращение к оператору "новыйдиалог(x27 x28)", реализующему интерфейс редактирования блока диалога.

30. x21 = "арксинус". Нажата клавиша Ctr-o (кир.). Организуется переход к тому пункту оглавления операторов ЛОСа, который соответствует заголовку просматриваемой программы.

31. x21 = "фигуры". Нажата клавиша Ctr-Enter. Происходит запуск программы текущего логического символа "пуск".

Мы проследили выполнение команды входа в режим просмотра подтермов операторов (нажатие клавиши "о") вплоть до обращения к оператору "автоклавиатура(x31)", запрашивающему очередную команду x31 данного режима. Рассмотрим реализацию процедур обработки команды x31. Для входа в фрагмент, содержащий оператор "автоклавиатура(x31)", можно использовать пункт "Точка входа в обработчик команд просмотра операторов в фрагменте ..." подраздела "редактор ЛОСа" оглавления программ.

Напомним смысловую нагрузку переменных, используемых в режиме просмотра подтермов операторов. x22 - набор из пяти цветов, используемых при выделении подтермов. x23 - стэк, сохраняющий наборы четверок (исходная позиция буфера текстов для перекраски - заключительная позицию для перекраски - столбец - строка для выдачи надтерма на экране) для перекраски надтермов текущего терма. x24 - вхождение текущего цвета в набор x22. x21 - пара (вхождение текущего выделенного оператора в фрагмент программы - указатель типа оператора (оператор перехода - 1, иначе - 0)). x25 - вхождение выделенного подтерма текущего оператора. x26 - указатель направления перемещения при просмотре подтермов одного уровня (1 - влево, 0 - вправо). x27, x30 - начало и конец выделенного подтерма в буфере текстов. x28, x29 - столбец и строка начала этого подтерма на экране.

Сразу после получения команды x31 определяется значение x32 цвета фона надтерма текущего терма.

1. x31 = "копияблокаанализа" либо x31 = "частичныйответ". Соответственно, нажата клавиша F4 для поиска очередного вхождения заданного терма либо клавиша "о" (кир.) для возвращения в обычный режим просмотра фрагмента программы. В первом случае проверяется наличие комментария посылок (перечислфрагментов ...), в котором сохраняются данные о виде искомого терма и области поиска. Если выделенный текущий подтерм - некорневой, то значения x27 - x30 переустанавливаются на выделение всего текущего оператора, в котором выделен этот подтерм. Если была нажата клавиша F4, то вводится комментарий (автоклавиатура ...), который после возвращения в обычный режим вызовет очередное нажатие F4 для поиска очередного вхождения терма. Иначе - удаляются комментарии (клавиатураоператора ...) и (перечислфрагментов ...), если они были; в последнем случае, строго говоря, комментарий не удаляется, а лишь обнуливаются его второй и третий разряды, чтобы впоследствии можно было использовать его для поиска того же терма в другой части программы. Далее происходит восстановление белого цвета фона текущего оператора в буфере текстов и перерисовка этого оператора. Если текущий фрагмент имел операторы перехода, то согласно значению x19 восстанавливается желтый цвет фона текущего оператора перехода. Восстанавливается меню обычного режима просмотра, и откат к запросу команды этого режима.

2. x31 = "внешконъюнкция". Нажата клавиша "курсор вправо"для выделения следующего подтерма текущего надтерма либо следующего оператора фрагмента программы. Определяется позиция x33 буфера текстов, следующая за крайней правой позицией выделенного подтерма. До тех пор, пока на позиции x33 оказывается символ пробела, происходит сдвиг x33 вправо. Если при этом выходим на пустую зону буфера текстов либо на закрывающую скобку, то выполнение команды отменяется. Иначе - переход через "иначе 2".

   Здесь восстанавливается цвет фона x32 текущего подтерма (он сливается с надтермом). Затем значения x28,x29,x27,x25,x21 изменяются таким образом, что указывают на подтерм с началом x33; x26 заменяется на 0 (движение вправо), и откат к изменению цвета фона текущего подтерма.

3. x31 = "циклвариантов". Нажата клавиша "курсор влево". Аналогично предыдущему. Проверяется, что начало x27 выделенного подтерма в буфере текстов не является началом этого буфера, после чего находится левый сосед x33 позиции x27. Значение x33 сдвигается влево, пока не появляется символ, отличный от пробела. Если этот символ - открывающая скобка, то команда отменяется. Иначе - цвет фона текущего подтерма изменяется на x32; определяющие выделение подтерма переменные корректируются на левого соседа текущего подтерма (он задается своей последней позицией x33), x26 присваивается 1, и откат к перекраске.

4. x31 = "подстановка". Нажата клавиша "курсор вниз". Проверяется, что текущий подтерм по вхождению x25 неоднобуквенный. Затем переменной x33 присваивается номер начальной позиции x27 этого подтерма в буфере текстов, и начинается цикл сдвигов x33 вправо до обнаружения открывающей скобки (очередной шаг сдвига - откат к оператору "повторение"). После обнаружения открывающей скобки (x34 = "соединение") - переменной x37 присваивается номер позиции буфера текстов, следующей за позицией открывающей скобки. В стэк x23 заносится четверка, соответствующая текущему подтерму; переменные x28,x27,x24,x25 корректируются для выделения первого операнда текущего подтерма, и откат к изменению цвета фона этого операнда.

5. x31 = "внешсумма". Нажата клавиша "курсор вверх". Из первого элемента стэка x23 извлекаются данные для выделения надтерма текущего терма, а сам этот элемент удаляется из стэка. Весь надтерм перерисовывается в своем цвете фона; значения x25,x27 - x30,x24 корректируются для ссылки на этот надтерм, и откат к запросу очередной команды.

6. x31 = "Стандплюс" либо x31 = "легковидеть". Соответственно, нажаты клавиша "к" либо "в" (кир.). В первом случае находится комментарий (копия A) к посылкам исходной задачи, который сразу же удаляется; во втором случае - извлекается элемент (ветвь A) из накопителя x14. В обоих случаях A - пара логических символов, ссылающаяся на некоторый фрагмент программы (во втором случае - отключенный). Если была нажата клавиша "в", то реализуется диалог выбора типа перехода к отключенной ветви. В верхней правой части экрана прорисовывается текст "ветвь - иначе", в котором левое слово выделено зеленым цветом фона. Переменная x27 будет при реализации данной команды (случай "в") играть роль указателя на тип оператора перехода к подключаемой

ветви; изначально ей присваивается символ "ветвь". Далее - обращение к клавиатуре для получения команды x39. Если x39 = "тринадцать" (клавиша Enter), то верхняя правая часть экрана расчищается, и выход из диалога выбора типа оператора перехода. Если x39 - "внешконъюнкция" (клавиша "курсор вправо") либо "циклвариантов" (клавиша "курсор влево"), то зеленый фон переносится на правое либо, соответственно левое слово текста, и соответственно изменяется x27. Наконец, в случае x39 = "Плюс" (клавиша Esc) - отмена команды.

Далее - переход через "ветвь 2", где в случае "в" происходит исключение из накопителя x14 пары (ветвь A). Переменным x23,x24 присваиваются значения, адресующие точку ссылки на изменяемый фрагмент программы. Если текущий фрагмент программы - корневой, то значением x23 становится заголовок программы, а x24 - 0. В случае некорневого фрагмента пара (x23,x24) - ссылка на оператор перехода надфрагмента в блоке программ, по которому осуществляется переход к текущему фрагменту.

Если была нажата клавиша "к", то далее происходит копирование ветви по ссылке A и вставка ее вместо ветви текущего фрагмента. Последняя удаляется. Значения x15,x16 устанавливаются для ссылки на новую ветвь, и откат к перерисовке текущего фрагмента - первого фрагмента новой ветви.

Если была нажата клавиша "в", то находится результат x34 вставки в текущий фрагмент перед текущим оператором оператора перехода типа x27, ссылающегося на начало ветви A. После исключения из x34 вспомогательных обозначений происходит изменение текущего фрагмента на фрагмент x34. Корректируются x15,x16,x11, и откат к перерисовке фрагмента.

7. x31 = "числитель". Нажата клавиша "р" (кир.) для разрезания текущего фрагмента на две части (выделенный оператор становится началом второй части). Происходит исключение из x17 вспомогательных обозначений и формирование значений x23,x24, определяющих ссылку на текущий фрагмент из надфрагмента - аналогично предыдущей команде. Далее - переход через "ветвь 2".

   Создается набор x33 всех операторов текущего фрагмента, предшествующих выделенному оператору. После добавления к концу x33 оператора перехода "ветвь(0)" и оператора "продолжение", получается набор x35 операторов новой версии текущего фрагмента. Определяется набор x36 операторов старой версии текущего фрагмента, расположенных начиная с выделенного оператора. Все ветви, достижимые из текущего фрагмента по операторам перехода, не вошедшим в x35, отключаются и сохраняются в блоке программ. Это делается для того, чтобы они не пропали при замене текущего фрагмента на x35. Далее предпринимается регистрация в блоке программ новой версии текущего фрагмента вместо старой, и сразу после этого - подключение фрагмента x36 по ссылке "ветвь(0)" из x35. Отключенные выше фрагменты здесь снова становятся подключенными к программе. Корректируются значения x15,x16,x11, и откат к перерисовке фрагмента.

8. x31 = "извлечениеварианта". Нажатие клавиши "т", которое, как и в случае обычного режима, является фиктивным. Данная команда запускается через комментарий (автоклавиатура ...) для прорисовки в верхней части экрана текста согласно комментарию (титр A), создаваемому при обнаружении особой точки в процессе просмотра ветви программы.

9. x31 = "конст" либо x31 = "группировки". Нажата клавиша Ctr-End либо, соответственно, End. В первом случае происходит поиск в оглавлении программ концевого пункта, к ветви которого относится выделенный оператор; во втором - возвращение в оглавление программ, если переход к редактору программ произошел оттуда.

   Если исходная задача имеет комментарий (подфрагмент ...), означающий, что переход к редактору программ произошел от оглавления программ, то этот комментарий удаляется; проверяется, что была нажата клавиша End, и вводится комментарий "прогртерм", инициирующий при откате к главному меню вход в оглавление программ. Удаляются отключенные фрагменты. Ссылка на программу текущего символа переустанавливается по блоку программ - эта переустановка предотвращает использование старых версий фрагментов программы, сохраняющихся в зоне программ, если они были заменены при редактировании на новые версии. Затем - откат к обработчику команд главного меню, который и осуществит (без прорисовки этого меню и запроса команды) переключение на оглавление программ.

   Если же исходная задача не имела комментария (подфрагмент ...) либо была нажата клавиша Ctr-End, то предпринимается просмотр всех позиций x36 в текущем фрагменте либо его надфрагментах, из которых достижима позиция x35 текущего выделенного оператора. Если при этом обнаруживается оператор "прием(N)", то в 9-м информационном блоке (т.е. блоке оглавления программ) находится N - й узел заголовка программы x13. Его логический терминал "оглавление" содержит путь в оглавлении программ к концевому пункту, соответствующему найденному оператору. Этот путь становится текущим путем оглавления программ, вводится комментарий "прогртерм", и далее - те же действия, что и в предыдущем случае.

10. x31 = "точкапривязки". Нажата клавиша Page Down. В этом случае должна быть выполнена операция вставки перед выделенным оператором оператора "прием(N)" и создание ссылки на этот оператор из нового концевого пункта оглавления программ. Прежде всего, удаляется, если он был, комментарий (подфрагмент ...) и вводится комментарий "прогртерм", обеспечивающий переключение к оглавлению программ от точки запроса команд в главном меню. В отличие от случая x31 = "группировки", перед откатом к этой точке создается комментарий (вставка ...), сохраняющий информацию, необходимую для автоматической вставки оператора "прием(N)" и создания ссылки на него после ввода нового концевого пункта оглавления программ. Использование этого комментария было описано выше в разделе, посвященном программе "оглавление".

11. x31 = "анализатор". Нажата клавиша "п". Реализуется процедура увеличения либо уменьшения номеров всех программных переменных, больших или равных заданного номера N, в операторах, достижимых из текущего оператора. Процедура заблокирована в случае программы символа "вход". Прежде всего, происходит обращение к текстовому редактору для ввода в левой верхней части экрана терма "плюс(xN k)" либо "минус(xN k)", указывающего величину k и направление (увеличение, уменьшение) сдвига номеров переменных. x36 становится равно переменной xN; x35 - сначала десятичному, а затем символьному числу k. Далее инициализируются переменные x37, x38, x39, используемые в

цикле просмотра ветви текущего оператора. Значением x37 будет ссылка на текущий просматриваемый фрагмент этой ветви; x38 - стэк, являющийся непосредственным продолжением стэка x12. Его элементами будут тройки, первые два разряда которых - ссылка на фрагмент, а последний - номер перехода из фрагмента. Наконец, x39 - текущий просматриваемый фрагмент. К оператору "повторение" будут происходить откаты для рассмотрения очередного фрагмента ветви. Далее расположен оператор, перечисляющий позиции x40 в текущем фрагменте операторов x41, для которых производится сдвиг номеров переменных. Сначала в x41 выполняется изменение номеров явно указанных переменных, затем - переменных xi, неявно упоминаемых в операторах "замена(i t)". После просмотра фрагмента - переход через "иначе 3".

Здесь устраняются вспомогательные обозначения в фрагменте x39 и происходит его регистрация в блоке программ вместо старой версии. Затем - переход через "ветвь 1" к циклу поиска первого оператора перехода в x39, относящегося к области изменения номеров переменных. В стэк x38 заносится тройка для перехода от x39 по найденному оператору. x39 заменяется на фрагмент, достижимый по данному оператору, и откат к его обработке. Если оператор перехода не найден, то переход через "иначе 1".

Здесь происходит откат по стэку x38 к первому фрагменту, у которого имеется еще не обработанный оператор перехода. Если такого не найдено, то выполнение команды завершается, и откат к перерисовке фрагмента. Иначе - x37, x38, x39 корректируются для обработки фрагмента, достижимого через найденный оператор перехода, и откат к изменению в этом фрагменте номеров переменных.

12. x31 = "узелраздела". Нажата клавиша F3. Происходит просмотр справочной информации о заголовке x33 выделенного оператора либо операторного выражения.

13. x31 = "мышь". Нажата левая либо правая кнопка мыши. Определяются указатель x33 этой кнопки (0 - левая, 2 - правая)и координаты курсора мыши (x34,x35). Если нажата правая кнопка, то создается комментарий (автоклавиатура ...) для нажатия клавиши F3. В случае левой кнопки - определяется выделенный курсором мыши оператор и создается комментарий (умножсимв ...) для перемещения многоцветной указки на этот оператор, после чего - откат. Если курсор мыши не указывал на оператор, то переход через "ветвь 3", где создается комментарий (автоклавиатура ...) для возвращения в обычный режим просмотра.

14. x31 = "усмчисло". Нажата клавиша "с". Находится текущий выделенный многоцветной указкой символ x33, и вводится комментарий (прогрблок x33) для перехода к просмотру корневого фрагмента программы этого символа. Вводится либо пополняется значением x13 стэк "пройденных" логических символов. Затем - откат к прорисовке корневого фрагмента программы символа x33.

15. x31 = "схемаидентификации". Нажата клавиша "ш". Нижняя часть экрана расчищается, вводится комментарий (автоклавиатура ...) для команды, продолжающей инициированный в обычном режиме поиск (клавиши "П" либо "У"), и откат к обработчику команд обычного режима просмотра.

Глава 15

Программа отладчика ЛОСа

Отладчик ЛОСа реализован программой символа "прерывание". В действительности его роль гораздо большая, чем просто отладка или пошаговый показ решения. Так как обращения к отладчику могут происходить, после установки соответствующего типа прерывания, практически в любом месте выполняемой программы, он становится той точкой, в которой естественным образом можно аккумулировать различного рода самоконтроль и самоанализ. Программа "прерывание" достаточно сложна; приводимое ниже ее описание может оказать существенную помощь при развитии интерфейсов, связанных с решением задач, а также при создании дополнительных средств отладки и контроля.

15.1 Предварительные сведения об интерпретаторе ЛОСа

Начнем с предварительных сведений об интерпретаторе ЛОСа, необходимых для понимания работы отладчика. При работе системы в оперативной памяти создаются следующие основные массивы:

1. Массив для хранения термов и наборов, создаваемых операторами ЛОСа. В частности, в нем хранятся все задачи текущей цепи задач. Этот массив называется зоной задач. Размеры зоны задач сравнительно невелики - они позволяют хранить всего около 250000 элементарных знаков (логических символов; переменных; ссылок на термы, наборы и вхождения). Однако, для потребностей логического процессора эти размеры оказываются более чем достаточны. Нормальным режимом работы системы является режим с "почти пустой" зоной задач; при этом поиск свободного места для записи новых термов и наборов происходит за время, пренебрежимо малое по сравнению с полным временем реализации очередного оператора. Для исключения из зоны задач неиспользуемых объектов ("мусора") используется следующая простая схема. Каждый объект имеет пометку четности номера цикла заполнения зоны задач. Те объекты, пометка которых не совпадает с четностью текущего цикла (зона на каждом цикле заполняется слева направо), рассматриваются как пустая область. Как только зона оказывается заполнена, по цепочкам ссылок выявляются все фактически используемые объекты (их мало), и их пометка четности меняется на противоположную. При переходе к очередному циклу все прочие объекты автоматически оказываются пустой областью.

2. Массив для фрагментов программ, считываемых из файлов блока программ и сохраняемых на некоторое время в оперативной памяти. Он называется зоной программ и представляет собой буфер, обеспечивающий необходимое быстродействие системы. Размеры зоны программ - около 2.5 Мбайт; она разбита на сегменты по 256 Кбайт. В блоке программ фрагменты хранятся в несколько сжатом виде, а при считывании в зону программ они разворачиваются так, чтобы стала возможной быстрая работа интерпретатора.

3. Массив для хранения значений программных переменных и другой информации, необходимой для выполнения программы. Этот массив называется глобальным стэком интерпретатора ЛОСа и разбит на кадры. Новый кадр вводится при обращении к программе оператора либо операторного выражения, либо при обращении к решению задач. Он сохраняет значения программных переменных на период выполнения программы (решения задачи). Если программа реализует перечисляющий оператор, то ее кадр сохраняется даже после получения результата - он будет использоваться при откатах. Наличие таких теневых кадров, используемых при откатах, нарушает обычную "стэковую" структуру глобального стэка, который, строго говоря, стэком уже не является. Новые кадры могут возникать в глобальном стэке также при выполнении тех перечисляющих операторов, которые реализованы непосредственно интерпретатором. Они хранят информацию, необходимую для продолжения перечисления при откатах. Более подробное описание структуры кадров глобального стэка приводится ниже.

4. Массив для накопления входных данных реализуемого интерпретатором ЛОСа оператора либо операторного выражения. Этот массив называется стэком выражений, и в отличие от глобального стэка он уже является обычным стэком. При вычислении значения операторного выражения результат снова заносится в стэк выражений, после того как из него были исключены входные данные. Кроме обычных преимуществ, предоставляемых при организации вычисления значения многоэтажного операторного выражения, данный стэк играет роль буфера, предохраняющего промежуточные записи от удаления при периодически проводимых расчистках зоны задач.

Значением программной переменной ЛОСа, как уже говорилось выше, служит объект одного из четырех типов: логический символ, символ переменной, терм либо набор, вхождение в терм либо набор. Это значение кодируется при помощи одного 32-битного слова, которое и сохраняется в кадрах глобального стэка либо в стэке выражений. Глобальный стэк и стэк выражений рассматриваются далее как наборы 32-битных слов.

В данном разделе нам понадобится более подробное описание структуры кадров глобального стэка. Эти кадры бывают 4-х типов. Первые 6 ячеек каждого кадра хранят следующую стандартную информацию:

В первой ячейке записано смещение относительно начала стэка начала следующего кадра.

Вторая ячейка хранит смещение начала предыдущего кадра (для первого кадра стэка - 0).

Третья ячейка хранит тип кадра - число от 1 до 4.

Четвертая ячейка кадра хранит смещение относительно начала стэка источника этого кадра. Источником кадра называется кадр того фрагмента программы, при вы-

полнении которого был введен данный кадр (и к которому нужно будет возвращаться для продолжения вычислений). С учетом наличия сохраняемых для продолжения перечисления кадров, источник кадра не обязательно должен быть расположен непосредственно перед ним.

Пятая и шестая ячейки кадра хранят адрес продолжения - информацию, необходимую для возобновления вычислений в источнике этого кадра. Младшие 16 бит пятой ячейки хранят номер сегмента M зоны программ, в котором хранится фрагмент программы F, по которому ведутся вычисления в источнике; старшие 16 бит пятой ячейки - смещение начала фрагмента F в массиве M. Младшие 16 бит шестой ячейки хранят смещение ячейки фрагмента F, с которой должны возобновляться прерванные вычисления для источника. Наконец, старшие 16 бит шестой ячейки хранят значение, которое должно быть установлено в специальном регистре "уровня перечисления" при возобновлении вычислений в источнике.

Назначение дальнейших ячеек кадра глобального стэка зависит от его типа:

1. Кадр типа 1. Этот кадр используется при реализации программы, осуществляющей сканирование задачи. Он называется также кадром задачи.

   Седьмая ячейка кадра задачи хранит так называемый отложенный уровень сканирования. Обычно он равен нулю и используется в тех случаях, когда при сканировании задачи встречается условие либо посылка, вес которой на единицу больше текущего уровня сканирования U. Тогда (см. раздел "Сканирование задачи" в главе, посвященной описанию общей схемы функционирования решателя) предпринимается серия локальных просмотров такого условия (посылки), в процессе которых текущий уровень меняется от 0 до $U-1$. На период этих просмотров отложенный уровень сохраняет исходное значение текущего уровня U, а затем опять становится равен 0. Его ненулевое значение является указателем на наличие серии локальных просмотров.

   Восьмая ячейка кадра задачи хранит в младших 16 битах максимальный уровень задачи, а в старших 16 битах - смещение вершины стэка выражений по отношению к его корню на момент ввода кадра задачи.

   Девятая ячейка кадра задачи хранит смещение по отношению к началу того же кадра первой не определенной на текущий момент программной переменной (все относящиеся к тому же кадру последующие ячейки также соответствуют не определенным программным переменным).

   Ячейки кадра задачи начиная с десятой хранят значения программных переменных x1,x2, и т.д. В частности, сама десятая ячейка хранит ссылку на задачу данного кадра.

   Переменная, значение которой не определено, представляется в кадре глобального стэка одним из трех способов:

   а) нулевой набор;

   б) набор, старшие 16 бит которого имеют вид $(10\ldots0)$, а младшие 16 образуют номер переменной в источнике кадра, которой будет присваиваться значение данной (выходной) переменной текущего кадра;

   в) набор, старшие 14 бит которого имеют вид $(010\ldots0)$, а младшие 18 образуют смещение в зоне задач позиции набора, на которую заносится значение данной (выходной) переменной текущего кадра. Этот способ передачи выходных значений используется при обращении к оператору "процедура(...)".

2. Кадр типа 2. Этот кадр используется при реализации оператора, запрограммированного на ЛОСе; он называется кадром оператора.

   Седьмая ячейка данного кадра хранит "начало расчистки". Оно используется в том случае, если оператор реализуется в режиме перечисления. Тогда при откате к продолжению определяемого им перечисления выполняется обнуление всех ячеек источника кадра оператора, начиная с ячейки, смещение которой (по отношению к началу источника) равно "началу расчистки".

   Восьмая ячейка хранит значение, переносимое в регистр "уровня перечисления" при откате к продолжению перечисления.

   Девятая ячейка хранит смещение относительно начала кадра первой не определенной переменной.

   Ячейки кадра начиная с десятой хранят значения программных переменных x1,x2,....

3. Кадр типа 3. Этот кадр используется при реализации операторного выражения, запрограммированного на ЛОСе; он называется кадром операторного выражения.

   Седьмая ячейка данного кадра хранит код логического символа - индекса обращения к программе (логический символ 0 в случае обычного операторного выражения, иначе - тип справочника).

   Восьмая ячейка хранит заголовок операторного выражения.

   Девятая ячейка хранит смещение относительно начала кадра первой не определенной переменной.

   Ячейки кадра, начиная с десятой, хранят значения программных переменных x1,x2,....

4. Кадр типа 4. Этот кадр используется для организации продолжения перечисления, выполняемого базисными (реализованными программой интерпретатора) перечисляющими операторами ЛОСа; он называется кадром продолжения. Такой кадр вводится процедурой интерпретатора, обрабатывающей первое обращение к перечисляющему оператору. При откатах к данному оператору применяется другая, обеспечивающая шаг перечисления, процедура интерпретатора, которая корректирует содержимое введенного изначально кадра продолжения и устраняет его по завершении перечисления.

   Седьмая ячейка кадра продолжения хранит "начало расчистки" - аналогично кадру оператора, оно определяет расчистку кадра источника при откате к повторной обработке оператора.

   Восьмая ячейка кадра хранит значение, передаваемое при откате в регистр "уровня перечисления".

   Девятая ячейка кадра хранит дополнительный тип кадра. Обычно это 0; в случае кадра, вводимого для оператора "лимит(...)", это значение равно 1; в случае кадра для оператора "контрольприема(...)" оно равно 3.

   Ячейки кадра, начиная с десятой, хранят значения "скрытых" переменных, обеспечивающих продолжение перечисления. Для каждого конкретного оператора число и назначение их - свои собственные.

Из-за режима перечисления глобальный стэк отличается от обычного стэка: даже после того, как перечисляющий оператор выполнен, кадр его сохраняется в стэке для продолжения перечисления. Это может приводить к тому, что фактически реализуемый "текущий" стэковый кадр будет находиться не в конце, а в середине глобального стэка - после него будет размещена группа кадров, обеспечивающих продолжение начатых перечислений. Данное обстоятельство несколько усложняет работу интерпретатора: при присвоении значений новым программным переменным может не хватить места в ранее зарезервированной области текущего стэкового кадра. Тогда приходится сдвигать все последующие кадры, расширяя эту область.

Вход в программу отладчика ЛОСа организуется интерпретатором при обнаружении некорректного обращения к оператору (операторному выражению) либо при усмотрении заранее установленного условия прерывания. При этом в конце глобального стэка формируется кадр фиктивного операторного выражения "прерывание". Этот кадр не имеет источника. Его четвертая ячейка хранит указатель кадра прерывания, роль которого играет константа 0xFFFFFFFF - i, где i - вспомогательный параметр, равный 1 либо 2. После создания кадра прерывания управление передается программе логического символа "прерывание" с таким типом обращения, как если бы вычислялось значение операторного выражения. Соответственно, программа отладчика начинается с оператора "обращение(0)", устанавливающего наличие данного типа обращения.

Условия на прерывания могут устанавливаться самой программой отладчика. Обычно в этом случае нужно, чтобы условие отслеживалось лишь после выхода из отладчика и возобновления внешних вычислений. Вместе с тем, при отладке отладчика может понадобиться установка прерываний, срабатывающих в нем самом. При этом произойдет обращение к программе отладчика из нее же, и снова возникнет проблема уточнения контекста, к которому будут отнесены новые установки на прерывание. Для решения этой проблемы создан стэк установок на прерывание. При каждом входе в отладчик ЛОСа предпринимается переход к следующему кадру этого стэка; при выходе из отладчика - последний кадр стэка отбрасывается. Оператор отладчика, вводящий установку на прерывание, получает информацию о том, куда ее следует занести - в предпоследний кадр стэка установок на прерывания (для активизации после выхода из отладчика) либо в последний (для немедленной активизации в рамках программы отладчика). Эта информация, называемая ниже указателем установки прерывания, в первом случае есть символ 1, во втором - 0.

15.2 Серия операторов "трассировка"

Отладчик ЛОСа использует серию операторов "трассировка($A_1 \ldots A_n$)", реализуемых непосредственно оператором ЛОСа и предназначенных для анализа и модификации структур данных интерпретатора с уровня программы ЛОСа. В частности, с помощью этих операторов устанавливаются либо отменяются установки на прерывания. Прежде чем перейти к описанию программы отладчика, перечислим операторы данной серии.

1. "трассировка(прерывание t_1)". Значение переменной t_1 становится равно ссылке на последний кадр прерывания. Эта ссылка представляет собой символьный номер смещения в глобальном стэке начала кадра.

2. "трассировка(внешзнак t_1 t_2 t_3)". Значением t_1 является ссылка на некоторый

кадр глобального стэка. Переменной t_2 присваивается ссылка на предыдущий кадр (не на источник кадра, а именно на предшествующий ему в стэке кадр); переменной t_3 становится равна типу предыдущего кадра (число от 1 до 4). Если кадр - первый в стэке, то оператор ложен.

3. "трассировка(контекст t_1 t_2)". t_1 - ссылка на кадр K глобального стэка. В зоне задач формируется набор $t_2 = (a_1 \dots a_p)$, содержащий информацию о кадре:

   а) Если K - кадр задачи, то $a_1 = 1$; a_2 - ссылка на источник кадра либо (для исходного кадра) логический символ "нет"; a_3 - отложенный уровень либо логический символ "нет"; a_4 - максимальный уровень задачи; a_5, a_6, a_7, a_8 - адрес продолжения, представленный символьными номерами: смещения фрагмента программы в сегменте зоны программ, сегмента зоны программ, смещения ячейки продолжения в фрагменте программы, уровня перечисления. a_9 - набор значений программных переменных кадра.

   б) Если K - кадр оператора, то $a_1 = 2$; a_2 - ссылка на источник кадра; a_3, a_4, a_5, a_6 - адрес продолжения; a_7 - заголовок оператора; a_8 - набор значений программных переменных.

   в) Если K - кадр операторного выражения, то $a_1 = 3$; a_2 - ссылка на источник кадра (у кадра прерывания вместо этого указана переменная "x1"); a_3, a_4, a_5, a_6 - адрес продолжения; a_7 - заголовок операторного выражения; a_8 - индекс обращения; a_9 - набор значений переменных.

   г) Если K - кадр продолжения, то $a_1 = 4$; a_2 - ссылка на источник кадра; a_3, a_4, a_5, a_6 - адрес продолжения; a_7 - символьный номер уровня перечисления; a_8 - набор значений переменных кадра продолжения, используемых для продолжения перечисления.

4. "трассировка(программа t_1 t_2 t_3 t_4 t_5 t_6)". t_1, t_2, t_3 - символьные номера, соответственно, смещения начала некоторого фрагмента программы в сегменте зоны программ; самого этого сегмента и смещения обращения к процедуре интерпретатора в фрагменте программы. Переменной t_4 присваивается представление фрагмента программы в виде набора термов, сформированных в зоне задач; переменной t_5 - вхождение текущего терма в t_4; t_6 - вхождение оператора в терм по t_5. Если обращение к процедуре интерпретатора расположено в завершающей части фрагмента (после обращений к процедурам, соответствующим собственно операторам и операторным выражениям ЛОСа), то значения t_5, t_6 становятся равны 0.

5. "трассировка(внешнийэлемент t_1 t_2 t_3 t_4 t_5)". t_1, t_2 - символьные номера смещения начала фрагмента программы в зоне программ и содержащего фрагмент сегмента этой зоны. t_3, t_4, t_5 становятся равны символьным номерам смещения надфрагмента в зоне программ, содержащего надфрагмент сегмента зоны программ и смещения обращения к оператору перехода в надфрагменте (относительно начала сегмента). Если надфрагмента нет, то t_3 становится равно 0, t_4 - логическому символу "нет", t_5 - заголовку программы.

6. "трассировка(значение $t_1 t_2$)". t_1 - символьный номер ячейки стэка выражений. Значением t_2 становится логический символ "нет" (если ячейка не используется) либо объект из этой ячейки.

7. "трассировка(набор t_1)". Значением t_1 становится набор $a_1 \ldots a_7$ сохраненных перед входом в прерывание значений регистров интерпретатора: a_1 - регистр "ответ" (если он содержал 0, то указывается символ "нет"); a_2 - регистр "символ" (логический символ, программа которого выполнялась на момент прерывания); a_3 - регистр уровня обращения (приведенный к виду символьного номера); a_4 - символьный номер смещения текущего кадра глобального стэка на момент входа в прерывание; a_5 - символьный номер смещения вершины стэка выражений; a_6 - регистр истинности; a_7 - заголовок того оператора, при выполнении которого возникло прерывание.

8. "трассировка(задача t_1 t_2)". Установка прерывания при выдаче ответа задачи кадра t_1. t_2 - указатель установки прерывания (0 - в текущем кадре стэка установок на прерывания, 1 - в предыдущем).

9. "трассировка(решить t_1 t_2)". Установка прерывания при преобразовании задачи кадра t_1. t_2 - указатель установки прерывания.

10. "трассировка(логсимвол t_1 t_2)". Установка прерывания при обращении к программе логического символа t_1. t_2 - указатель установки прерывания.

11. "трассировка(ветвь t_1 t_2 t_3)". Установка прерывания при обращении к некорневому фрагменту программы, имеющему ссылку (t_1, t_2) в блоке программ. t_3 - указатель установки прерывания.

12. "трассировка(слово t_1 t_2 t_3 t_4)". Установка прерывания при обращении в кадре t_1 к некорневому фрагменту программы, имеющему ссылку (t_2, t_3) в блоке программ. t_4 - указатель установки прерывания.

13. "трассировка(позиция t_1 t_2 t_3)". Установка прерывания при обращении к оператору, имеющему смещение t_2 в t_1 - м сегменте зоны программ. t_3 - указатель установки прерывания.

14. "трассировка(конец t_1)". Сброс всех прерываний по указателю установки прерывания t_1.

15. "трассировка(стоп t_1)". Ввод пошаговой трассировки по указателю установки прерывания t_1.

16. "трассировка(сброс t_1 t_2)". t_1 - ссылка на кадр глобального стэка. Ввод трассировки по шагам в этом кадре; t_2 - указатель установки прерывания.

17. "трассировка(обрыв t_1 t_2)". t_1 - ссылка на кадр глобального стэка. Ввод пооператорной трассировки в этом кадре; t_2 - указатель установки прерывания.

18. "трассировка(выход t_1 t_2)". t_1 - ссылка на кадр глобального стэка. Ввод прерывания по выходу из этого кадра. t_2 - указатель установки прерывания.

19. "трассировка(ответ t_1)". Ввод прерывания при выходе на кадр задачи. t_1 - указатель установки прерывания.

20. "трассировка(число t_1 t_2)". Ввод прерывания после t_1 шагов работы интерпретатора (t_1 - число). t_2 - указатель установки прерывания.

21. "трассировка(актив)". Включение счетчика шагов работы интерпретатора.

22. "трассировка(пассив)". Выключение счетчика шагов работы интерпретатора. При обращении к отладчику ЛОСа счетчик шагов автоматически выключается, а при выходе из отладчика - включается. Это происходит без явного использования данного и предыдущего операторов; они нужны вне программы отладчика.

23. "трассировка(величина t_1)". Переменной t_1 присваивается число на счетчике шагов работы интерпретатора.

24. "трассировка(файл t_1 t_2 t_3 t_4)". (t_1, t_2) - ссылка на начало фрагмента программы в зоне программ. (t_3, t_4) становится равно ссылке на начало этого фрагмента в блоке программ.

25. "трассировка(зона t_1 t_2 t_3 t_4 t_5 t_6)". (t_1, t_2) - ссылка на начало фрагмента программы в зоне программ (t_1 - номер сегмента, t_2 - смещение в сегменте). t_3 - фрагмент программы, записанный в зоне задач, то есть набор термов $(A_1 \ldots A_p)$. t_4 - вхождение терма A_i в t_3; t_5 - вхождение оператора либо операторного выражения B в A_i. t_6 становится равно смещению B в сегменте t_1.

26. "трассировка(Текзадача t_1)". Переменной t_1 присваивается смещение последнего кадра задачи.

27. "трассировка(исходнаязадача t_1)". Если было введено прерывание на преобразования задачи некоторого кадра, то переменной t_1 присваивается ссылка на этот кадр. Иначе оператор ложен.

28. "трассировка(тип t_1)". Переменной t_1 присваивается тип обращения к программе "прерывание". Типом обращения к этой программе называется некоторое число, характеризующее причину прерывания. Рассматриваются следующие типы обращений:

    1 - переполнение зоны задач;

    2 - переполнение глобального стэка;

    3 - превышение при откате сброса максимально допустимого число перечисляющих операторов;

    4 - исчерпание переменных;

    5 - нет программы логического символа, к которой произошло обращение;

    6 - превышение максимально допустимого размера текста;

    7 - некорректное обращение к оператору;

    8 - переполнение стэка выражений;

    9 - обращение из трассировки согласно заранее установленному условию;

    10 - контроль зацикливания по размеру глобального стэка (для обнаружения циклически повторяющихся рекурсивных обращений).

29. "трассировка(0)". Внутренний перезапуск логической системы - расчистка зоны задач и основных регистров (зона программ сохраняется) с выходом в главное меню.

30. "трассировка(результат t_1)". Переменной t_1 присваивается содержимое регистра "ответ" (если этот регистр обнулен, то оператор ложен).

31. "трассировка(ключ t_1)". Переменной t_1 присваивается тип установки на прерывание (символьный номер), хранящейся в предпоследнем кадре стэка установок на прерывание (при отсутствии таковой - логический символ "нет"). Рассматриваются следующие типы установок на прерывание:

    1 - прерывание при выдаче ответа задачи заданного кадра;

    2 - прерывание при преобразовании задачи заданного кадра;

    3 - прерывание при обращении к программе заданного логического символа;

    4 - прерывание при обращении к фрагменту программы по ссылке в блоке программ;

    5 - прерывание при обращении к оператору по ссылке в зоне программ;

    6 - прерывание при пошаговой трассировке;

    7 - прерывание при пошаговой трассировке в заданном кадре;

    8 - пооператорная трассировка в заданном кадре;

    9 - прерывание при выходе из заданного кадра;

    10 - прерывание при выходе на кадр задачи;

    11 - прерывание по достижении заданного числа шагов;

    12 - прерывание при обращении в заданном кадре к оператору по ссылке в зоне программ;

    13 - прерывание при обращении в заданном кадре к фрагменту программы по ссылке в блоке программ;

    14 - прерывание при обращении в заданном кадре к оператору "замена" по заданной переменной.

    15 - прерывание при обращении к оператору "изменение" для вхождения в заданный набор;

    16 - прерывание при обращении к оператору "контрольприема";

    17 - прерывание при обращении в заданном кадре к операторам "стандследствие" либо "контрольнормализации", либо при изменении значения переменной "x1";

    18 - прерывание при обращении к оператору "прием(N)"программы заданного логического символа;

    19 - прерывание при обращении к оператору "записьтеоремы";

    20 - прерывание при переустановки трассировки после отката по оператору "лимит".

32. "трассировка(точка t_1 t_2 t_3 t_4)". Ввод прерывания при обращении в кадре t_1 к оператору, имеющему смещение t_3 в t_2 - м массиве зоны программ. t_4 - указатель установки прерывания.

33. "трассировка(замена t_1 t_2 t_3)". Ввод прерывания при выполнении в кадре t_1 оператора "замена"для переменной t_2. t_3 - указатель установки прерывания.

34. "трассировка(изменение t_1 t_2)". Ввод прерывания при изменении разряда набора t_1. t_2 - указатель установки прерывания.

35. "трассировка(контрольприема t_1 t_2 t_3 t_4 t_5)". Ввод прерывания при обращении в кадре t_1 к оператору "контрольприема(t_2 t_3 t_4)". t_5 - указатель установки прерывания. После выхода на оператор "контрольприема(...)" автоматически осуществляется переустановка на пооператорную трассировку в кадре. Если t_1 - не ссылка на кадр, а переменная, то прерывание выполняется в произвольном кадре.

36. "трассировка(контроль t_1)". Включение при $t_1 = 1$ и выключение при $t_1 = 0$ режима контроля холостого хода приемов. Предварительно создается комментарий (выписка пустоеслово) к посылкам исходной задачи. Этот комментарий накапливает статистику о холостом ходе приемов (трудоемкости неудавшихся попыток применения приема) и числе их срабатываний. Второй элемент комментария (изначально пустой набор) заполняется парами (логический символ A - набор пар (заголовок приема S - набор троек (номер приема N - счетчик его холостого хода, представляющий собой символьное число, увеличиваемое после очередной неудавшейся попытки применения приема на число тысяч шагов работы интерпретатора, занятых в этой попытке; если число шагов было меньше 1000, то увеличение не предпринимается - счетчик числа применений приема))). Заполнение этого комментария при включенном режиме контроля холостого хода происходит автоматически на уровне интерпретатора ЛОСа.

37. "трассировка(окончание)". Завершение работы системы. Этот оператор позволяет завершать работу системы из произвольной точки программы. Фактически для выхода из системы вместо него используется оператор "ответ", обращение к которому из программы символа "вход" инициирует выдачу ответа на исходную фиктивную задачу, т.е. выход из системы.

38. "трассировка(клавиатура)". Ложно, если в период после последнего ввода информации с клавиатуры (мыши) была нажата клавиша клавиатуры либо кнопка мыши.

39. "трассировка(процедура t_1 t_2)". Ввод прерывания при обращении в кадре t_1 к операторам "контрольнормализации", "контрольбуфера", "синтезатор", "подслучаи" либо при изменении значения переменной "x1". t_2 - указатель установки прерывания.

40. "трассировка(обращение t_1 t_2 t_3)". t_1 - ссылка на кадр. Переменной t_2 присваивается ссылка на источник этого кадра либо, в случае кадра прерывания, ссылка на предыдущий кадр. Переменной t_3 присваивается тип источника. Если кадр исходный, то оператор ложен.

41. "трассировка(прием t_1 t_2 t_3)". Ввод прерывания при обращении к оператору "прием(t_2)"программы логического символа t_1; здесь t_2 - набор цифр. t_3 - указатель установки прерывания. После выхода на этот оператор автоматически осуществляется переустановка на пооператорную трассировку.

42. "трассировка(внешкадр t_1 t_2)". t_1 - символьное число. Переменной t_2 присваивается ссылка на кадр, являющийся t_1-м источником (по цепочке переходов от кадра к источнику) текущего кадра глобального стэка. При $t_1 = 0$ - присваивается ссылка на текущий кадр.

43. "трассировка(смответ t_1)". Переменной t_1 присваивается содержимое регистра "смответ". Этот регистр хранит смещение кадра глобального стэка, при выходе из которого организуется обращение к программе "прерывание". Используется для отслеживания момента выхода из процедуры, внутри которой ведется трассировка.

44. "трассировка(суммавсех t_1 t_2)". t_1 - либо 0, либо набор. В первом случае определяется количество используемых 8-байтных слов в зоне задач (общее число таких слов в ней равно 256K); во втором - количество 8-байтных слов в зоне задач, достижимых по цепочкам ссылок из набора t_1. Переменной t_2 присваивается найденное количество слов в формате символьного числа.

45. "трассировка(приемы t_1)". Если $t_1 = 1$, то включаются обращения к программе символа "контрольприема" (по умолчанию они отключены); если $t_1 = 0$, то они выключаются.

46. "трассировка(фильтр t_1 t_2 t_3)". Серия процедур, предназначенных для работы с буфером фильтрации. Он состоит из списка пар (логический символ - номер узла теоремы), указывающих на теоремы приемов ГЕНОЛОГа, применение которых блокируется. Блокировка выполняется интерпретатором при попытке обращения к оператору "контрольприема" безотносительно к тому, включен ли режим выполнения ЛОС-программы этого оператора. Если $t_1 = 1$, то устанавливается режим блокировки с применением буфера фильтрации. Если $t_1 = 2$, то режим блокировки отменяется. Если $t_1 = 0$, то буфер фильтрации расчищается. Если $t_1 = 3$, то проверяется наличие режима блокировки с применением буфера фильтрации: при наличии режима оператор истинен, иначе ложен. Если $t_1 = 4$, то в буфер фильтрации заносится пара (t_2 t_3); t_2 - логический символ, t_3 - номер узла теоремы в формате десятичного числа.

47. "трассировка(теорема t_1)". Ввод прерывания при обращении к процедуре "записьтеоремы". t_1 - указатель установки прерывания.

48. "трассировка(справка t_1 t_2)". Переменной t_1 присваивается тип прерывания, установленного до входа в последнее прерывание; переменной t_2 - смещение кадра, по которому оно было установлено.

49. "трассировка(См t_1 t_2)". Аналогично предыдущему, но для текущего прерывания.

50. "трассировка(запись t_1)". На счетчике числа шагов работы интерпретатора устанавливается десятичное число t_1.

51. "трассировка(повторение t_1)". t_1 - ссылка на кадр прерывания. Происходит откат к этому кадру и повторная инициализация программы "прерывание".

52. "трассировка(вход)". Истинно до первого внутреннего перезапуска, а затем ложно.

15.3 Начало программы "прерывание"

Программе "прерывание" не передаются значения каких-либо изначально определенных программных переменных. Анализ текущей ситуации она начинает с помощью

операторов серии "трассировка". Прежде всего, переменной x1 присваивается тип обращения к программе, указывающий причину обращения. При этом переменной x2 присваивается набор значений регистров интерпретатора, сохраненных перед входом в прерывание.

Последним элементом набора x2 является заголовок того оператора, при реализации которого интерпретатор обратился к отладчику. Если этот заголовок указывает на работу с комментариями задачи (ввод либо удаление комментария), то проверяется, предусматривает ли режим трассировки отображение на экране соответствующего действия. Для этого извлекается комментарий (трасснабор A), хранящий в наборе A список логических символов - установок на режим трассировки. Эти установки вводятся либо при просмотре задачи в задачнике перед запуском ее решения, либо непосредственно в процессе решения; вход в диалоговый блок для изменения установок - через нажатие клавиши "т". Типы установок таковы: "полный" - ручной выбор входа в подпроцессы (т.е. во вспомогательные задачи и пакетные операторы); "легковидеть" - пропуск отображения обращений к проверочным операторам; "исключение" - пропуск шагов удаления посылок и условий; "сопровождение" - пропуск изменения сопровождающих утверждений; "обращение" - пропуск входа в любые подпроцессы; "примечание" - просмотр изменений комментариев задачи; "одз" - пропуск шагов регистрации условий на область допустимых значений. Если оператор, по которому выполнено прерывание, изменяет комментарии, а в наборе A нет символа "примечание", то выход из отладчика. Последний обеспечивается оператором "ответ(...)"; здесь несущественно, что именно выдается в качестве ответа.

Если выхода из отладчика не произошло, то переменной x3 присваивается ссылка на тот кадр глобального стэка, который обслуживает выполнение данной программы "прерывание". Отбрасывается, если он был, комментарий (актив ...) к посылкам исходной задачи, после чего вводится новая версия такого комментария, в которой сохраняется номер текущего активированного информационного блока. При работе с отладчиком активированный информационный блок может меняться, и данный комментарий обеспечит восстановление исходной активации перед выходом из прерывания.

Далее идет оператор "повторение", к которому происходят откаты для перерисовки исходного текста, выдаваемого при входе в отладчик. Инициализируется переменная x4, которая будет указывать номер первой свободной строки на экране. Если обращение к отладчику произошло в результате обнаружения ошибки (тип обращения от 1 до 8), то в верхней части экрана прорисовывается сообщение об ошибке; в случае типа обращения 5 - отсутствия программы логического символа - прорисовывается также этот символ. Во всех случаях далее - переход через оператор "ветвь 3".

Если тип обращения к прерыванию равен 10 (контроль зацикливания при рекурсивных обращениях), то удаляется, если он был, комментарий (повторчисло A_1 A_2) к посылкам исходной задачи. Затем находится кадр глобального стэка, непосредственно предшествующий текущему кадру прерывания, и прослеживается цепочка источников этого кадра. Для текущего просматриваемого кадра создается тройка (тип кадра - заголовок оператора либо операторного выражения либо тип задачи, соответствующих кадру - текущий набор входных данных кадра). Эти тройки заносятся в список x6 так, что первая из них оказывается в конце списка, а последняя занесенная - в его начале. Если все тройки в списке различные, то зацикливания нет, и осуществляется выход из отладчика. Иначе - создается список x14, образованный тройками найденного цикла; вводится комментарий (повторчисло x14 пустоеслово)

к посылкам исходной задачи; создается установка на прерывание при обращении к программе того логического символа x15, который должен быть очередным в найденном цикле, и выход из отладчика.

Если тип обращения отличен от 10, то переход через "ветвь 1". Здесь проверяется наличие комментария (повторчисло ...), свидетельствующего о найденном зацикливании по рекурсивным обращениям, занесение в этот комментарий некоторой дополнительной информации, и обращение к оператору "трассировка(стоп 0)", приводящему к выходу в отладчик ЛОСа с показом на экране данного фрагмента программы самого отладчика и с установкой пошаговой трассировки. Далее предпринимается ручной анализ ситуации; фактически эта часть программы - заготовка для последующего развития автоматических средств анализа причин зацикливания.

При отсутствии указаний на зацикливание - переход через "ветвь 1", где попадаем на контрольную точку "прием(4)", с которой начинается анализ цепочки кадров глобального стэка, предшествующих кадру прерывания. Прежде всего, будет определяться набор x5 ссылок на кадры глобального стэка, начиная с того кадра, информация о котором будет отображаться на экране, и кончая текущим кадром прерывания. Изначально в x5 заносится единственная ссылка x3 на текущий кадр прерывания. Инициализируется нулем заготовка x6 набора ссылок на операторы продолжения по выходу из кадров набора x5. Определяется тип x7 той установки на прерывание, которая привела ко входу в отладчик.

Далее анализируется тип установки на прерывание x7. Если x7 = "десять" (прерывание при выходе на кадр задачи), то просматривается цепочка источников кадра, предшествующего прерыванию, пока не встретится кадр задачи. Отменяются ранее имевшиеся установки на прерывания, и вводится установка на прерывания при выполнении преобразований данной задачи. Если x7 = "четырнадцать" (прерывание при обращении в заданном кадре к оператору "замена" по заданной переменной), то цепочка источников просматривается до первого кадра операторного выражения, после чего происходит переустановка прерывания на изменение значения переменной "x1" в найденном кадре (такие замены используются в нормализаторах для изменения текущего преобразуемого терма). В обоих случаях вводится комментарий (буфер пустоеслово) к посылкам исходной задачи, инициализирующий трассировку на семантическом уровне - на экране будет отображаться ход решения задачи либо применения пакетного оператора, сопровождаемый поясняющими текстами. Далее - переход через "ветвь 1".

Переменной x8 присваивается ссылка на тот кадр глобального стэка, который являлся текущим кадром перед входом в отладчик. Если x7 = "двадцать", то имеет место вход в отладчик для переустановки прерывания после отката по оператору "лимит". Без такой переустановки трассировка может оказаться неожиданно прерванной вплоть до выдачи ответа задачи. Для переустановки просматривается цепочка кадров начиная с x8 (каждый раз переходим к источнику кадра); если встречается кадр задачи, то устанавливается трассировка по преобразованиям этой задачи; если встречается кадр оператора либо операторного выражения, которые суть пакетные операторы ГЕНОЛОГа, то устанавливается трассировка по их действиям; в прочих случаях - продолжение просмотра цепочки. После переустановки прерывания - выход из отладчика.

Если x7 не равнялось "двадцать", то определяется набор x9, описывающий содержимое текущего кадра прерывания. По этому набору определяется четверка - ссылка на оператор продолжения после выхода из отладчика, и x6 заменяется на одноэлементный набор, состоящий из этой четверки. Далее идет цикл заполнения

набора x5. На очередном шаге к началу набора x5 присоединяется ссылка на кадр x8. Определяется набор x11, описывающий содержимое кадра x8, и к началу x6 добавляется извлекаемая из x11 ссылка на оператор продолжения после выхода из кадра x8. Затем проверяется, нужно ли рассматривать источник кадра x8 (тогда значение переменной x8 заменяется ссылкой на этот источник), или x8 и является тем кадром, информацию о котором нужно отображать на экране. Необходимость перехода к источникам легко понять хотя бы из рассмотрения ситуации, складывающейся при отображении очередного преобразования задачи. В этом случае изменение задачи выполняется оператором, расположенным внутри вспомогательной процедуры, к которой обращается прием. Если не обеспечить автоматический переход от кадра вспомогательной процедуры к его источнику - кадру приема, то для получения информации о программе приема придется каждый раз переходить к нему вручную.

После заполнения набора x5 - откат к надфрагменту, в котором определялось x8, и переход из него через "ветвь 1". При этом значения начиная с x8 оказываются не определенными.

Далее располагается ветвь, обеспечивающая проверку дополнительных условий на прерывание при реализации заданного оператора программы. Находится комментарий (контекст A) к посылкам исходной задачи, определяющий такие дополнительные условия. A представляет собой список располагаемых последовательно термов либо пар термов одного из следующих типов:

1. "терм(B)", C. B - программное выражение в терминах контекста рассматриваемого оператора прерывания, определяющее некоторый объект; C - терм. Условие состоит в совпадении указанного объекта с C.

2. "заголовок(xi B)". Условие состоит в том, что значением программной переменной xi является логический символ B либо набор, начинающийся с логического символа B.

3. "входит(B xi)". Условие состоит в том, что значением переменной xi является терм либо набор, содержащий логический символ либо символ переменной B.

4. "корень", B. Условие состоит в том, что текущий терм задачи должен совпадать с термом B.

5. "См(B)". B - программная переменная xi либо выражение "буква(xi)". Этот элемент определяет не дополнительное условие на прерывание, а указание на необходимость выдачи на экран при прерывании значения выражения B, если только этим значением служит терм либо логический символ.

Обнаружив элемент (контекст A) такого типа с непустым списком A, программа проверяет, что установка на прерывание была связана с обращением к заданному оператору программы (в некотором кадре или безотносительно к выбору кадра). Затем определяется набор x10, описывающий содержимое того кадра, информация о котором будет отображаться на экране, то есть кадра, в котором выполняется оператор прерывания. Из x10 извлекается набор x11 значений программных переменных кадра. Далее реализуется цикл просмотра набора A и проверки находящихся в нем условий; при этом значения упоминающихся в A программных переменных берутся из набора x11. Если какое-либо условие не выполнено, то выход из отладчика. Иначе, по окончании просмотра A, переход через оператор "ветвь 1".

Здесь инициализируется набор x8 ссылок на фрагменты программы, относящиеся к цепи того фрагмента, информация о котором должна отображаться на экране (далее называем его текущим фрагментом). Первым элементом набора x8 станет логический символ, к программе которого относится текущий фрагмент; последний элемент набора x8 (с него x8 инициализируется) - ссылка на текущий фрагмент. Под ссылкой на фрагмент в x8 понимается четверка (смещение фрагмента в его сегменте зоны программ - номер сегмента зоны программ - смещение в фрагменте оператора перехода к подфрагменту цепи либо (для текущего фрагмента) текущего оператора - уровень перечисления). Для просмотра цепи надфрагментов текущего фрагмента используется оператор "трассировка(внешнийэлемент ...)". После заполнения x8 - переход через "ветвь 1".

Проверяется наличие комментария (техпротокол P_1 P_2) к посылкам исходной задачи. Этот комментарий указывает на режим трассировки с формированием либо использованием технического протокола. Технический протокол - инструмент отладки, позволяющий выявить и локализовать различия в процессах решения задачи двумя версиями решателя. Если в первой версии решателя запустить решение задачи с помощью клавиши "П", то в 4-м информационном блоке этой версии создается технический протокол решения. Этот блок представляет собой файл R0000.lsi, размещенный в поддиректории INF. Чтобы определить отличия в шагах решения той же задачи второй версией, следует скопировать указанный файл в ее одноименную поддиректорию и запустить решение с помощью клавиши "Ctr-п".

Структура технического протокола в 4-м информационном блоке такова. Из корневого указателя-списка этого блока по метке "техпротокол" - переход к цепочке указателей - списков $A_1, \ldots, A_n$, образующих протокол (узлов протокола). Из каждого A_i переход к очередному A_{i+1} - также по метке "техпротокол". Метки узлов протокола таковы:

1. "прием" - переход к логическому терминалу, хранящему терм "контрольприема(A_1 A_2 A_3)" либо терм "прием(A_1 A_2)" - ссылку на примененный прием (в последнем случае A_1 - логический символ, A_2 - группа цифр, образующая номер контрольной точки "прием(A_2)" в программе символа A_1).

2. "преобразование" - переход к логическому терминалу, хранящему заголовок примененного оператора, выполнившего преобразование ("заменаусловия", "занесениепосылки", "удалениепосылки" и т.п.), а также один либо пару термов, участвовавших в преобразовании: в случае преобразования замены - заменяемый и заменяющий термы; в случае занесения либо удаления терма - данный терм.

3. "число" - переход к логическому терминалу, хранящему десятичную запись номера шага, на котором было выполнено преобразование.

4. "точкапривязки" - терм, при сканировании которого сработал прием.

Комментарий (техпротокол P_1 P_2) состоит из набора информационных элементов P_1, указывающего тип режима трассировки (режим протокола) и набора информационных элементов P_2, характеризующих формируемый либо используемый протокол (блок протокола). В режиме протокола используются элементы "новый" (режим создания нового протокола) и "различны" (режим сравнения решения с техническим протоколом). В блоке протокола используются элементы следующих типов:

1. (файл A) - A есть ссылка в 4-м информационном блоке на последний обработанный узел протокола (из него при формировании протокола будет делаться ссылка на следующий узел, а при сравнении с созданным протоколом - выполняться переход к следующему узлу).

2. (набор A_1 A_2) - A_1 есть номер шага сравнения с протоколом, на котором был введен данный элемент; A_2 - набор троек (содержимое логического терминала "прием" - содержимое логического терминала "преобразование" - ссылка на узел протокола) для элементов технического протокола, следующих за тем, на котором возникло рассогласование. Этот элемент применяется для восстановления сравнения с техническим протоколом, если после небольшой полосы различий снова наблюдается полное согласование действий.

3. (число A_1 A_2) - A_1 есть пара (номер предыдущего шага прерывания для текущего решения - номер предыдущего шага прерывания для технического протокола); $A2$ - набор пар (пара для начала и конца интервала наблюдения по текущему решению - величина замедления на этом интервале), в котором отбираются 5 наибольших замедлений, выявленных на шагах трассировки при сравнении текущего решения с техническим протоколом.

Возвращаясь к рассмотрению программы отладчика, заметим, что после извлечения комментария (техпротокол ...) проверяется наличие трассировки по преобразованиям задачи текущего кадра. Создание и использование технического протокола пока предусмотрено только для анализа такой цепочки преобразований. Переменной x11 присваивается заголовок текущего оператора. Сначала рассматривается случай, когда этот оператор изменяет посылки либо условия задачи. Здесь формируется список x12 термов, участвующих в преобразовании: для случая замены - пара (заменяемый терм - заменяющий терм), иначе - одноэлементный набор из добавляемого либо исключаемого терма. После перехода через "ветвь 4" с помощью набора x8 определяется ссылка x14 на примененный прием (в формате логического терминала "прием" узла протокола). Находится номер x15 текущего шага работы интерпретатора, который несколько операторов спустя корректируется на число шагов, прошедшее с момента запуска решения задачи. Определяются набор x16, описывающий содержимое текущего кадра задачи, а также набор x17 значений программных переменных этого кадра. Находится текущий логический символ x18 сканирования задачи (для начала цикла сканирования - тип задачи). Далее действия разветвляются. Если имеет место режим создания протокола, то в 4-м информационном блоке вводятся, в соответствии с описанным выше форматом, новый узел протокола и сопровождающие его терминалы "прием", "преобразование", "число", "точкапривязки", после чего - выход из отладчика. Если имеет место режим сравнения с имеющимся протоколом, то переход через "иначе 1".

Здесь определяется ссылка x23 на узел протокола, следующий за уже просмотренным на предыдущем шаге. Из логических терминалов этого узла извлекаются данные для сравнения с описывающими текущее преобразование значениями x14, x11, x12. После того, как установлена идентичность действий, предпринимается коррекция накопителя "число" из комментария x10, в котором сохраняются 5 наибольших замедлений по сравниваемым шагам (при просмотре информации о замедлениях будут выдаваться промежуток времени, в шагах интерпретатора, на котором имело место замедление, а также величина замедления). Далее - выход из отладчика.

Если же обнаружено рассогласование с протоколом, то переход через "ветвь 1". Здесь проверяется наличие в блоке протокола информационного элемента "набор". Если элемента "набор" нет, то он вводится. Если он есть и в нем обнаруживается указание на преобразование, идентичное текущему, то выполняется переустановка указателя "файл" из блока протокола для продолжения сравнения преобразований с найденной точки совпадения, элемент "набор" удаляется, и выход из отладчика. В остальных случаях - откат к точке учета комментария "техпротокол" и переход через оператор "ветвь 1", расположенный непосредственно перед оператором "исходнаязадача(x9)".

Инициализируется переменная x9, указывающая вхождение в наборе x8 ссылки на фрагмент, программа которого отображается на экране (вначале это правый край набора x8). Вводится пустой накопитель x10 объектов, для которых будут вводиться при просмотре значений программных переменных временные обозначения. Вводится пустой накопитель x11 четверок (буква, указывающая тип объекта - номер объекта в списке x10 - строка для прорисовки на экране - тип просмотра (0 - формульный, 1 - текстовый)). Инициализируются вспомогательные накопители x12,x13. Если исходная задача имела комментарий (сдвиготкатов A), указывающий, что на экране должно быть прорисовано не текущее преобразование, а его двойник из технического протокола, то этот комментарий удаляется, а в накопитель x13 заносится пара (техпротокол A). Далее - переход через "ветвь 1".

Проверяется, что вход в отладчик произошел при трассировке (x1 = 9). Если на экране должен быть отображен на верхнем, семантическом уровне очередной шаг решения задачи либо вывода теорем, то переход через оператор "иначе 2". В противном случае проверяется, не имела ли место установка на прерывание по достижении заданного числа шагов (x7 = "одиннадцать"). Если это так, то сначала проверяется наличие комментария (транскоммент A_1 A_2), вводимого при повторном запуске решения вспомогательной задачи типа A_2 либо пакетного оператора с заголовком A_2. У этого комментария A_1 задает число шагов интерпретатора перед входом в трассировку с небольшим недостатком, и по достижении этого числа шагов (т.е. в данный момент работы отладчика) предпринимается замена A_1 на переменную, с выходом из отладчика. Результатом будет повторный вход в отладчик и установка семантической трассировки, как только появится обращение к задаче типа A_2 либо пакетному оператору с заголовком A_2. Этот способ входа в задачу либо пакетный оператор при повторном запуске применяется в тех случаях, когда нужно компенсировать небольшое возможное отклонение числа шагов при семантической трассировке от числа шагов без трассировки. При отсутствии комментария (транскоммент ...) - сбрасывается установка на прерывание по достижении заданного числа шагов и тип установки x7 заменяется на 6 (пошаговая трассировка, которая здесь еще не установлена). Далее безотносительно к тому, имела ли место установка на прерывание по достижении заданного числа шагов, выполняется расчистка экрана, x12 изменяется на 0 (что определяет в дальнейшем прорисовку текущего фрагмента программы), и откат к оператору "ветвь 1", расположенному перед "равно(x1 9)".

Если выше имел место переход через оператор "иначе 2", то далее сначала рассматривается случай отображения на экране текущего преобразования задачи, текущей цепи задач и вспомогательных задач, решавшихся для текущего преобразования. Программа для этого случая располагается после оператора "или(входит(цепьзадач комментариипосылок(x14))Входит(буфер комментариипосылок(x14)))". Для понимания ее необходимо представление о программе интерфейса задачника, поэтому данную ветвь отладчика мы опишем ниже, после рассмотрения указанной програм-

мы. Оператор "ветвь 1", расположенный перед оператором "исходнаязадача(x14)", переводит в несложную процедуру отображения на экране теоремы, полученной при логическом выводе.

Возвращаемся к продолжению "главного ствола" программы отладчика, переходя через оператор "ветвь 1" перед "равно(x1 9)". Если x12 не равно 0, то под уже выведенной на экране информацией проводится горизонтальная черта. Если произошло прерывание при обращении к программе символа "пуск", то устанавливается пошаговая трассировка. Далее - обращение к циклу запросов с клавиатуры команд отладчика.

15.4 Цикл обработки команд отладчика

Начальный фрагмент обработчика команд легко найти по оглавлению программ ("Отладчик ЛОСа - Обработчик команд - Исходная точка"). Если x12 равно 0, то переменной x14 сразу присваивается команда "частичныйответ" для прорисовки на экране текущего фрагмента программы вплоть до текущего оператора; в противном случае команда x14 вводится через клавиатуру либо мышь. Реализуются следующие команды:

1. x14 = "тринадцать". Нажата клавиша Enter. Активируется тот информационный блок, который был активирован при входе в отладчик, и выход из отладчика.

2. x14 = "частичныйответ" либо x14 = "нормпроизведениевсех". Соответственно, нажата клавиша "о" либо "ф". В первом случае происходит прорисовка текущего фрагмента до текущего оператора, во втором случае - всего текущего фрагмента. Переменной x15 присваивается ссылка на текущий фрагмент, извлеченная из набора x8. Если экран не расчищен, то он расчищается. По комментарию (трассировка ...) определяется номер шага, на котором было начато решение задачи из задачника; находится номер x18 текущего шага, и определяется число x19 шагов с момента начала решения. В правом верхнем углу экрана прорисовывается текст "шаг № x19".

   Далее - переход через "ветвь 2" и прорисовка в левом верхнем углу экрана текста "Программа S", где S - заголовок программы, к которой относится прорисовываемый фрагмент. Этот заголовок расположен в начале набора x8. По ссылке на фрагмент x15 определяются значения x21 (представление фрагмента в виде набора термов), x22 (вхождение текущего терма в x21), x23 (вхождение в x22 текущего оператора). Если просматривается фрагмент из цепи надфрагментов текущего фрагмента, то под текущим термом понимается оператор перехода к очередному фрагменту цепи. При наличии установки на пооператорную трассировку происходит замена x23 на корневое вхождение текущего терма. В случае команды "частичныйответ" в наборе x21 отбрасываются все термы, расположенные после текущего терма. Далее - переход через "ветвь 1".

   Переменной x12 присваивается набор, состоящий из тройки (смещение начала фрагмента в сегменте зоны программ - номер сегмента - копия набора x21). После этого каждый оператор перехода "ветвь(A B)" либо "иначе(A B)" в наборе x21 заменяется на "ветвь(N)" либо "иначе(N)", где N - порядковый номер перехода.

Далее - переход через "ветвь 1" к циклу прорисовки фрагмента. Позиция начала прорисовки устанавливается по левому краю второй сверху 19-пиксельной полосы. Перед прорисовкой очередного терма набора x21 определяются координаты x25-x26 текущей позиции на экране. Если терм представляет собой оператор перехода, то после его прорисовки к концу набора x12 добавляется четверка (x25 - x26 - прорисованный терм - в зависимости от того, является ли прорисованный терм текущим термом фрагмента, корневое вхождение этого терма либо 0). Оператор перехода к подфрагменту текущей цепи фрагментов прорисовывается малиновым цветом. Иначе - переход через "иначе 2", где после прорисовки терма (в случае текущего терма в нем малиновым цветом выделяется текущий оператор) к концу набора x12 добавляется четверка (x25 - x26 - прорисованный терм - в зависимости от того, является ли терм текущим термом фрагмента, вхождение в него текущего оператора либо 0).

По окончании прорисовки термов набора x21 - переход через "иначе 1", определение координаты x24 - x25 текущей позиции, и еще раз переход через "ветвь 1", к определению номера x4 строки для проведения под фрагментом программы горизонтальной линии. После прорисовки этой линии проверяется, имеет ли исходная задача комментарий (контекст ...), содержащий указатель (См A) на автоматическую выдачу после прорисовки программы значения выражения A. Если имеет, причем выход в отладчик произошел по установке на прерывание при обращении к заданному оператору, то выполняется прорисовка выражения A, двоеточия, и с новой строки - прорисовка формульным редактором терма, являющегося значением A.

Заметим, что по окончании прорисовки фрагмента в x12 остается набор, начинающийся с тройки (смещение начала фрагмента в сегменте зоны программ - номер сегмента - набор термов фрагмента), после которой идут четверки (столбец - строка - терм - 0 либо выделенный оператор этого терма), указывающие места прорисовки на экране термов фрагмента. Этот набор будет использоваться последующими командами.

3. x14 = "Плюс". Нажата клавиша Esc. Происходит внутренний перезапуск системы с выходом в главное меню.

4. x14 = "попытка". Нажата клавиша Ctr-F3. Происходит выход из отладчика с предварительным обращением к оператору "тест(1)". Это обращение можно отслеживать из отладчика языка СИ для перехода к нему от отладчика ЛОСа в текущей точки трассировки.

5. x14 = "Текзадача". Нажата клавиша Ctr-F4. Происходит обращение к оператору "трассировка(стоп 0)" для просмотра отладчиком ЛОСа его собственного стэкового кадра. Используется при отладке отладчика.

6. x14 = "заменагруппы" либо x14 = "префикснаярекурсия". Соответственно, нажаты клавиши "б" либо "Ctr-б". В первом случае происходит просмотр описания примененного либо применяемого приема, во втором случае - просмотр при обнаружении рассогласования с техническим протоколом описания того приема, который должен был бы применяться согласно техническому протоколу. В первом случае программа обращается для получения ссылки x16 на текущий прием к оператору "поискприема"; во втором - считывает эту ссылку из 4-го

информационного блока по информации, хранящейся в комментарии (техпротокол ...). В обоих случаях найденная ссылка присваивается переменной x15, и далее - переход через "ветвь 2".

Если x15 есть ссылка на прием ГЕНОЛОГа (т.е. является набором длины 3), то находится ссылка x22 на логический терминал концевого пункта оглавления базы приемов, хранящий ссылку на узел приема. В списке ссылок этого терминала нужная ссылка выделяется заголовком "текприем", и далее (переход через "ветвь 3") происходит обращение к оператору "блокприемов(x22)", реализующему интерфейс просмотра описания приема. Этот оператор будет более подробно описан впоследствии; он реализует интерфейс редактора приемов ГЕНОЛОГа. По окончании просмотра приема вводится комментарий (автоклавиатура ...) к посылкам исходной задачи, обеспечивающий эмуляцию нажатия клавиши "p" (кир.) для восстановления исходного изображения кадра трассировки.

Если же x15 было ссылкой на прием, реализованный на ЛОСе, то текущий путь в оглавлении программ переустанавливается на концевой пункт, соответствующий данному приему, и происходит обращение к оглавлению программ. По выходе из этого оглавления - те же действия по восстановлению исходного кадра, что и выше.

7. x14 = "подстановка". Нажата клавиша "курсор вниз". Это нажатие переводит в режим выделения операторов текущего просматриваемого фрагмента программы многоцветной указкой. Прежде всего, удаляется комментарий (просмотртерма ...), если он был введен предыдущими процедурами. Затем рассматривается длина набора x12. Если она равна 2, т.е. прорисован единственный терм, то сразу же предпринимается обращение к оператору "просмотртерма", который перекрашивает фон терма в голубой цвет и выделяет многоцветной указкой текущий подтерм этого терма. Если при просмотре терма была нажата клавиша Enter, то оператор "просмотртерма" создаст комментарий (просмотртерма A_1 A_2), у которого A_1 - вхождение выделенного в процессе просмотра подтерма. После выхода из оператора "просмотртерма" A_2 заменяется на вхождение в x12 четверки, соответствующей прорисованному терму фрагмента программы. Этот комментарий будет далее использован для переустановки прерывания перед выходом из отладчика. Такая переустановка выполняется после отката к оператору "ветвь 2"; ветвь этого оператора (к нему будут происходить откаты и из других точек программы) называем ниже циклом установки прерывания.

   Если длина набора x12 больше 2, т.е. прорисовано больше одного оператора фрагмента программы, то переход через "иначе 3". Здесь переменной x15 присваивается вхождение в набор x12 четверки, соответствующей первому прорисованному терму фрагмента программы. Далее идет оператор "повторение", к которому будут происходить откаты при переходе к выделению очередного терма. Переменной x16 присваивается четверка по вхождению x15, переменной x17 - терм этой четверки, x18 - 0 либо вхождение выделяемого малиновым цветом подтерма (текущий выполняемый оператор либо оператор перехода к следующему фрагменту текущей цепи). Затем терм x17 выделяется - перерисовывается на старой позиции, но при голубом цвете фона. Далее - переход через "ветвь 1" к фрагменту, в котором реализуется цикл обращений к клавиатуре и мыши для получения команды x20. Эта команда определяет действия при

фиксированном выделенном голубым цветом фона терме x17 просматриваемого фрагмента программы. Рассматриваются следующие случаи:

а) x20 = "подстановка". Нажата клавиша "курсор вниз". Если оператор x17 не является оператором перехода и неоднобуквенный, то он перерисовывается с белым цветом фона, и выполняется обращение к процедуре "просмотртерма" для выделения в нем какого-либо подтерма. Если после выделения подтерма было нажато Enter, то процедура создает комментарий (просмотртерма A_1 A_2). После выхода из нее A_2 заменяется на вхождение x15 в набор x12 четверки, соответствующей оператору x17. В этом случае происходит откат к циклу установки прерывания. Если же указанный комментарий не был создан, то откат к получению новой команды x20.

б) Если x20 = "внешконъюнкция" (клавиша "курсор вправо") либо x20 = "циклвариантов" (клавиша "курсор влево"), причем слева либо, соответственно, справа от выделенного оператора x17 нет других операторов, то команда x20 блокируется, и откат к получению новой команды.

в) Если x20 = "внешсумма" (клавиша "курсор вверх") либо x20 = "внешконъюнкция" либо x20 = "циклвариантов" (клавиши "курсор вправо", "курсор влево"), то текущий оператор x17 перерисовывается на белом фоне, и продолжение анализа команды x20 после перехода через "ветвь 1".

г) Если x20 = "внешсумма", то откат к циклу установки прерывания.

д) Если x20 = "внешконъюнкция", то позиция x15 текущей четверки в наборе x12 сдигается на единицу вправо, и откат к выделению нового текущего терма.

е) Если x20 = "циклвариантов", то позиция x15 сдвигается на единицу влево, и откат к выделению нового текущего терма.

ж) Если x20 = "фигуры" либо x20 = "тринадцать" (клавиши Ctr-Enter и Enter). Если комментарий (просмотртерма A_1 A_2) уже имелся, то в нем A_2 заменяется на x15, иначе такой комментарий вводится, с A_1 равным вхождению левого края текущего оператора x17 и A_2 равным x15. Если x20 = "тринадцать", то вводится комментарий "точка", указывающий, что вводимая установка на прерывание относится только к текущему кадру глобального стэка. Затем - откат к циклу установки прерывания.

з) Если x20 = "узелраздела" (клавиша F3), то предпринимается обращение к просмотру и редактированию справочной информации по заголовку оператора x17.

и) Если x20 = "мышь" (нажата клавиша мыши), то определяются тип x21 сообщения от мыши (0 - левая кнопка, 2 - правая) и столбец-строка x22,x23 позиции курсора мыши. Если нажата левая кнопка (команда перехода к выделению подтерма, на который указывает курсор мыши), то предпринимается поиск позиции x24 в наборе x12, определяющей прорисовку того оператора, на котором расположен курсор. После перехода через "иначе 3" проверяется, что курсор действительно находился над одним из операторов (x24 не равно 0), и определяется вхождение x26 в этот оператор наименьшего подтерма, выделенного курсором. Затем формируется набор x27 кодов клавиатуры, определяющих перемещение от ранее выделенного оператора к оператору, на котором расположен курсор. Если вхождение x26 - не корневое, то к концу набора x27 добавляется код клавиши "курсор вниз" для входа в оператор "просмотртерма", а

также символ "мышь" для обработки сообщения от мыши (того же самого, что и в данной команде) после входа в "просмотртерма". Вводится комментарий (автоклавиатура x27) к посылкам исходной задачи, и откат к запросу команды x20.

Если левая кнопка курсора мыши была нажата не над оператором, то переход через "ветвь 2", где формируется комментарий (автоклавиатура ...) для нажатия клавиши "курсор вверх", выводящего из режима выделения подтермов операторов. Наконец, при нажатии правой кнопки - переход через "иначе 1", где выполняются те же действия, что и после нажатия Enter.

Переходим к рассмотрению ветви программы, определяющей цикл установки прерывания (переход через "ветвь 2" из фрагмента, начинающегося оператором "равно(x14 подстановка)"). Здесь извлекается комментарий x16 вида (просмотртерма A_1 A_2). Находится вхождение x18 в набор операторов текущего фрагмента того оператора, внутри которого был выделен подтерм. Если этот оператор не является оператором перехода, то находится смещение x19 в содержащем текущий фрагмент сегменте зоны программ обращения к реализации выделенного подтерма. Сбрасываются ранее имевшиеся установки на прерывания, и вводится установка на прерывание при обращении к процедуре интерпретатора по смещению x19. Если имелся комментарий "точка", то установка активизируется только при нахождении в текущем кадре глобального стэка, иначе - при любом выходе на данную процедуру. Если был выделен оператор перехода, то определяется ссылка x19,x20 на фрагмент в блоке программ, к которому имеется переход, и после сброса старых установок на прерывания создается установка на прерывание при обращении к этому фрагменту. Комментарий "точка" учитывается так же, как в предыдущем случае.

8. x14 = "числитель". Нажата клавиша "р". Откат к повторной прорисовке исходного кадра прерывания, выданного при входе в отладчик.

9. x14 - один из символов "нижняяоценка" (клавиша "х"), "группировка" (клавиша "н"), "легковидеть" (клавиша "в"), "извлечениеварианта" (клавиша "т"), "деление" (клавиша "а"), "дробь" (клавиша "Х"), "контрольприема" (клавиша "Н"); все клавиши - кириллица. Эти нажатия клавиш используются (в сочетании с последующим набором номера) для просмотра значений программных переменных и элементов ранее вызванных на экран объектов. В ряде случаев выдается не одно такое значение, а группа значений, расположенных одно под другим.

   Прежде всего, заметим, что начиная с данного пункта (еще до идентификации значения x14) переменной x15 присваивается набор, описывающий содержимое того кадра глобального стэка, к которому относится выданный на экран фрагмент программы.

   Определяется номер x16 нижней строки экрана. Если для выдачи запрашиваемой информации, начиная с текущей строки x4, не осталось ни одной свободной 19-пиксельной полосы, то предпринимаются действия по прокрутке вверх содержимого той части экрана, которая расположена под текстом фрагмента программы. Здесь сначала проверяется, что накопитель x11 координат выданных на экран информационных элементов, определяющих значения программных переменных либо компонент ранее выделенных объектов, непуст. Как уже

говорилось выше, в x11 заносятся четверки (буква, указывающая тип объекта - номер объекта в списке x10 выделенных при просмотре объектов - строка для прорисовки на экране - тип просмотра (0 - формульный, 1 - текстовый)). Активный прямоугольник для прорисовок устанавливается начиная с верхней строки, содержащей эти элементы. Предпринимается просмотр накопителя x11, пока не будет найден элемент, отстоящий вниз от верхнего элемента не менее чем на 19 пикселей. После того, как он найден, содержимое активного прямоугольника сдвигается вверх так, чтобы найденный элемент был прорисован в верехней части. Затем корректируется значение x4 первой свободной строки; из x11 исключаются ссылки на первые, удаленные с экрана, элементы, и корректируются номера строк оставшихся элементов. Если для расчистки места нужно удалить все элементы списка x11, то это делается отдельным фрагментом (переход через "иначе 4").

Если необходимое место удалось расчистить, то переход через "ветвь 2" (иначе команда отменяется). Здесь сначала прорисовывается буква, соответствующая нажатой клавише. Затем реализуется обращение к текстовому редактору для набора в той же строке номера программной переменной либо выделенного в накопителе x10 объекта, быть может, сопровождаемого дополнительной информацией. Введенный набор термов присваивается переменной x18. Определяется значение x21 указателя выдачи информации в формульном (x21 = 0) либо скобочном (x21 = 1) виде. Переменной x22 присваивается одноэлементный набор, состоящий из списка x10 выделенных при просмотре компонент объектов (этот список позволяет вызывать компоненты для просмотра через их номера в списке). Затем выполняется обращение к оператору "трассперечисл(...)", обеспечивающему перечисление пар x25,x26 (тип объекта - номер объекта) для просмотра, и выдающему на каждом шаге перечисления пополненную версию x24 списка x10. Очередная пара x25,x26 передается оператору "блоктрассировки" для фактической прорисовки на экране. Переменная x29 при этом указывает на то, была ли прорисовка выполнена в формульном редакторе (0) либо в теккстовом (1). Если места для нее вообще не хватило, то x29 присваивается значение 2, а x28 - число дополнительных 19-пиксельных полос, необходимых для прорисовки объекта по текущей паре x25,x26. В этом последнем случае находится позиция x31 такого элемента набора x11, что прокрутка его вверх высвобождает необходимое количество места. Если при прокрутке вверх теряется первый из выданных в цикле элементов (соответствующая ему четверка набора x11 заблаговременно присваивается переменной x23), то перед прокруткой происходит обращение к оператору "клавиатура" - создается необходимая для просмотра пауза. Цикл выдачи элементов на этом этапе можно отменить, нажав клавишу пробела.

10. x14 = "высотаполосы" либо x14 = "Стандплюс" (клавиши "K" либо "к"; кир.). Выполняется сквозной просмотр набора, определяемого программной переменной с заданным номером (первая команда) либо объектом списка x10 с заданным номером (вторая команда). Такой просмотр позволяет просматривать объекты и их элементы, используя только клавиши курсора ("курсор вверх", "курсор вниз" - смена текущего элемента; "курсор вправо" - вход в просмотр элемента; "курсор влево" - возвращение во внешний набор). Программа обработки команды прорисовывает букву, соответствующую нажатой клавише, и обращается к текстовому редактору для ввода номера переменной либо объекта. Про-

веряется, что введенный терм x18 является десятичной записью числа, определяется символьное представление x22 этого числа, и по номеру x22 в наборе x10 (случай "к") либо в конце набора x15 (случай "К") находится объект x24 для просмотра. Проверяется, что этот объект является набором, создается структура данных x25 для просмотра (в формате оператора "просмотрсписка"), и для x25 предпринимается обращение к процедуре "сквознойпросмотр".

11. x14 = "прогрблок". Нажата клавиша F5 для удаления с экрана текста фрагмента программы и перерисовки размещенных под ним информационных элементов непосредственно с верхней строки.

12. x14 = "приведение" либо x14 = "группировки". Соответственно, нажата клавиша Home либо End. Команда обеспечивает переход к просмотру предыдущего либо следующего фрагмента в цепи фрагментов, связанных между собой операторами перехода. Рассматривается вхождение x9 в набор x8 ссылки на текущий отображаемый на экране фрагмент программы. В первом случае переменной x16 присваивается вхождение левого соседа вхождения x9, во втором - вхождение правого соседа. Проверяется, что x16 не равно x9 (т.е. сдвиг в указанном направлении был возможен) и не равно левому краю набора x8 (так как в начале набора x8 находится не ссылка на фрагмент, а логический символ, к программе которого относятся просматриваемые фрагменты). Затем x9 заменяется на x16, x12 - на 0 (сброс информации о прорисовке операторов фрагмента при переходе к просмотру другого фрагмента), и откат к запросу очередной команды, где из-за x12=0 будет эмулировано нажатие клавиши "о" для прорисовки нового фрагмента.

13. x14 = "контрольунификации" либо x14 = "точкапривязки". Соответственно, нажаты клавиша PageUp либо PageDown для перехода к предыдущему или следующему кадру глобального стэка в цепи кадров, связанных последовательными обращениями к программам.

    В первом случае рассматривается начальный элемент x16 набора x5, который представляет собой ссылку на текущий просматриваемый кадр глобального стэка. Определяется набор x17, описывающий содержимое этого кадра. Далее предпринимается цикл переходов к источнику кадра (если встречается кадр прерывания, то переход делается к кадру, расположенному непосредственно перед ним), пока не встретится кадр задачи, оператора либо операторного выражения. По мере прохождения кадров ссылки на них добавляются к началу набора x5, а в набор x6 заносятся ссылки на операторы продолжения при выходе из этих кадров.

    Во втором случае ссылки на кадры, расположенные после первого элемента набора x5, просматриваются до тех пор, пока не встретится кадр задачи, оператора либо операторного выражения. После этого предшествующие элементы набора x5, а также соответствующие им элементы набора x6 отбрасываются.

    В обоих случаях далее обновляется набор x8 ссылок на фрагменты программы, относящиеся к новому кадру глобального стэка. Сначала в него заносится ссылка на последний фрагмент, которая совпадает с о ссылкой на оператор продолжения следующего кадра. Затем к началу набора x8 последовательно добавляются ссылки на предшествующие фрагменты той же цепи, прослеживаемой вплоть до корневого фрагмента программы, а далее - в начало набора

x8 (случай "равно(x18 нет)") заносится логический символ x19, к программе которого относится цепь фрагментов. Текущее вхождение x9 в набор x8 устанавливается на его правый край; сбрасывается содержимое наборов x10,x11,x12, и откат к запросу очередной команды. Так как x12 равно 0, то при этом будет выполнена прорисовка фрагмента программы, соответствующего x9.

14. x14 = "усмчисло" либо x14 = "фильтрквадратов". Нажата клавиша "с" либо "м". В комментарии (трассировка ...) предпринимается замена на 1 (первый случай) либо на 0 указателя прорисовки в текстовом либо формульном режиме. Этот указатель определяет только прорисовку термов, вводимых после смены режима, не изменяя способа прорисовки ранее выданных термов.

15. x14 = "подборнеизвестных". Нажата клавиша "з" для входа в просмотр цепи задач, если он уже не имеет места. Программа вводит комментарий "цепьзадач" и откатывается к повторной обработке входа в отладчик. Для прорисовки цепи задач будет использована ветвь программы "прерывание", описание которой отложено до раздела, посвященного интерфейсу задачника.

16. x14 = "анализатор". Нажата клавиша "п" для прорисовки соответствия между обозначениями используемых в текущей задаче переменных для текстового и формульного редакторов.

17. x14 = "соответвхожд". Нажата клавиша Delete. Расчищается экран, номер x4 первой свободной строки заменяется на 0, x11 и x12 заменяются на пустой набор, и откат к запросу очередной команды.

18. x14 = "усмцелое". Нажата клавиша "й" для ввода трассировки по преобразованиям текущей задачи. Начиная с текущего кадра прерывания (ссылка на него расположена в конце набора x5), просматриваются предшествующие ему кадры до обнаружения кадра задачи x18. После этого вводится комментарий (буфер пустоеслово) (если комментарий (буфер A) уже имелся, то в нем A заменяется на "пустоеслово"). Этот комментарий будет использоваться процедурами семантической трассировки для сохранения информации о вспомогательных задачах, решенных для выполнения текущего преобразования. Затем сбрасывается старая установка на трассировку, и устанавливается трассировка по преобразованиям задачи кадра x18.

19. x14 = "обл". Нажата клавиша пробела для ввода трассировки по преобразованиям текущих пакетного оператора либо задачи. Действия аналогичны предыдущему пункту, но просматриваются не все кадры, предшествующие текущему кадру прерывания, а лишь цепочка источников того кадра, в котором произошло прерывание. При этом берется первый кадр x18, оказывающийся кадром задачи, либо кадром нормализатора, имеющего в описании своего формата символ "контрольтитра" или "контрольнормализации", либо кадром проверочного оператора или синтезатора.

20. x14 = "минимум". Нажата клавиша "л" для установки прерывания по достижении программы заданного логического символа. Название символа x17 вводится текстовым редактором; удаляется, если он имелся, комментарий (буфер ...), и трассировка переустанавливается указанным образом.

21. x14 = "префиксныйфильтр". Нажата клавиша "0". Удаляется комментарий (буфер ...) и отменяется установка на прерывание.

22. x14 = "унисборка". Нажата клавиша "1". Удаляется комментарий (буфер ...), отменяется старая установка на прерывание, и вводится режим пошаговой трассировки.

23. x14 = "если". Нажата клавиша "2". Удаляется комментарий (буфер ...), отменяется старая установка на прерывание, и вводится режим пооператорной трассировки в том кадре глобального стэка, к которому относится просматриваемый фрагмент программы (ссылка на этот кадр - начало набора x5).

24. x14 = "то". Нажата клавиша "3". Аналогично предыдущему, но в соответствующем кадре глобального стэка устанавливается не пооператорная, а более мелкая пошаговая трассировка.

25. x14 = "комплексныечисла". Нажата клавиша "4". Установка прерывания при выходе из кадра, к которому относится просматриваемый фрагмент программы.

26. x14 = "усмотрениефильтра". Нажата клавиша "5". Установка прерывания при выходе на кадр задачи.

27. x14 = "схемаидентификации". Нажата клавиша "ш" для ввода прерывания по достижении заданного числа шагов работы интерпретатора (число шагов отсчитывается с момента запуска решения задачи). После прорисовки буква "ш" происходит обращение к текстовому редактору для ввода номера шага. Этот номер (терм) преобразуется в формат x19 десятичного числа. Определяется текущий номер шага x20; с помощью комментария (трассировка ...) находится номер шага, с которого было начато решение задачи. Находится результат x23 увеличения последнего на x19. Если x20 меньше x23, то удаляется комментарий (буфер ...) и устанавливается прерывание по достижении шага x23.

28. x14 = "новыесвязки" либо x14 = "верхняягрань" либо x14 = "объединениефрагментов" либо x14 = "транслоперандов". Соответственно, нажаты клавиша "и" либо "И" либо "7" либо "6". Происходит установка прерываний. В первых двух случаях устанавливается прерывание при изменении разряда заданного набора; в первом случае набор берется из списка выделенных объектов, во втором - как значение некоторой программной переменной. В третьем случае устанавливается прерывание при изменении в текущем кадре значения заданной программной переменной; в четвертом - прерывание при достижении фрагмента программы заданного логического символа, имеющего заданный оператор.

    После нажатия клавиши происходит прорисовка, соответственно, буквы "и" либо "И" либо "з" либо "о". Предпринимается обращение к текстовому редактору для ввода сопровождающих команду параметров (термов). x19 - результирующий набор. Дальнейшие действия зависят от типа команды:

    а) Нажата клавиша "6". Проверяется, что x19 имеет длину 2. Первый элемент набора x19 должен быть однобуквенным термом, заголовок которого - логический символ x22, в программе которого нужно найти оператор, являющийся вторым элементом набора x19. Для поиска используется оператор "прогфайл

(имя x22 конец(19)x23)". Результат поиска - набор x23 пар логических символов, представляющих собой ссылки на фрагменты пути от корня программы символа x22 к найденному фрагменту. Эти ссылки расположены справа налево, так что ссылкой на найденный фрагмент служит начало набора x23. Вводится установка на прерывание при достижении данного фрагмента программы.

б) В остальных случаях проверяется, что набор x19 имеет длину 1 и находится его начало x21. Если была нажата клавиша "7", то заголовком x21 служит символьный номер некоторой переменной x23. Вводится установка на прерывание при изменении значения этой переменной (только оператором "замена") в текущем кадре.

в) Если были нажаты клавиши "и" либо "И", то первый терм набора x19 указывает номер объекта списка x10 выделенных объектов либо номер программной переменной. Соответственно, находится объект x26 с данным номером либо значение указанной программной переменной. Проверяется, что этот объект является набором. Далее вводится установка на прерывание при изменении разряда данного набора.

Заметим, что во всех перечисленных случаях после установки прерывания происходит расчистка строки, на которой отображалась введенная команда.

29. x14 = "новыйузел". Нажата клавиша Ctr-з для обрыва серийного решения задач, логического вывода либо тестирования. Здесь рассматриваются следующие подслучаи:

а) Исходная задача имеет комментарий "задачи". В этом случае происходит цикл решения задач из задачника. Предпринимается обращение к оператору "Текзадача(x18 x19 x20)", определяющему набор x18 ссылок на указатели оглавления задачника, лежащие на пути к текущей выбранной задаче, набор x19 меток перехода из этих указателей и ссылка x20 на корневой указатель того раздела R, в котором реализуется серийное решение. В логическом терминале "метка" корневого указателя второго информационного блока хранится установка "задачи(N)" на серийный вывод, $N \neq 0$, и прежде всего эта установка сбрасывается - в терминал помещается терм "задачи(0)". Без такой замены режим серийного решения продолжался бы даже после выхода из логической системы и повторного ее запуска. Далее удаляется комментарий "задачи" к посылкам исходной задачи; если ранее был включен режим контроля холостого хода приемов, то оператор "трассировка(контроль 0)" отменяет этот режим. Наконец, логические терминалы "позиция" оглавления задачника, лежащие на пути от корня раздела R до текущей выбранной задачи, нормализуются - вместо термов "число(N 0)", обозначавших область серийного решения, в них заносятся термы "число(N N)".

б) Если имеется логический терминал, достижимый из корня 6-го информационного блока по меткам "метка","повторение", то он удаляется. Наличие этого терминала поддерживает продолжение серийного логического вывода в базе теорем. Удаляется также комментарий "вывод" к посылкам исходной задачи.

в) Исходная задача имеет комментарий к посылкам "подраздел", указывающий на наличие серийного тестирования генератора приемов в подразделе базы приемов. Активируется 8-й информационный блок, и предпринимается обращение к оператору "Текприем(x18 x19 x20)". Он определяет значения x18,x19,x20,

аналогично значениям из пункта а). В логический терминал, достижимый из корневого указателя 8-го информационного блока по меткам "прием","повторение", заносится 0, и таким образом цикл тестирования останавливается. Удаляется комментарий "подраздел". Наконец, как и в пункте а), нормализуется содержимое логических терминалов "позиция" пути в оглавлении базы приемов.

г) Исходная задача имеет комментарий "команды", указывающий на цикл тестирования синтеза установок на генерацию приемов по заданному разделу базы теорем. Аналогично пункту в), но на 0 заменяется содержимое логического терминала, достижимого из корня 6-го информационного блока по меткам "прием","повторение".

30. x14 = "стандменьше". Нажата клавиша F1. Происходит обращение к справочнику по системе.

31. x14 = "внешнийквантор". Нажата клавиша "г" для входа в оглавление базы приемов. В x17 сохраняется номер текущего активного информационного блока, активируется 8-й информационный блок и находится одноэлементый набор x19, состоящий из ссылки на корень оглавления базы приемов. Далее - оператор "повторение", к которому будут происходить откаты при возвращении в оглавление базы приемов из просмотра приемов концевого пункта этого оглавления. Комментарий (вид A), указывающий тип A текущего оглавления, переустанавливается на оглавление базы приемов. Вводится комментарий "блоктрассировки", указывающий на обращение к базе приемов из трассировки. Сбрасывается, если он был, комментарий (прием ...), указывающий на ранее выбранный концевой пункт оглавления базы приемов и прием этого пункта. Затем предпринимается обращение к оглавлению базы приемов. Если выход из него происходит по истинностному значению "ложь", то восстанавливаются исходные параметры, и выполнение команды обрывается. Иначе - вводится комментарий "точкапривязки", блокирующий операции изменения базы приемов, и предпринимается обращение к редактору приемов ГЕНОЛОГа (оператор "блокприемов(x20)"). После этого обращения удаляется комментарий "точкапривязки" и проверяется наличие комментария "продолжение", инициирующего выход из прерывания и продолжение вычислений. Такой комментарий будет создан, если при просмотре базы приемов будет введена установка на прерывание при попытке применения заданного приема (нажатием Enter, F9 либо Ctr-Enter). Если этого комментария нет, то проверяется наличие комментария "блоктрассировки". Если он был удален при просмотре приема, то обрыв просмотра базы приемов, иначе - откат к повторному входу в ее оглавление. Если же комментарий "продолжение" есть, то он удаляется; вместе с ним удаляется также комментарий "блоктрассировки". Сбрасывается старая установка на прерывание, находится комментарий (начало ...) либо (левыйкрай ...), в котором сохраняется ссылка на прием прерывания, выбранный внутри оператора "блокприемов", и предпринимается установка на прерывание при попытке применения этого приема. Под попыткой применения приема понимается обращение к оператору "контрольприема(...)", расположенного в начальной части программы этого приема и идентифицирующему данный прием. В первом случае (F9 при просмотре приема) прерывание происходит только при попытке применения приема в текущем кадре, во втором (Ctr-Enter) - безотносительно

к кадру. Если при просмотре приема была нажата клавиша Enter (прерывание при фактическом применении приема), то введен комментарий (трассперечисл ...), однако он не инициирует какой-либо установки на прерывание в отладчике, а сохраняется после выхода из него. Операторы, выполняющие преобразования приемов ГЕНОЛОГа (их число невелико), контролируют совпадение вызвавшего их приема с приемом, указанным в комментарии (трассперечисл ...), и при наличии такового инициируют вход в отладчик с помощью оператора "трассировка(стоп 0)".

Во всех трех случаях далее происходит выход из отладчика.

32. x14 = "внешвывод". Нажата клавиша "щ". Эта клавиша при трассировке используется для сохранения текущего значения счетчика шагов работы интерпретатора, чтобы при повторном запуске задачи или процедуры логического вывода с помощью клавиши "а" произошел выход в отладчик на том же самом шагу. Прежде всего, проверяется наличие комментария (следствия ...), указывающего, что происходит процесс логического вывода в базе теорем. В этом случае значение счетчика шагов сохраняется в структурах данных базы теорем, которые будут описаны в последующих частях книги. Если такого комментария нет, то переход через "ветвь 2". Здесь проверяется наличие комментария (просмотрзадачи A) к текущей задаче либо какой-либо ее надзадаче. Если такой комментарий есть, то A - пара (логический символ - номер узла), определяющая координаты в задачнике той задачи, решение которой реализуется. Находится ссылка x21 на узел этой задачи; вычисляется разность x24 текущего номера шага и номера шага, на котором было начато решение, и десятичная запись числа x24 сохраняется в логическом терминале "повторение" узла x21.

33. x14 = "попыткапараметризации" либо x14 = "узелраздела". Нажата клавиша F2 либо F3. Во втором случае определяется заголовок x20 текущего выполняемого оператора либо вычисляемого операторного выражения, и предпринимается обращение к справочной информации этого заголовка. В первом случае - обращение к справочной информации о логическом символе, вводимом в специальном окне диалога.

34. x14 = "мышь". Если нажата левая клавиша мыши, то вводится комментарий (автоклавиатура ...), эмулирующий последовательное нажатие клавиш "курсор вниз" (для перехода в режим выделения операторов многоцветной указкой) и "мышь" (для создания прерывания перед выполнением того оператора, на который показывает курсор мыши, и выхода из отладчика).

35. x14 = "числзначение" либо x14 = "подобл". Нажаты клавиша "Ч" либо "ч" для прорисовки текущего чертежа. В первом случае прорисовка выполняется в общем потоке информационных элементов, вызываемых на экран; во втором - экран расчищается, фрагмент программы не прорисовывается, и сразу выдается чертеж.

36. x14 = "сдвигпеременных". Нажата клавиша Ctr-г для входа в оглавление программ. Комментарий (вид ...), указывающий тип просматриваемого оглавления, переустанавливается на оглавление программ; вводится комментарий "блок-трассировки", указывающий, что обращение к этому оглавлению предпринимается из трассировки, и происходит вход в данное оглавление. Если был выбран

некоторый концевой пункт оглавления программ, то определяются логический символ x23 и номер x24 контрольной точки "прием(x24)" в программе символа x23, после чего устанавливается прерывание при выходе на данную контрольную точку, и выход из отладчика.

37. x14 = "внутрвывод". Нажата клавиша "y" (кир.) для установки дополнительных ограничений на прерывание по достижении заданного оператора. Переменная x17 инициализируется пустым словом; если имеется комментарий (контекст ...), перечисляющий ранее установленные дополнительные ограничения на прерывание, то его содержимое передается списку x17. Происходит обращение к оператору "просмотрсписка" для коррекции набора x17; новая версия этого набора создается в информационном элементе (набор ...) структуры данных x19 оператора "просмотрсписка". На основе этой версии корректируется, вводится либо удаляется комментарий (контекст ...), и далее - выход из отладчика.

38. x14 = "конъюнктоперанд". Нажата клавиша Ctr-y (кир.) для сброса комментария (контекст ...).

39. x14 = "оператор". Нажата клавиша Ctr-p (кир.) для просмотра альтернативного шага трассировки при сравнении текущего решения с решением, находящимся в техническом протоколе. По комментарию (техпротокол ...) к посылкам исходной задачи находится ссылка x21 на узел протокола, соответствующий текущему шагу решения. По содержимому x23 логического терминала данного узла вводится комментарий (сдвиготкатов x23) к посылкам исходной задачи, используемый процедурой просмотра задач при трассировке (см. следующий раздел) для выдачи информации об альтернативном шаге.

Глава 16

Интерпретатор ЛОСа

При развитии логической системы может возникнуть необходимость ввести в язык ЛОС различные дополнительные макрооператоры, обеспечивающие ускоренную работу со специальными объектами. Это, а также возможное развитие отладчика ЛОСа, потребует внесения тех или иных коррекций в программу интерпретатора ЛОСа, а следовательно, потребует информации об организации этой программы. В данной главе и содержится такая (в общем, достаточно краткая) информация: дается представление об архитектуре и основных функциях интерпретатора ЛОСа.

Программа интерпретатора хранится в файлах LOGSYST.C, POLINOM.C, RISUNOK.C, ARIFM96.C; при ее написании использованы только возможности языка C, без выхода в C++. Конечно, использование для реализации ЛОС-программ компилятора вместо интерпретатора могло бы ускорить работу системы, однако в этом случае усложнилось бы легко доступное для интерпретатора изменение программой самой себя в процессе своей работы. С другой стороны, быстродействие современных процессоров обеспечивает достаточно высокую скорость работы логической системы, и проблема ускорения этой работы не представляется сейчас, на этапе предварительных исследований в области моделирования логических процессов, слишком актуальной. В перспективе такое ускорение могло бы быть обеспечено, например, за счет создания ЛОС - процессора.

Схемная реализация операторов ЛОСа, по сравнению с программной их реализацией в интерпретаторе, могла бы уже ускорить работу системы на порядок, даже без какого-либо распараллеливания сканирования. Так как главную часть времени работы системы составляет сейчас поиск очередного приема, то представляется перспективной следующая простая схема распараллеливания сканирования: несколько независимых процессоров, имеющих каждый в своей памяти некоторое подмножество приемов, осуществляет сканирование своей копии задачи и поиск приема. Результаты поиска пропускаются через древовидный коллектор, в котором отбирается для реализации единственный прием - по наименьшему уровню срабатывания. После применения отобранного приема в центральном процессоре, предпринимается рассылка копий измененной ситуации процессорам, продолжающим поиск приема для следующего шага.

16.1 Основные структуры данных интерпретатора

16.1.1 Хранение основных данных в оперативной памяти

Чтобы хранить в оперативной памяти системы наборы и термы, введен специальный массив, называемый зоной задач. Он имеет размер 2 Мбайт и разбит на 64-битные (двойные) слова - всего 262144 таких слов. Каждый терм либо набор представлен некоторым отрезком в зоне задач (эти отрезки называются текстами зоны задач). Первое 64-битное слово текста, называемое префиксом текста, имеет вид $N\ A$, где N - 32-битная запись длины текста, измеренной в 32-битных (обычных) словах. A - 32-битная константа, старший бит которой равен 1 (этот бит служит индикатором при поиске начала текста), а младший бит - является указателем такта закрепления текста и используется для расчистки мусора в зоне задач. Прочие биты константы A равны 0 (оператор "трассировка(суммавсех ...)"может использовать бит константы A, следующий за старшим битом). Суффикс текста имеет вид $C\ D$, где старший бит 32-битного слова C равен 1 (он служит индикатором при поиске конца текста); 32-битное слово D используется при уплотнении зоны задач как вспомогательная информационная емкость.

Набор $A_1 \dots A_n$ представляется в виде текста, образованного (кроме префикса и суффикса) последовательно расположенными двойными словами $B_1, \dots, B_n$, кодирующими элементы $A_1, \dots, A_n$. Фактически используются лишь первые (младшие) 32 бита каждого такого слова; вторые 32 бита тождественно нулевые (они используются при задании термов). Применяется следующее 32-битное кодирование рассматриваемых в ЛОСе типов данных:

а) Логические символы. Набор, кодирующий логический символ с номером N, имеет вид "x0008$mnpq$", где m, n, p, q - шестнадцатиричные цифры записи числа N.

б) Переменные. Набор, кодирующий переменную с номером N, имеет вид "x0010m npq", где m, n, p, q - шестнадцатиричные цифры записи числа N. Хотя число бит в данной кодировке позволяет работать с 65535 символами переменных, ряд операторов ЛОСа запрограммирован таким образом, что поддерживает лишь 512 первых переменных. Это связано с применением масок для задания множеств переменных - чтобы число ячеек, определяющих такие множества, было не слишком большим. Впрочем, на практике 512 переменных до сих пор было вполне достаточно - редко приходится использовать в одном процессе решения задачи более 100 переменных.

в) Наборы и термы кодируются 32-битным смещением в зоне задач (как в массиве 32-битных слов) своего префикса. Фактически в этом 32-битном смещении первые 14 бит всегда равны 0. Префикс легко отличается от внутреннего двойного слова текста по индикатору начала текста (внутри текста его бит всегда равен 0).

г) Вхождению в набор либо терм соответствует некоторое двойное слово внутри текста, представляющего этот набор либо терм. Оно кодируется 32-битным смещением в зоне задач первого (младшего) 32-битного подслова этого двойного слова.

Терм представляется в виде последовательности кодов своих логических символов и переменных (как и для набора, такой код занимает первые 32 бита двойного слова внутри текста). Скобки при этом пропускаются; необходимая для их восстановления информация сохраняется во вторых 32 битах двойных слов. Если двойное слово представляет заголовок подтерма A всего терма (в частности, A может состоять из одного символа - логического либо переменной), то вторые его 32 бита имеют вид $(N\ M)$. Здесь M - младшие 16 бит - равно числу 32-битных слов, используемых в записи подтерма A (фактически это константа сдвига для выхода на начало следую-

щего за подтермом двойного слова). N равно смещению влево для выхода на начало того подтерма $B = f(\ldots A \ldots)$, непосредственным операндом которого является A. Если же A было корнем всего терма, то $N = 3$. Однобуквенные наборы, составленные из логического символа либо символа переменной, автоматически преобразуются к формату терма.

Для вычислений с сопроцессором, а также для вычислений с применением обычной 32-битной машинной арифметики используется специальное представление чисел в зоне задач. Форматов таких чисел два - "с плавающей запятой" (64-битное представление) и "целое со знаком" (32 бита). Для каждого из них форматы префикса и суффикса - те же, что и для обычных наборов.

Число "с плавающей запятой" представляется в зоне задач набором N длины 2. Элементы этого набора кодируются так, чтобы их можно было отличить от элементов "обычных" наборов. 64 бита стандартного представления числа "с плавающей запятой" разбиваются на два идущих подряд 32-битных слова A_1, A_2. Каждый из двух элементов набора N представляет собой пару 32-битных слов. Первое из этих слов имеет шестнадцатиричный код (40080001); второе слово для первого набора равно A_1, а для второго - A_2.

Число "целое со знаком" представляется в зоне задач набором длины 1. Первые 32 бит единственного элемента этого набора имеют шестнадцатиричный код (60080001); далее следуют 32 бита, образующие собственно запись целого числа со знаком в стандартном машинном формате.

Операторы ЛОСа предусматривают ввод в процессе вычислений новых массивов чисел в одном из указанных выше форматов. Ссылка на такой массив для дальнейшего манипулирования с ним хранится в виде набора длины 2 из зоны задач. Первые 32 бита каждого из двух элементов данного набора указывают тип представления чисел: шестнадцатиричный код (50080001) соответствует формату "с плавающей запятой"; (70080001) - формату "целое со знаком". Вторые 32 бит первого элемента хранят указатель на начало массива (в формате языка C), а вторые 32 бит второго элемента - handle массива. Заметим, что длина массива в такой ссылке не указывается и (если она нужна) должна быть сохранена независимо при создании массива. Удаление массива осуществляется автоматически: если при расчистке зоны задач удаляется ссылка на массив, то удаляется и сам массив.

16.1.2 Хранение программ

Прежде чем перейти к описанию структур данных, используемых для хранения программы на ЛОСе, остановимся на формате значений программных переменных. Такими значениями в ЛОСе служат логические символы, символы переменных, наборы, термы, а также вхождения в наборы и термы. Они кодируются 32-битными наборами в точности так же, как это делалось в текстах зоны задач.

Для эффективной работы решателя, вообще говоря, нет необходимости хранить в оперативной памяти всю (достаточно большую и постоянно увеличивающуюся по мере обучения) программу его базы приемов. Достаточно использовать буфер, в котором сохранялись бы наиболее часто активизируемые при сканировании фрагменты программ. Такой буфер называется зоной программ; он состоит из сегментов - массивов оперативной памяти по 256 Кбайт. В настоящей версии интерпретатора число сегментов зоны программ равно 10; оно легко может быть изменено.

Все ЛОС-программы системы хранятся в блоке программ, образованном группой файлов с названиями Lijkl.los; (ijkl) - восьмеричный номер файла в группе. Уже при

хранении в файле фрагмент программы преобразован специальным конвертором к виду, удобному для быстрого выполнения (напомним, что при работе с редактором программ ЛОСа этот фрагмент имел вид последовательности термов - операторов ЛОСа). В файле либо в зоне программ он представляет собой последовательность групп $(A_1 \ldots A_n\ B)$, где $A_1, \ldots, A_n$ - программные переменные либо логические символы, определяющие значения, загружаемые в стэк выражений интерпретатора; B - номер процедуры интерпретатора, применяемой после указанной загрузки стэка. После группы может быть размещена не загружаемая в стэк выражений, но используемая процедурой B информация. Передача управления к следующей группе по завершении очередной процедуры в необходимых случаях корректируется этой процедурой таким образом, чтобы информационные вставки были пропущены.

Кодируются элементы программы в файле и в зоне программ по-разному: в зоне программ кодирование избыточное, ориентированное на быстрейшее выполнение процедур интерпретатора; в файле - более экономное, хотя процедура сжатия здесь тривиальная и состоит, по существу, в отбрасывании лишних нулей. Близость двух способов кодирования позволяет быстро преобразовывать один формат в другой при считывании программы из файла в зону программ.

Хранение программы в зоне программ

Фрагменты программ размещаются в каждом сегменте зоны программ подряд, без переноса окончания фрагмента в следующий сегмент. Перед каждый фрагментом размещается обнуленное 32-битное слово. Запись фрагмента в зоне программ состоит из префикса, текста фрагмента и суффикса.

Префикс состоит из четырех 32-битных слов A_1, A_2, A_3 и A_4. A_1 есть смещение от начала префикса до начала суффикса (все смещения берутся в предположении, что сегмент зоны программ есть массив 32-битных слов). A_2 есть смещение от начала префикса до первой ячейки, расположенной после суффикса. A_3 есть адрес данного фрагмента в блоке программ. Первый (старший) бит этого адреса равен 1; следующие (в порядке убывания номеров) 12 бит хранят 4 восьмеричных константы ijkl - номер файла Lijkl.los, в котором хранится фрагмент. Последние 19 бит хранят смещение начала фрагмента в указанном файле. A_4 в случае корневого фрагмента представляет собой набор, старшие 16 бит которого равны 0, а младшие 16 равны номеру логического символа, корнем программы которого служит фрагмент. Если же фрагмент не корневой, то старшие 16 бит хранят число $N+1$, где N - номер сегмента зоны программ, хранящего надфрагмент (от него имеется переход к данному фрагменту по "ветвь ..." либо "иначе ..."). Младшие 16 бит при этом хранят смещение начала префикса надфрагмента в указанном сегменте. Нумерация сегментов зоны программ начинается с 0.

Текст фрагмента состоит из 32-битных слов одного из трех типов:

а) Слово, хранящее номер N пересылаемой в стэк выражений программной переменной.

б) Слово, старшие 16 бит которого имеют вид (0 0 ... 0 1 0 0 0), а младшие 16 - номер логического символа, пересылаемого в стэк выражений.

в) Слово, старшие 16 бит которого имеют вид (0 0 ... 0 0 1), а младшие 16 - номер i применяемой процедуры интерпретатора Fi. Заметим, что все адресуемые из фрагмента программы ЛОСа функции программы интерпретатора обозначены буквой F с идущим после нее номером.

Специальные информационные слова, группы которых размещаются после но-

меров процедур интерпретатора, описываются в разделе, посвященном конвертору интерпретатора (как правило, они используются для указания смещений при условных переходах внутри фрагмента).

Суффикс фрагмента представляет собой набор 32-битных слов, указывающих адреса переходов из фрагмента по операторам "ветвь ...", "иначе ...". Они расположены в суффиксе в том же порядке, что и соответствующие операторы перехода в фрагменте. Если подфрагмент, к которому нужно перейти, считан в зону программ, то адрес перехода к нему имеет в старших 16 битах число $N + 1$, где N - номер сегмента, содержащего подфрагмент, а в младших 16 битах - смещение префикса подфрагмента в этом сегменте. Если же подфрагмент не считан в зону программ, то указывается адрес его в блоке программ - в том же формате, что и элемент A_3 из префикса. В принципе, блок программ может хранить фрагменты с временно неопределенными переходами (они доопределяются в процессе редактирования ЛОС - программы). Если такой фрагмент будет считан в зону программ, то у него старшие 13 бит будут равны 1, а младшие 16 бит - номер логического символа, выбранного в качестве временной метки перехода.

Хранение программы в блоке программ

Блок программ предназначен для хранения ЛОС-программ. В него входят файл K.los, называемый каталогом программ, а также файлы вида Lijkl.los, где (ijkl) - четырехразрядный восьмеричный номер файла. Файл K.los имеет ровно 256 Кбайт; каждый из файлов Lijkl.los - длину не более 512 Кбайт.

Каталог программ определяет адреса корневых фрагментов программ логических символов в файлах Lijkl.los. Он представляет собой последовательность 32-битных слов; i-е слово соответствует логическому символу с номером i (первое слово имеет в файле K.los смещение 0). Если программа i-го логического символа отсутствует, то соответствующее слово состоит из нулей. Иначе старший бит этого слова равен 1; следующие за ним 12 бит определяют номер (ijkl) файла Lijkl.los, хранящего корневой фрагмент программы i-го логического символа, а последние 19 бит определяют смещение начала записи данного фрагмента в файле Lijkl.los.

Файлы Lijkl.los используются для хранения фрагментов ЛОС-программ. Фрагменты программы одного и того же логического символа хранятся в одном и том же файле. Если размер файла Lijkl.los приближается в процессе развития системы к 512 Кбайт, причем к нему относятся программы более чем одного логического символа, то он автоматически подразбивается на два файла, причем нумерация следующих за ним файлов сдвигается на единицу (при этом предпринимается необходимая коррекция каталога программ и ссылок из зоны программ).

Запись фрагмента программы в файле блока программ состоит из префикса, текста фрагмента и суффикса.

Префикс имеет ровно 7 байт; он разбивается на слова A_1, A_2, A_3, первые два из которых имеют по 16 бит, а последнее - 24 бит. Старший бит слова A_1 - индикатор "мусора"; если он равен 1, то это означает, что запись уже не используется. Неиспользуемые записи устраняются при уплотнении блока программ. Следующий бит слова A_1 - индикатор корневого фрагмента: если он равен 1, то фрагмент корневой. Младшие 14 бит слова A_1 равны смещению (здесь все смещения берутся в байтах) от начала префикса до начала суффикса. Слово A_2 - смещение от начала префикса до первого байта после фрагмента (старшие 2 бита слова A_2 равны 0). Если фрагмент корневой, то слово A_3 есть номер логического символа, соответствующего программе

(тогда старшие 8 бит равны 0). Иначе младшие 19 разрядов слова A_3 - смещение в том же файле начала надфрагмента данного фрагмента (от него имеется ссылка по "ветвь ...", "иначе ..." к данному фрагменту); старшие биты этого слова равны 0.

Текст фрагмента образован последовательностью 11-ти и 17-ти -битных слов одного из трех типов:

а) Слово, хранящее номер N программной переменной, загружаемой в стэк выражений - набор длины 11. Старшие 9 бит хранят номер N, а младшие 2 бита равны (01).

б) Слово, хранящее номер логического символа, загружаемого в стэк выражений - набор длины 17; старшие 16 бит этого набора равны данному номеру, а младший равен 0.

в) Слово, хранящее номер процедуры интерпретатора - набор длины 11; старшие 9 бит этого набора равны данному номеру, а младшие равны (11).

Если текст программы завершился раньше конца байта, то до этого конца идут нули, а затем начинается суффикс.

Суффикс фрагмента программы образован последовательностью 24-битных слов, хранящих ссылки на переходы из фрагмента. Каждая ссылка в младших 19 разрядах хранит смещение префикса очередного подфрагмента данного фрагмента. Эти смещения располагаются в том же порядке, что и соответствующие им операторы перехода. Старшие 5 бит каждого слова обнулены. Возможен особый случай - если переход из фрагмента временно не определен (в процессе редактирования программы). Такой переход имеет метку - некоторый логический символ; младшие 16 бит 24-битной ссылки равны номеру этого символа, а старший бит равен 1.

Представление фрагмента программы в зоне задач

Для редактирования фрагментов программы ЛОСа предусмотрены считывание их из блока программ в зону задач и запись из зоны задач в блок программ. Фрагмент программы в зоне задач представляется при этом как набор термов - операторов этого фрагмента. Операторы перехода здесь хранятся в виде "ветвь(A_1 A_2)" либо "иначе(A_1 A_2)", где A_1, A_2 - пара логических символов, кодирующая ссылку на подфрагмент в блоке программ (нумерация ссылок на подфрагменты, используемая при отображении программы на экране редактором программ ЛОСа - виртуальная, она вводится толкьо при прорисовке на экране).

Чтобы из пары логических символов A_1, A_2 получить ссылку на фрагмент программы в блоке программ (в том же формате, в каком представлена ссылка из хранящегося в зоне программ фрагмента на его представление в блоке программ), берутся числа N_1 и N_2, образованные младшими 16 битами 32-битных слов A_1, A_2, определяются 16-битное представление B_1 числа $N_1 - 1$, а также 15-битное представление B_2 числа $N_2 - 1$ (отбрасывается старший бит числа $N_2 - 1$, который всегда равен 0). Затем формируется 32-битный набор $(1, B_1, B_2)$ - он и дает искомую ссылку. В этой ссылке первые 12 бит после старшего бита - четыре восьмеричных константы номера файла Lijkl.los; младшие 19 бит - смещение в этом файле начала фрагмента.

Некоторые операторы ЛОСа (см. серию операторов "прогфрайл(...)") работают со ссылками на вхождения операторов "ветвь(...)"; "иначе(...)" в фрагменты, хранящиеся в блоке программ. Такая ссылка - хранящийся в зоне задач набор длины 2 (пара), первым элементом которой служит пара логических символов (A_1, A_2) - ссылка на фрагмент программы в указанном выше формате, а вторым элементом является символьный номер данного перехода в фрагменте.

16.1.3 Хранение вспомогательной информации в файлах логической системы

Для хранения вспомогательной информации в системе служат группы файлов, называемые информационными блоками. Информационные блоки пронумерованы, и за каждым из них закреплена определенная смысловая нагрузка. В каждом информационном блоке файлы обозначаются некоторой буквой, после которой идут 4 восьмеричные цифры - номер файла в группе. Все такие файлы имеют расширение ".los". Используются следующие информационные блоки:

1. 1-й информационный блок хранит шрифты и текстовые заготовки, используемые в различных интерфейсах. Он образован единственным файлом I0000.los.

2. 2-й информационный блок хранит задачник системы. Он образован группой файлов Eijkl.los.

3. 3-й информационный блок хранит справочную информацию о логических символах. Он образован группой файлов Hijkl.los.

4. 4-й информационный блок - "почтовый ящик", используемый для сохранения технического протокола, передаваемого от одной вер сии системы к другой. Он состоит из единственного файла R0000.los.

5. 5-й информационный блок хранит справочные оглавления, используемые в различных интерфейсах системы. Он образован группой файлов Qijkl.los.

6. 6-й информационный блок хранит базу теорем. Он образован группой файлов Mijkl.los.

7. 7-й информационный блок хранит общий справочник по системе. Он образован группой файлов Wijkl.los.

8. 8-й информационный блок хранит базу приемов, реализованных на ГЕНОЛОГе. Он образован группой файлов Pijkl.los.

9. 9-й информационный блок хранит оглавление программ. Он образован группой файлов Tijkl.los.

10. 10-й информационный блок хранит архив задачника. Он образован группой файлов Aijkl.los.

11. 11-й - информационный блок - резервный.

12. 12-й информационный блок хранит контейнер системы (емкость для передачи от одной версии системы к другой различных компонент - приемов, программ, задач, справочных текстов и т.п.). Он образован группой файлов Cijkl.los.

13. 13-й информационный блок хранит битмэпы задач на анализ рисунков. Он образован группой файлов Dijkl.los.

Общая организация информационного блока

Информационный блок используется для хранения древовидной структуры данных. Эта структура данных состоит из объектоа двух типов - указателей и терминалов. Указатели представляют собой неконцевые вершины древовидной структуры, от которых ведут ориентированные ребра с метками к другим указателям либо терминалам. Из терминалов ребра не исходят. Выделен корневой указатель, и из него достижимы все остальные указатели и терминалы информационного блока.

Указатели информационного блока бывают двух типов - каталоги и списки. Каталог представляет собой вершину, из которой может вести очень большое число ребер - равное числу логических символов, поддерживаемому интерпретатором (в данной версии интерпретатора - 65536). Все эти ребра имеют своими метками различные логические символы; фактически число выходящих из каталога ребер может быть любым - от 0 до указанного максимума. Информационный блок может иметь не более одного каталога, причем этот каталог обязательно является корневым указателем. Каталога не имеют лишь первый и четвертый блоки.

Указатель-список обычно имеет существенно меньшее число выходящих из него ребер - как правило, не более 100. Хотя здесь нет жесткой границы, но использование списков с большим ветвлением не рекомендуется, так как это может замедлить работу системы. Меткой перехода из указателя-списка может служить не только логический символ (как в случае каталога), но и терм. Отличие от указателя-каталога состоит также в том, что по заданной метке от указателя-списка осуществляется переход не к единственному объекту, а к нескольким объектам, образующим упорядоченный набор. Операторы ЛОСа позволяют получать набор ссылок на объекты данного набора и работать с каждым таким элементом по отдельности. В большинстве случаев указанные наборы - одноэлементные.

Терминалы информационного блока делятся на текстовые и логические. Текстовый терминал хранит последовательность байт, кодирующую текст либо битмэп (например, при задании шрифта); логический терминал хранит набор термов и логических символов. Допускаются ссылки на один и тот же терминал из различных указателей (в отличие от указателей, на которые можно ссылаться лишь из одного указателя). Впрочем, кратные ссылки на терминалы почти нигде не используются.

Кодирование объектов информационного блока

Указатель - каталог информационного блока (если он есть) вынесен в отдельный файл этого блока. Номер этого файла всегда состоит из нулей - (0000). Каталог состоит из 65536 32-битных слов. Эти слова пронумерованы начиная с 1, причем i-е слово соответствует переходу из указателя по ребру, меткой которого служит i-й логический символ. Если переход по логическому символу из указателя отсутствует, то соответствующее слово состоит из нулей. Иначе оно адресует объект, к которому ведет ребро. В этом случае старший бит слова равен 1; следующие за ним 12 бит образуют четырехразрядный восьмеричный номер файла информационного блока, в котором находится объект. Младшие 19 бит равны смещению в указанном файле начала объекта.

Указатели-списки и терминалы хранятся в виде текста, начинающегося с 3 обнуленных байт (они используются при уплотнении информационного блока). Затем размещается 16-битное слово A, уточняющее тип объекта, задаваемого текстом. Наконец, после слова A размещается собственно запись объекта, называемая полем объекта. Пусть $A = (a_1, \ldots, a_{16})$. Если $a_1 = 0$, то объект представляет собой указа-

тель-список, иначе - терминал. Если $a_2 = 0$, то этот объект используется, иначе он представляет собой "мусор". Заметим, что при переводе объекта в категорию "мусора" в нем может быть размещена ссылка на новую, фактически используемую версию, и для выхода на нее интерпретатор будет сначала пробегать по цепочке ссылающихся друг на друга объектов типа "мусор". Ссылка на новую версию (смещение его начала в том же файле) размещается в первых 3 байтах поля объекта - "мусора". В случае терминала a_3 уточняет его тип: $a_3 = 0$ означает, что терминал текстовый, иначе - логический. В случае указателя $(a_5, \ldots, a_{16})$ - число байт в начале поля объекта, уже занятых для хранения переходов из данного указателя. В случае терминала $(a_5, \ldots, a_{16})$ - число ссылок на данный терминал из различных указателей информационного блока.

Из указателя-списка можно ссылаться лишь на объекты, размещенные в том же файле информационного блока, что и этот указатель. Как и в случае файлов блока программ, размеры одного файла информационного блока ограничены в данной версии интерпретатора 512 Кбайтами. Это ограничение вытекает из используемой адресации: 12-битный номер файла в блоке оставляет лишь 20 бит для указания смещения в одном файле, причем один из этих 20 бит используется для других целей. При необходимости размеры одного файла в блоке можно было бы увеличить, уменьшив в соответствующее число раз количество самих файлов. Большие файлы, приближающиеся по своему размеру к 512 Кбайтам, автоматически разрезаются интерпретатором на две части, со сдвигом нумерации последующих файлов. Так как из указателя-списка можно ссылаться только на объекты того же файла, то фактически за каждым файлом оказывается закреплена некоторая группа логических символов, по которым возможны переходы в него из корневого указателя-каталога (если он есть), Разрезание большого файла на две части происходит лишь тогда, когда за ним закреплено более одного логического символа.

Поле указателя-списка состоит из последовательности расположенных подряд полей меток, за которыми идет незанятая часть поля. Каждое поле метки начинается с 16-битного слова, хранящего длину следующей за ним части этого поля метки. Далее располагается сама метка (логический символ либо терм), представленная в том же формате, как и термы логического терминала (см.ниже). После метки размещается последовательность ссылок на те объекта, к которым имеется переход по данной метке из данного указателя-списка. Каждая такая ссылка представляет собой 24-битное смещение начала объекта (объект должен размещаться в том же файле, что и указатель - список).

Поле логического терминала состоит из 24-битных слов. Младшие 16 бит такого слова образуют номер логического символа либо переменной, представляемых данным словом. Старший байт слова - $(a_1, \ldots, a_8)$ - уточняющий. Если $a_1 = 0$, то слово представляет логический символ, иначе - переменную. Если $(a_7, a_8) = 00$, то после символа не идет скобки; если $(a_7, a_8) = 01$, то после символа идет открывающая скобка; если $(a_7, a_8) = 10$, то после символа идет закрывающая скобка. При работе с таким форматом записи термов оказываются необходимы слова, кодирующие отдельно идущие закрывающие скобки (если закрывающих скобок встречается подряд несколько). Младшие 16 бит этих слов содержат нули; $a_1 = 1$; $(a_7, a_8) = 10$. В конце логического терминала могут располагаться неиспользуемые обнуленные 24-битные слова.

Поле текстового терминала состоит из произвольной последовательности байт.

При работе с объектами информационных блоков операторы ЛОСа используют специальные ссылки на них. Каждая такая ссылка представляет собой набор длины

2, хранящийся в зоне задач. Этот набор образован двумя логическими символами A_1, A_2. Чтобы найти номер файла и смещение в нем, интерпретатор вычитает из номеров N_1, N_2 символов A_1, A_2 номер 260 логического символа "0". Возникающие при этом числа в 16-битном представлении записываются подряд. Старший бит полученного 32-битного слова заменяется на 1. Теперь следующие за старшим 12 бит дают номер файла, а младшие 19 бит - смещение в нем. Заметим, что в действительности такой способ кодирования может привести к необходимости использовать логические символы A_1, A_2 с номерами, большими максимально допустимой величины 65535. Однако, так как 15-й бит кода логического символа интерпретатором не используется, необходимый перенос попадает в него и сохраняется вплоть до декодирования.

16.1.4 Вспомогательные файлы

Кроме файлов информационных блоков и блока программ, логическая система использует ряд вспомогательных файлов, типы которых перечисляются ниже.

Учет файлов, требующих уплотнения

При любом изменении файла F блока программ либо информационного блока в файле U.los автоматически регистрируется информация о том, что файл F был изменен. Это помогает при уплотнении файлов (исключении из них объектов, отнесенных к категории "мусора") не рассматривать уже уплотненные ранее файлы.

В файле U.los хранится последовательность 32-битных слов, ссылающихся на измененные файлы (повторные ссылки не вводятся). Старший бит ссылки равен 0; следующие за ним 12 бит определяют четырехразрядный восьмеричный номер файла в блоке; младшие 19 бит определяют номер информационного блока либо указывают, что рассматривается блок программ - в этом случае данные биты обнулены.

Для уплотнения информационных блоков и блока программ предусмотрены специальные операторы (в случае информационного блока уплотняется только текущий блок, т.е. блок, активизированный до этого оператором "файл(актив...)"). После выполнения оператора из файла U.los исключаются все ссылки, относящиеся к только что уплотненному блоку.

Хранение названий логических символов и словарных фрагментов

Для сохранения названий логических символов, а также для сохранения словарных фрагментов, используемых анализатором текстов естественного языка, служат две группы файлов. Первая из них называется словарем логической системы; вторая - внешним словарем.

Словарь состоит из нескольких файлов вида Vi.los; i - номер файла словаря (эти файлы называются также блоками словаря). Каждый файл словаря обслуживает ровно 10000 логических символов: V1.los - первые 10000; V2.los - следующие 10000, и т.д. Фактически в настоящее время существует только V1.los, так как номера используемых логических символов пока существенно меньше 10000. Файл Vi.los имеет ровно 260000 байт. Первые 20000 байт образуют 10000 16-битных слов, являющихся номерами логических символов, расположенными в порядке лексикографического возрастания текстов названий этих символов. Используется начальный отрезок этих 20000 байт, после которого все байты (до 20000 -го) обнулены. Байты начиная с 20001 разбиты на 10000 групп по 24 байта каждая. Эти группы хранят названия логических символов; группа с номером i (если начинать нумерацию групп с 1) в файле

Vj.los хранит название символа с номером $10000(j-1)+i$. Название символа завершается группой обнуленных байт (если длина его меньше 24). Названия символов, длина которых больше 24, не допускаются.

Внешний словарь устроен совершенно аналогично словарю; его файлы имеют названия Si.los. Отличие между внешним словарем и словарем относится, главным образом, к использующим их операторам ЛОСа. Словарь нужен просто для идентификации названия логического символа с этим символом (т.е. с его номером). Внешний словарь используется в режиме разбиения слова на группу допустимых фрагментов, т.е.нужен для перечисления таких начинающихся с заданной точки слова подслов, каждое из которых может рассматриваться в качестве допустимого фрагмента.

16.1.5 Вспомогательные массивы

Кроме зоны задач и зоны программ, логическая система имеет ряд других вспомогательных массивов; некоторые из них перечисляем ниже.

Глобальный стэк

Структура и принципы использования глобального стэка интерпретатора были подробно описаны в разделе, посвященном отладчику ЛОСа. Глобальный стэк представляет собой обобщение обычного стэка, необходимое для организации режима перечисления; в обычном смысле стэком он не является.

Стэк выражений

Перед обращением к процедуре интерпретатора, реализующей некоторый оператор либо операторное выражение, ее входные данные пересылаются в специальный стэк, называемый стэком выражений. В этот же стэк заносится результат, если вычисляется значение операторного выражения. В отличие от глобального стэка, это - обычный стэк.

Буфер текстов

Для работы с текстовыми данными в оперативной памяти введен массив, называемый буфером текстов. Он состоит из 2000 32-битных слов. Старшие 16 бит в каждом таком слове не используются; младшие 16 представляются в виде (A_1 A_2 A_3), где A_1 есть 4-битный номер цвета фона; A_2 - 4-битный цвет символов; A_3 - 8-битный номер символа.

Кроме основного буфера текстов, введены два дополнительных буфера текстов (различаемые по номерам 1 и 2), а также буфер текстового редактора. Каждый из этих буферов, как и буфер текстов, содержит ровно 2000 32-битных слов. Первые два буфера используются для временного сохранения содержимого основного буфера; буфер текстового редактора используется для сохранения текстового фрагмента и многократного его использования при редактировании текста.

Каталог программ

Содержимое файла K.los, хранящего ссылки на корневые фрагменты программ логических символов в блоке программ, при запуске системы копируется в специальном

массиве, называемом каталогом программ. Если корневой фрагмент программы логического символа считывается в зону программ, то ссылка на этот фрагмент из каталога программ обновляется - она начинает указывать на положение фрагмента в зоне программ. Формат ссылки - такой же, как для ссылок между фрагментами в зоне программ. После исключения корневого фрагмента из зоны программ автоматически восстанавливается ссылка на него в блоке программ. При редактировании программы (в том числе в процессах автопрограммирования) происходит изменение ее в блоке программ. В зоне программ после этого могут оставаться старые, неизменные версии фрагментов. Во избежание ошибочного поведения системы, следует в таких случаях использовать операторы ЛОСа, сбрасывающие ветвь программы заданного логического символа и восстанавливающие ссылку на нее из каталога программ по блоку программ.

Словарь и внешний словарь

Для быстрой работы с текстовыми данными все файлы словаря и внешнего словаря считываются в массивы словаря и внешнего словаря, пронумерованные так же, как и соответствующие файлы.

Буфер рисунков

Для анализа одноцветных изображений введен массив в 40000 байт, называемый буфером рисунков. В него загружается (в стандартном монохроматическом формате) битмэп рисунка.

Буфер линий

Буфер линий используется для накопления результатов анализа содержимого буфера рисунков - выделения линий и особых точек. Он имеет 12500 32-битных ячеек. Первые 2500 из них образуют массив точек; последние 10000 - массив линий.

Первая ячейка массива точек хранит смещение первой свободной ячейки этого массива; далее идут подряд массивы точек. Первая ячейка массива точки хранит длину этого массива. Затем идут координаты точки - столбец и строка, и после них - ссылки (смещения в буфере линий) на линии с концом в данной точке. Завершает список ссылок на линии ноль (после этого нуля могут находиться несколько резервных ячеек массива точки).

Первая ячейка массива линий хранит смещение в буфере линий первой свободной ячейки этого массива. Далее располагаются подряд массивы линий. Первая ячейка массива линии хранит длину этого массива. Затем идут ссылки на начало и конец линии (смещения в массиве точек). Далее - указатель вертикальной (1) либо горизонтальной (0) развертки линии. После него - абсцисса в случае горизонтальной развертки и ордината в случае вертикальной развертки, указывающая начальную точку развертки линии (на каждом шаге развертки абсцисса либо, соответственно, ордината, будет увеличиваться на 1). Наконец, размещаются ординаты в случае горизонтальной развертки и абсциссы в случае вертикальной развертки для последовательно проходимых точек линии.

Массив текстов

Этот массив играет роль памяти, в которой сохраняются извлекаемые из буфера текстов фрагменты, для последующей вставки их в буфер текстов при сборке нужного текста. В отличие от прочих упомянутых выше массивов, он не инициализируется автоматически, а вводится по мере надобности и потом удаляется. Размеры его - 256 Кбайт.

Стэк установок на прерывание

Массив установок на прерывание разбит на шестерки 32-битных ячеек, хранящих информацию о ситуациях, в которых нужно выходить в отладчик ЛОСа. При каждом обращении к отладчику началом стэка становится первая свободная шестерка в конце массива, в которую и будет заноситься информация о прерывании, устанавливаемом для обращения к отладчику при анализе его же работы. При выходе из отладчика начало стэка откатывается на 6 ячеек. Первая из шести ячеек каждого кадра хранит номер типа прерывания либо (при отсутствии установки на прерывание) 0; вторая хранит смещение кадра глобального стэка, к которому относится установка, либо 0; ячейки с третьей по шестую хранят информацию, сопровождающую тип прерывания. Третья ячейка хранит логический символ, при выходе на программу которого выполняется прерывание, либо другую специальную информацию; четвертая - ссылку в блоке программ на фрагмент прерывания; пятая - ссылку в зоне программ на оператор прерывания (старшие 16 бит - смещение в сегменте зоны программ; младшие - номер сегмента); шестая - число на счетчике шагов работы интерпретатора, по достижении которого предпринимается прерывание.

Стэк прерываний

В этом стэке хранятся группы значений: (peresm, otvet, symbol, ind, ist, uroven, delta, cadr, смещение в стэке выражений T до его вершины st, текущая операция) - сохраняемых перед входом в прерывание.

16.1.6 Обозначения массивов интерпретатора

Перечислим обозначения основных массивов, используемых интерпретатором ЛОСа.

1. Зона задач - Z;
2. Глобальный стэк - S;
3. Стэк выражений - T;
4. Зона программ состоит из сегментов (массивов), массив указателей на которые есть P[NP];
5. Каталог программ - K;
6. Словарь состоит из массивов, массив указателей на которые есть V[NV];
7. Внешний словарь состоит из массивов, массив указателей на которые есть SL[NSL];
8. Буфер текстов - B;

9. Буфер рисунков - RS;

10. Буфер линий - LN;

11. Стэк установок на прерывания - PR;

12. Стэк прерываний - SP;

13. Массив текстов - MT;

14. Вспомогательные массивы R,F,C, используемые для временного хранения различных данных. Первые два из них имеют по 60000 байт, последний (называемый буфером битмэпов, так как обычно в нем создаются битмэпы непосредственно перед прорисовкой функциями WINAPI) - 76800 байт.

16.1.7 Основные регистры интерпретатора

Перечислим основные регистры интерпретатора и их названия (более подробные пояснения, относящиеся к регистрам, будут даваться по мере описания процедур интерпретатора).

1. ind - переключатель обращений в основном цикле работы интерпретатора: 0 - к сканированию задачи; 1 - к выполнению фрагмента программы; 2 - выход для ввода внешней информации; 3 - завершение программы; 4 - прерывание при обнаружении ошибки.

2. peresm - указатель на необходимость пересмотра в сканировании задачи (1 - пересмотр, 0 - нет пересмотра);

3. otvet - регистр ответа задачи (0 - нет ответа);

4. cadr - смещение в глобальном стэке текущего кадра;

5. symbol - логический символ, программа которого реализуется;

6. nomprog - номер текущего сегмента зоны программ, в котором находится реализуемый фрагмент; нумерация начинается с 0;

7. A - указатель на начало текущего сегмента зоны программ;

8. fragm - смещение начала текущего фрагмента программы в текущем сегменте зоны программ;

9. tec - указатель на текущую ячейку текущего фрагмента программы либо (в паузах между обработкой фрагментов, например, при считывании фрагмента в зону программ) 0;

10. level - текущий уровень перечисления. Уровни перечисления вводятся для того, чтобы можно было отличить перечисление, организуемое отдельным оператором, от внутреннего перечисления, обеспечиваемого его подоператором. При входе в выполнение составного перечисляющего оператора (например, оператора "или") уровень перечисления увеличивается на единицу, так что кадры

вспомогательных перечисляющих операторов внутри этого "или" будут помечены ненулевыми уровнями перечисления. Такая разметка позволяет игнорировать вспомогательные кадры перечисляющего типа при откатах и сбрасывать заданное количество лишь основных (имеющих нулевой уровень перечисления) кадров. Напомним, что перечисление обеспечивается либо кадром продолжения, либо кадром оператора, программа которого предназначена для перечисления;

11. st - указатель на первую незанятую ячейку стэка выражений (он заполняется в порядке увеличения смещений ячеек);

12. svob - смещение в зоне задач ячейки, начиная с которой происходит поиск свободного места при вводе новых текстов;

13. lcadr - смещение в глобальном стэке последнего стэкового кадра;

14. ist - регистр истинностного значения оператора (0 - ложь, 1 - истина);

15. step - счетчик шагов работы интерпретатора;

16. suff - указатель на первую ячейку суффикса текущего фрагмента в зоне программ;

17. svobprog - номер сегмента зоны программ, в которую будет считываться очередной фрагмент из блока программ;

18. svobfrag - смещение в сегменте зоны программ, указанном регистром svobprog, начальной точки для записи считанного из блока программ фрагмента;

19. prer - смещение в стэке PR установок на прерывания текущего стэкового кадра;

20. tact - указатель такта закрепления в текущем цикле поиска места в зоне задач. Этот указатель поочередно принимает значения 0x80000000 и 0x80000001. Смена значения происходит каждый раз, когда при поиске свободного места достигается конец зоны задач. Подробнее - см. описание процедур поиска места и расчистки мусора в зоне задач.

21. tipprer - регистр типа обращения к отладчику.

22. uroven - регистр уровня обращения. Если его содержимое отлично от 0, то оно определяет максимальный уровень задачи, к которой произойдет ближайшее обращение. После обращения автоматически устанавливается на 0.

23. input - регистр типа возвращения к основному циклу интерпретатора: 0 - простое возвращение; 1 - возвращение в оператор "клавиатура"; 2 - возвращение в оператор "меню"; 3 - возвращение после оператора "карандаш";

24. shift - флаг нажатой клавиши "shift"; используется в драйвере клавиатуры;

25. ctrl - флаг левой клавиши "ctrl";

26. ctrr - флаг правой клавиши "ctrl";

27. rus - флаг русского (1) либо латинского (0) регистров;

28. stroka - номер текущей строки на экране (от 0 до 479);

29. stolbec - номер текущего столбца на экране (от 0 до 639);

30. rectleft, rectright, recttop, rectbottom - левый, правый, верхний и нижний края активной рамки экрана;

31. delta - приращение счетчика шагов (0 при выключенном счетчике, иначе 1);

32. stp - смещение в стэке прерываний первой свободной ячейки;

33. mousex,mousey - столбец и строка курсора мыши;

34. mouse - тип сообщения от мыши: 0 - нажатие левой кнопки, 2 - нажатие правой кнопки;

35. stop - номер шага, на котором предпринимается анализ необходимости откатов согласно ранее выполненным операторам "лимит";

36. nachl - индикатор периода до первого внутреннего перезапуска: сначала 0, а затем 1;

37. iMT - индикатор наличия массива текстов: 0 - нет массива, 1 - есть массив;

38. jMT - смещение в массиве текстов MT первой свободной ячейки;

39. kontrhol - индикатор режим контроля холостого хода приемов;

40. kontrpr - индикатор обращения к программе оператора ЛОСа "контрольприема"; если он равен 1, то обращения к этому оператору происходят, иначе - нет;

16.2 Преобразование программы ЛОСа в формат, используемый интерпретатором

Операторы ЛОСа, представляющие собой некоторые термы, не удобны для быстрого выполнения определяемых ими вычислений. Для использования интерпретатором эти операторы преобразуются в специальный формат - набор 32-битных слов; в этом формате они хранятся как в зоне программ, так и (после описанного выше сжатия) в блоке программ. Опишем формат, соответствующий хранению фрагмента программы в зоне программ.

16.2.1 Переменные и логические символы

Программная переменная либо логический символ представляются 32-битным словом. В случае переменной это слово просто является номером переменной; в случае логического символа старшие 16 бит образуют набор $(0, \ldots, 0, 1, 0)$, а младшие 16 бит образуют номер символа.

16.2.2 Непосредственно реализуемые неперечисляющий оператор либо операторное выражение

Непосредственно реализуемые (т.е. обрабатываемые специальной функцией интерпретатора) неперечисляющий оператор либо операторное выражение $f(A_1 \dots A_n)$ преобразуются в последовательность 32-битных кодов зоны программ (логических символов, переменных либо номеров процедур интерпретатора) следующим образом. Сначала располагаются последовательности $B_1, \dots, B_n$, где B_i получено преобразованием операторного выражения A_i. Если число операндов n - плавающее, то затем размещается код символьного представления для n (т.е. код логического символа с номером $260+n$). Далее идет код номера процедуры интерпретатора для реализации f (коды номеров процедур интерпретатора имеют в старших 16 битах набор $(0, \dots, 0, 1)$ а в младших - сам номер). Наконец, если вхождение $f(A_1 \dots A_n)$ в оператор фрагмента программы - корневое, в конце добавляется код номера 243 процедуры интерпретатора - эта процедура реализует откат, если оператор оказался ложным.

16.2.3 Отрицание оператора

Отрицание "не(A)"оператора A кодируется последовательностью, получаемой добавлением в конце последовательности, кодирующей оператор A, кода номера 298 процедуры интерпретатора (эта процедура меняет значение регистра истинности ist на противоположное).

16.2.4 Дизъюнкция операторов

Оператор "или($A_1 \dots A_n$)" кодируется последовательностью вида $K_1, B_1, K_2, B_2, \dots, B_n, K_{n+1}$. Здесь $B_1, \dots, B_n$ - последовательности, кодирующие операторы $A_1, \dots, A_n$. Фрагмент K_1 состоит из двух 32-битных слов; фрагменты $K_2, \dots, K_n$ - из четырех таких слов каждый; фрагмент K_{n+1} - из трех слов. Первое слово в K_1 является кодом символьного представления смещения при переходе от этого слова к третьему слову фрагмента K_2 (смещение измеряется в числе 32-битных слов). Второе слово фрагмента K_1 является кодом номера 247 процедуры интерпретатора. Первое слово фрагмента K_i при $2 < i < n+1$ является кодом символьного представления смещения при переходе от этого слова на первый код после K_{n+1}. Второе слово в K_i - код номера 248 процедуры интерпретатора. Третье слово в K_i - код символьного представления смещения при переходе от этого слова к третьему слову фрагмента K_{i+1}. Четвертое слово в K_i - код номера 249 процедуры интерпретатора. Первые два слова фрагмента K_{i+1} устроены так же, как у $K_2, \dots, K_n$, а третье слово является кодом номера 246 процедуры интерпретатора.

16.2.5 Конъюнкция операторов

Оператор "и($A_1 \dots A_n$)" кодируется последовательностью вида $K_1, B_1, B_2, \dots, B_n, K_2$. Здесь $B_1, \dots, B_n$ - последовательности, кодирующие операторы $A_1, \dots, A_n$, к каждой из которых в конце добавлен код номера 243 процедуры интерпретатора. Фрагменты K_1, K_2 состоят каждый из двух 32-битных слов. Первое слово фрагмента K_1 есть код символьного представления смещения от этого слова до второго слова фрагмента K_2. Второе слово фрагмента K_1 есть код номера 244 процедуры интерпретатора. Слова фрагмента K_2 суть коды номеров 245 и 246 процедур интерпретатора.

16.2.6 Оператор "альтернатива"

Оператор "альтернатива($A_1\ A_2\ A_3$)" кодируется последовательностью $B_1, K_1, B_2, K_2, B_3, K_3$. Здесь B_1, B_2, B_3 - последовательности, кодирующие операторы A_1, A_2, A_3. Фрагменты K_1, K_2 состоят из четырех 32-битных слов; фрагмент K_3 - из трех 32-битных слов. Первое слово фрагмента K_1 является кодом символьного представления смещения при переходе от него к третьему слову набора K_2. Второе слово фрагмента K_1 - код номера 295 процедуры интерпретатора. Третье слово - код символьного представления смещения от него до последнего слова фрагмента K_3. Четвертое слово фрагмента K_1 - код номера 247 процедуры интерпретатора. Первое слово фрагмента K_2 - код символьного представления смещения от него до первого слова после K_3. Второе слово фрагмента K_2 - код номера 256 процедуры интерпретатора. Третье слово фрагмента K_3 - код символьного представления смещения от него до последнего слова набора K_3. Четвертое слово - код номера 247 процедуры интерпретатора. Первое слово фрагмента K_3 - код символьного представления смещения от него до первого слова после K_3 (т.е. 3); второе и третье слова - коды номеров 256 и 246 процедур интерпретатора.

16.2.7 Оператор "длялюбого"

Оператор "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)" кодируется последовательностью $K_1, B_1, \dots, B_m, B_0, K_2$. Здесь $B_1, \dots, B_m, B_0$ - последовательности, кодирующие операторы $A_1, \dots, A_m, A_0$, причем к концу каждой из последовательностей $B_1, \dots, B_n$ добавлен код номера 243 процедуры интерпретатора. Фрагмент K_1 состоит из четырех 32-битных слов; K_2 - из двух слов. Первое слово фрагмента K_1 - код символьного представления смещения от него к последнему слову фрагмента K_2. Второе слово фрагмента K_1 - код номера 250 процедуры интерпретатора. Третье и четвертое слова фрагмента K_1 - коды переменных x_1 и x_n (в кванторных операторах номера переменных связывающей приставки образуют отрезок натурального ряда, так что вся эта приставка однозначно восстанавливается по данным кодам). Фрагмент K_2 состоит из кодов номеров 294 и 251 процедур интерпретатора.

16.2.8 Оператор "существует"

Оператор "существует($x_1 \dots x_n A$)" кодируется последовательностью K_1, B, K_2. Здесь B - последовательность, кодирующая оператор A. Фрагмент K_1 состоит из четырех 32-битных слов; фрагмент K_2 - из двух 32-битных слов. Первое слово фрагмента K_1 - код символьного представления смещения от него до второго слова фрагмента K_2. Второе слово фрагмента K_1 - код номера 250 процедуры интерпретатора. Третье и четвертое слова фрагмента K_1 - коды переменных x_1 и x_n. Первое и второе слова фрагмента K_2 - коды номеров 254 и 246 процедур интерпретатора.

16.2.9 Операторное выражение "вариант"

Операторное выражение "вариант($A_1\ A_2\ A_3$)" кодируется последовательностью B_1, K_1, B_2, K_2, B_3. Здесь B_1, B_2, B_3 - последовательности, кодирующие оператор A_1 и операторные выражения A_2, A_3. Фрагменты K_1, K_2 состоят из двух 32-битных слов. Первое слово фрагмента K_1 есть код символьного представления смещения от этого слова до первого слова фрагмента B_3. Второе слово фрагмента K_1 есть код номера

295 процедуры интерпретатора. Первое слово фрагмента K_2 есть код символьного представления смещения от этого слова до первого слова после B_3. Второе слово фрагмента K_2 есть код номера 257 процедуры интерпретатора.

16.2.10 Операторные выражения "выписка", "перечисление", "суммавсех"

Операторные выражения "выписка($x_1 \dots x_n\ A\ t$)"; "перечисление($x_1 \dots x_n\ A\ t$)"; "суммавсех($x_1 \dots x_n\ A\ t$)" кодируются последовательностью K_1, B_1, B_2, K_2. Здесь B_1, B_2 - последовательности, кодирующие оператор A и операторное выражение t. Фрагмент K_1 остоит из пяти 32-битных слов; фрагмент K_2 - из двух таких слов. Первое слово фрагмента K_1 есть код номера 258 (в случаях "выписка", "перечисление") либо номера 264 (в случае "суммавсех") процедуры интерпретатора. Второе слово фрагмента K_1 есть код символьного представления смещения перехода от этого слова к последнему слову фрагмента K_2. Третье слово фрагмента K_1 есть код номера 259 процедуры интерпретатора. Четвертое и пятое слова фрагмента K_1 суть коды переменных x_1 и x_n. Первое слово фрагмента K_2 есть код номера 261 (случай "перечисление") либо номера 262 (случай "выписка") либо номера 265 (случай "суммавсех") процедуры интерпретатора. Второе слово фрагмента K_2 есть код номера 263 (в случаях "выписка", "перечисление") либо номера 266 (в случае "суммавсех") процедуры интерпретатора.

16.2.11 Программируемые оператор либо операторное выражение

Реализованные с помощью ЛОС-программы оператор либо операторное выражение $f(A_1 \dots A_n)$ кодируются последовательностью $B_1, \dots, B_n, K$. Здесь $B_1, \dots, B_n$ - коды для $A_1, \dots, A_n$. Фрагмент K состоит из трех 32-битных слов; если рассматриваемое вхождение оператора $f(A_1 \dots A_n)$ является корневым, то к этому добавляется четвертое слово - код номера 243 процедуры интерпретатора. Первое слово фрагмента K есть код логического символа f; второе слово - код символьного представления числа операндов n. Третье слово есть код номера 293 (в случае оператора) либо 294 (в случае операторного выражения) процедуры интерпретатора.

16.2.12 Непосредственно реализуемый перечисляющий оператор

Непосредственно реализуемый перечисляющий оператор $f(A_1 \dots A_n)$ кодируется последовательностью $B_1, \dots, B_n, K$. Здесь $B_1, \dots, B_n$ - коды для $A_1, \dots, A_n$. Если число операндов n для f не фиксировано, то в начале фрагмента K располагается код символьного представления числа n. Далее располагаются коды двух номеров N_1, N_2 процедур интерпретатора. Первый из этих номеров является номером логического символа f. Соответствующая процедура обеспечивает присвоение значений переменным в начале перечисления ("инициализация цикла"). Второй номер относится к процедуре, обеспечивающей переприсвоение значений на очередном шаге перечисления ("шаг цикла"). Эти процедуры расположены в конце перечня функций интерпретатора Fi - начиная с номера $i = 268$, быть может, с некоторыми паузами, и

легко находятся по сопровождающим их комментариям вида "шаг f". В случае корневого вхождения оператора фрагмент K завершается кодом номера 243 процедуры интерпретатора.

16.2.13 Операторы перехода "ветвь", "иначе"

Оператор "ветвь N" кодируется тройкой K_1, K_2, K_3. K_1 есть код номера 290 процедуры интерпретатора; K_2 - код символьного представления числа N; K_3 - код номера 291 процедуры интерпретатора.

Оператор "иначе N" кодируется парой K_1, K_2. K_1 есть код номера 297 процедуры интерпретатора; K_2 - код символьного представления числа N.

16.3 Общая схема функционирования интерпретатора

При запуске программы интерпретатора применяется функция initsolv(), которая расчищает зону задач и инициализирует регистры интерпретатора. Эта же функция используется при внутреннем перезапуске системы. Она не затрагивает зоны программ.

Основной цикл работы интерпретатора - функция maincycle(), обращения к которой имеются при запуске программы и при обработке сообщений операционной системы. Перед завершением программы применяется функция remove(), обеспечивающая освобождение памяти.

При вводе новых регистров интерпретатора следует обеспечивать их инициализацию, добавляя необходимые операторы к функции initsolve.

Процедура maincycle() состоит из единственного цикла, управляемого регистром ind. Значение ind = 0 означает, что следует обращаться к сканированию текущей задачи, ind = 1 - к обработке текущего фрагмента программы, ind = 2 - запрашивать внешнюю информацию, ind = 3 - завершать работу системы, ind = 4 - организовывать выход в отладчик ЛОСа. Для сканирования задачи служит процедура scanpr(); для обработки текущего фрагмента программы - interp(); для выхода в отладчик ЛОСа - initprer(n), где в случае обращения из процедуры maincycle параметр n равен 1.

16.3.1 Сканирование задачи

Сканирование задачи осуществляется процедурой scanpr(). Прежде всего, эта процедура проверяет содержимое регистра ответа otvet. Если оно отлично от 0, то является найденным ответом задачи (логическим символом либо смещением начала терма в зоне задач). Выдача этого ответа осуществляется процедурой endpr(). Если ответ был найден для исходной задачи, то работа программы завершается (ind := 3, и далее через цикл процедуры maincycle). Если ответ был получен для промежуточной задачи, то он регистрируется в стэке выражений, и регистр ответа обнуливается. Если же ответ не найден, то выполняется шаг сканирования: либо (в начале сканирования) обращение к программе типа задачи, либо смена текущего терма задачи и рассмотрение его первого символа, либо смена текущего символа в текущем терме - так, как это указано в подразделе "Сканирование задачи" главы "Общая схема функционирования решателя".

Переход к очередному терму и рассмотрение его первого символа выполняются процедурой scanblock(). В свою очередь, эта процедура использует вспомогательную процедуру scanstep(a), выполняющую переход к очередному терму задачи в кадре глобального стэка, соответствующем данной задаче, и выдающую новое смещение a вхождения веса терма в соответствующий список задачи. Рассмотрение каждого последующего символа текущего терма выполняется уже процедурой scanpr. Фактически "рассмотрение" текущего логического символа в указанных процедурах сводится к тому, что этот символ заносится в регистр symbol (символ, программа которого реализуется); регистр ind устанавливается на 1 (режим выполнения ЛОС-программы), и в регистр tec (указатель на текущую ячейку зоны программ) заносится 0 для инициализации поиска начального фрагмента программы символа symbol. Затем предпринимается выход в цикл процедуры maincycle, которая далее переда ет управление процедуре interp.

16.3.2 Обработка текущего фрагмента программы

Рабочий цикл интерпретатора, связанный с последовательным выполнением операторов текущего фрагмента программы, реализуется функцией interp().

Прежде всего, проверяется наличие в регистре tec ссылки на текущий оператор фрагмента. Если этот регистр равен 0, то происходит обращение к процедуре nachprog(). Она считывает в зону программ корневой фрагмент программы символа, хранящегося в регистре symbol (если такой фрагмент уже не считан в зону программ). Затем устанавливаются значения следующих регистров, необходимые для выполнения фрагмента программы:

а) nomprog - номер текущего сегмента зоны программ (нумерация сегментов начинается с 0);

б) fragm - смещение начала текущего фрагмента в текущем сегменте;

в) A - указатель на начало текущего сегмента зоны программ;

г) suff - указатель на первую ячейку суффикса текущего фрагмента (используется при определении переходов из фрагмента);

д) tec - указатель на текущую ячейку текущего фрагмента.

Далее начинается цикл загрузки стэка выражений входными данными для очередной процедуры интерпретатора, реализующей оператор либо операторное выражение ЛОСа. Если tec указывает на программную переменную xi, значение которой определено, то в стэк выражений пересылается это значение, иначе в стэк выражений пересылается набор, старшие 16 бит которого имеют вид (10 ... 0), а младшие хранят номер переменной. Такой набор указывает процедуре интерпретатора, что соответствующее выходное значение должно быть передано переменной xi. Если tec указывает на логический символ, то этот символ заносится в стэк выражений. По окончании загрузки tec будет указывать на номер применяемой процедуры интерпретатора.

При переходе к выполнению очередной процедуры интерпретатора, реализующей ЛОС - вычисления, предпринимается увеличение на величину delta регистра step счетчика шагов интерпретатора. Обычно delta хранит 1, однако при входе в прерывание отладчика ЛОСа delta временно устанавливается на 0 - чтобы процесс отладки не влиял на "пошаговую адресацию" анализируемой траектории.

Далее предпринимается контроль наличия сообщений Windows, требующих коррекции регистров (например, переключение режимов русской и латинской клавиатуры); контроль необходимости отката при превышении заданного лимита шагов, а

также обработка ранее введенных установок на прерывание для выхода в отладчик ЛОСа. При отсутствии прерывания происходит обращение к той процедуре интерпретатора, на номер которой указывает tec. Здесь используется массив inter[ ] указателей на функции Fi, реализующие ЛОС - процедуры интерпретатора.

Цикл обработки фрагмента программы продолжается до тех пор, пока либо регистр ind не становится отличен от 1, либо tec не оказывается равным 0 или установленным на начало суффикса фрагмента. В первых двух случаях - немедленный выход в maincycle. В последнем случае, если сканируется задача, то ind устанавливается на 0, и (через maincycle) выход в продолжение сканирования. Если реализуется оператор ЛОСа, то регистр истинности ist устанавливается на 0 (ложное значение); если реализуется операторное выражение, то в стэк выражений заносится его фиктивное значение - логический символ 0. В обоих случаях перед выходом в maincycle применяется процедура contin(), организующая продолжение вычислений по источнику текущего кадра глобального стэка.

16.3.3 Функции вызова фрагментов в зону программ

Для считывания фрагмента в блоке программ и перенесения его в зону программ служит функция fragment(a). Она получает в качестве значения a ссылку на фрагмент в блоке программ и возвращает ссылку на начало фрагмента в зоне программ (при неудаче возвращает 0).

Номер того сегмента зоны программ, в который считывается очередной фрагмент программы, хранится в регистре svobprog. Смещение в этом сегменте точки, начиная с которой записывается фрагмент, хранится в регистре svobfrag. Начиная с этой точки и до конца данного сегмента все ячейки - неиспользуемые. Если для записи фрагмента не хватает места, то происходит переход к следующему (в циклическом порядке) сегменту зоны программ. Следующий сегмент расчищается: сначала в нем находятся все уже вовлеченные в вычисления фрагменты программы, которые закрепляются и сдвигаются (сплошным образом друг за другом) в начало сегмента; соответственно корректируются все ссылки на прерванные вычисления в данных фрагментах. Зона после закрепленных и перенесенных в начало сегмента фрагментов - свободная для записи новых фрагментов.

Расчистка очередного сегмента осуществляется функцией raschistca(). Закрепление фрагмента, расположенного в сегменте с номером a по смещению b, выполняется функцией zacrep(a, b); при этом закрепляются также все надфрагменты данного фрагмента. Функция perenos(a) получает смещение a начала заполняемого фрагмента в текущем сегменте (по svobprog) зоны программ. Она переходит к следующему сегменту, расчищает его, переносит в него указанное начало фрагмента, и возвращает новое смещение этого начала.

16.3.4 Поиск свободного места и расчистка мусора в зоне задач

Для поиска свободного места в зоне задач служит функция mem(a). Ей сообщается длина a (в 32-битных словах) необходимого для размещения нового текста отрезка; в эту длину включаются префикс и суффикс текста. Функция создает в зоне задач пустую заготовку текста, с введенными префиксом и суффиксом, и возвращает смещение в зоне задач начала этого текста (смещение - в 32-битных словах). При

необходимости выполняется уплотнение зоны задач (все необходимые для дальнейшего тексты сдвигаются в начало зоны). Если, несмотря на предпринятое уплотнение, места не хватило, возвращается 0 и регистр ind устанавливается на 4 (для последующего выхода в отладчик ЛОСа и выдачи сообщения о переполнении зоны задач).

Функция mem(a) прежде всего обращается к функции mesto(a), которая собственно и пытается найти необходимое место. Для этого используется регистр svob, в котором хранится смещение текста зоны задач, начиная с которого ведется поиск свободного места. Префикс каждого текста содержит в младшем бите второго 32-битного слова указатель такта закрепления. Если этот указатель не совпадает с младшим битом регистра tact, то текст заведомо является "мусором". Функция "mesto(a)", перемещаясь в направлении к концу зоны задач от текста, указанного в svob, пытается составить из нескольких идущих подряд текстов "мусора" отрезок необходимой длины. Если такой отрезок найден, то начальная часть его преобразуется в заготовку нового текста (с указателем такта закрепления, равным младшему биту регистра tact), а конец - помечается как "мусор", и ссылка на него заносится в svob.

Если необходимый отрезок не найден, то это еще не означает, что нужно уплотнять зону задач - просто поиск мог начаться с позиции, близкой к концу этой зоны. Поэтому сначала предпринимается переустановка svob на начало зоны задач. Одновременно значение младшего бита регистра tact меняется на противоположное. При этом автоматически все тексты, которые были введены на последнем цикле просмотра зоны задач, попадают в категорию "мусора". Так как нормальный режим работы системы - очень мало заполненная зона задач, то большинство перенесенных таким образом в "мусор" текстов и в самом деле должны быть отнесены к "мусору". Для дальнейшего использования нужна лишь малая их доля. Поэтому экономнее выполнить просмотр и закрепление (путем установки указателей такта закрепления по регистру tact) всех нужных текстов, а ненужные не просматривать вовсе. К категории нужных текстов здесь относятся все такие тексты, к которым возможен переход по цепочкам ссылок, начинающимся в глобальном стэке, стэке выражений и в ряде специальных регистров интерпретатора - прочие тексты все равно для программы уже недостижимы.

Указанные выше действия по смене такта закрепления выполняются функцией retact(). Для закрепления нужных текстов служит функция zac(a) - она закрепляет все тексты, достижимые по цепочкам ссылок из текста, у которого смещение начала равно a.

Если, несмотря на смену такта закрепления, свободное место в зоне задач не найдено, предпринимается попытка уплотнения зоны задач. При уплотнении все нужные тексты переписываются подряд в начале зоны задач, с необходимой коррекцией ссылок на них - как из них самих, так и из стэков и регистров. Уплотнение осуществляется функцией uplotn(); для коррекции ссылок применяется функция correctadr(a).

16.4 Добавление нового оператора либо операторного выражения

Для добавления нового оператора либо операторного выражения, непосредственно реализуемого интерпретатором, следует прежде всего выбрать заголовок этого оператора либо операторного выражения - через интерфейс словаря логической системы. Номер такого заголовка должен быть не больше 240 - заголовки с большими но-

мерами воспринимаются интерпретатором как указания на обращения к программе ЛОСа для соответствующего логического символа.

Затем следует присоединить к программе интерпретатора функцию "Fn", где n - десятичная запись номера выбранного заголовка, реализующую новый оператор (операторное выражение). Если это - перечисляющий оператор, то данная функция реализует лишь первый шаг перечисления. Последующие шаги будут обеспечиваться еще одной функцией "Fm", где m - номер, больший последнего из введенных на текущий момент для F-функций интерпретатора.

В массиве inter[ ] указателей на F-функции, расположенном в конце программы интерпретатора, следует ввести названия новых F-функций Fn,Fm (нужные позиции легко устанавливаются по номерам соседних F-функций).

В массиве konv[ ] указателей на функции, используемые для преобразования операторов ЛОСа в формат, используемый интерпретатором (функции "конвертора"), на n-й позиции размещается ссылка на необходимую функцию Kp (она определяется по комментариям к имеющимся K-функциям либо аналогии с ранее введенными операторами и операторными выражениями). Заметим, что K-функция для каждого перечисляющего оператора создается индивидуальная (хотя и состоящая из единственного обращения к функции konvperechisl), так как в ней содержится ссылка на функцию Fm, обеспечивающую продолжение перечисления (последний операнд обращения к оператору konvperechisl имеет вид 0001M, где M - шестнадцатиричная запись числа m).

В массиве dekonv[ ] указателей на функции, используемые для перехода от формата интерпретатора к представлению операторов ЛОСа термами, на n-й позиции размещается ссылка на необходимую функцию Dq (она определяется по комментариям к имеющимся D-функциям либо по аналогии с ранее введенными операторами и операторными выражениями).

16.5 Вспомогательные функции интерпретатора

В этом разделе перечисляются, для удобства ориентации в программе интерпретатора, используемые им вспомогательные функции.

16.5.1 Функции, используемые для организации общих действий интерпретатора

1. initsolv() - инициализация зоны задач, стэков и основных регистров;
2. maincycle() - основной цикл интерпретатора;
3. scanpr() - основной цикл сканирования задачи;
4. ves() - возвращает смещение ячейки, хранящей вес текущего терма задачи;
5. scanblock() - ищет очередной терм задачи и организует переход к рассмотрению его первого символа;
6. scanstep(a) - a есть 0 либо смещение веса текущего терма задачи; реализуется переход к очередному текущему терму и возвращается новое смещение его веса либо 0;

7. endpr() - выдача ответа на задачу;

8. contin() - возвращение к программе источника текущего кадра глобального стэка по завершении всех действий (включая перечисление), связанных с этим кадром;

9. interp() - рабочий цикл интерпретатора;

10. nachprog() - подготовка вычислений по корневому фрагменту программы символа symbol;

11. fragment(a) - считывание фрагмента из блока программ по ссылке a в зону программ;

12. raschistca() - расчистка очередного сегмента зоны программ для записи в него фрагментов из блока программ;

13. zacrep(a, b) - закрепление при расчистке в сегменте зоны программ, имеющем номер a, фрагмента программы, смещение которого в этом сегменте равно b;

14. perenos(a) - перенесение в следующий сегмент зоны программ фрагмента, для которого не хватило при считывании из блока программ места в текущем сегменте. a - начало фрагмента в текущем сегменте; при перенесении происходит расчистка следующего сегмента.

15. uchhol(a) - процедура используется в режиме учета холостого хода приемов ГЕНОЛОГа при очередном откате к оператору "контрольприема". a - число шагов, затраченных на неудачную попытку применения приема. Происходит регистрация попытки в комментарии (выписка ...) к посылкам исходной задачи.

16. otkat() - откат к продолжению перечисления по последнему кадру глобального стэка. Заметим, что перечисляющий оператор, к которому происходит откат, может оказаться программируемым, и тогда при откате нужно продолжать перечисление согласно этой "отложенной" программе. В свою очередь, ее последний перечисляющий оператор тоже может оказаться программируемым, и т.д. Таким образом, возможна цепочка "отложенных" кадров операторов, выполняющих перечисления, каждый предыдущий кадр в которой является источником последующего. Начинается эта цепочка с текущего кадра cadr той программы, в которой происходит откат. Последний ее кадр либо является источником кадра продолжения, завершающего глобальный стэк, либо сам завершает глобальный стэк. Во втором случае перечисление по последнему кадру закончено; он становится текущим, и оператор contin() обеспечивает переход к продолжению действий согласно его источнику. В любой из этих ситуаций при откате предпринимается сброс во всех кадрах данной цепочки значений программных переменных, определенных после того перечисляющего оператора, к которому происходит откат. Если последний кадр цепочки являлся источником завершающего глобальный стэк кадра продолжения, то он становится текущим кадром. Кроме собственно отката, оператор выполняет ряд вспомогательных действий (обращение к процедуре учета холостого хода; сброс комментария (буфер ...), и др.).

17. raschist() - сброс значений переменных источника последнего кадра продолжения, определенных после оператора этого кадра, и удаление последнего кадра продолжения.

18. perehod(a) - изменение регистров, необходимое для перехода из текущего фрагмента программы по ссылке с символьным номером a.

19. zac(a) - закрепление в зоне задач текста, смещение некоторой (не обязательно начальной) ячейки которого равно a; одновременно закрепляются все достижимые из него по цепочке ссылок тексты. Процедура используется при изменении регистра tact. Собственно закрепление состоит в изменении указателя такта закрепления на новое значение tact. Все тексты со старым значением указателя такта при этом автоматически попадают в категорию "мусора".

20. retact() - изменение регистра tact, перезакрепление всех фактически используемых текстов зоны задач, и установка регистра svob начала поиска свободного места на начало зоны задач.

21. mesto(a) - поиск в зоне задач места для текста длины a, включая суффикс и префикс этого текста. Если место найдено, возвращает смещение начала нового текста, в котором создает префикс и суффикс. Если место не найдено, возвращает 0. Изменение регистра tact и уплотнение зоны задач этой процедурой не выполняются.

22. mem(a) - поиск в зоне задач места для текста длины a. Это - основная функция поиска места; при необходимости она изменяет регистр tact и уплотняет зону задач. Возвращает смещение начала нового текста. Если, все же, место не найдено, то возвращает 0 и устанавливает регистр ind на 4 для последующего выхода в отладчик с сообщением о переполнении зоны задач.

23. uplotn() - уплотнение зоны задач. Все используемые тексты смещаются в начало этой зоны; корректируются все ссылки на эти тексты и на вхождения в них.

24. correctadr(a) - используется при уплотнении зоны задач; по старому адресу a (текста либо вхождения в текст) определяет новый адрес, по которому соответствующая точка окажется после уплотнения.

16.5.2 Функции, используемые при реализации операторов ЛОСа

Здесь перечисляются функции, используемые реализующими операторы ЛОСа функциями Fi.

1. opr(a, b) - присвоение в текущем кадре глобального стэка переменной a значения b. Здесь a имеет указанный выше формат ссылки на не определенную переменную - старшие 16 бит имеют вид (10 ... 0), а младшие 16 образуют номер переменной. Это - очень часто используемая функция; через нее происходит выдача значений выходных переменных операторов, реализуемых интерпретатором. Заметим, что размеры кадра глобального стэка могут оказаться недостаточными для размещения в нем значения переменной a. Тогда процедура выполняет расширение кадра (с небольшим запасом) и сдвиг всех последующих кадров стэка, с необходимой коррекцией ссылок.

2. ravntext(a, b) - сравнение двух текстов в зоне задач, определяемых смещениями a, b своих начал. При сравнении текстов предпринимается сравнение их элементов, расположенных на соответствующих позициях. Если эти элементы тоже оказываются текстами, то начинается (если не удается сразу установить идентичность элементов по идентичности их смещений) просмотр их соответствующих позиций, и т.д. - до полного просмотра всех достижимых из текстов объектов. Здесь важно, чтобы отсутствовали циклические ссылки из текстов на тексты, иначе процедура зациклится. Если при просмотре встречается ссылка не на текст, а на вхождение в него, то анализ элементов такого текста не выполняется - вхождения сравниваются просто как смещения в зоне задач. Поэтому в тех случаях, когда есть опасность появления циклов ссылок, при программировании на ЛОСе используются ссылки не на текст, а на вхождение первого его элемента (т.наз. "левыйкрай").

   При равенстве текстов процедура возвращает 1, иначе 0.

3. sravn(a, b) - сравнение объектов a, b. При равенстве возвращает 1, иначе 0. Для текстов обращается к функции ravntext.

4. slog(a, b, c) - Сложение десятичных чисел a, b (в формате ЛОСа, т.е. каждое из них - либо логический символ, определяющий цифру, либо смещение в зоне задач набора цифр, в котором могут также встречаться логические символы запятой и минуса). Возвращает (также в формате ЛОСа) сумму модулей чисел a, b, взятую со знаком "плюс" при $c = 0$ и "минус" при $c = 1$.

5. text1(a) - a есть указатель на некоторую ячейку стэка выражений. В зоне задач создается текст набора, разряды которого суть объекты стэка выражений начиная с a и кончая последней занятой ячейкой этого стэка (т.е. $st - 1$). Возвращает смещение начала этого текста (при нехватке места в зоне задач возвращает 0). Если $a = st$, то возвращает логический символ "пустоеслово". По окончании формирования текста в регистр st передается значение a (объекты, зарегистрированные в тексте, исключаются из стэка выражений). Эта и следующая функция - наиболее часто применяемые способы создания нового текста в зоне задач. Если в процессе формирования нового текста могут вводиться еще какие - то тексты, то до его создания могут произойти уплотнение зоны задач и связанная с этим коррекция всех ссылок на нее. Чтобы коррекция затронула и формируемый текст, его следует накапливать в стэке выражений, а по завершении накопления - использовать данную либо следующую функцию.

6. text2(a) - аналогично предыдущему, но считывание объектов из стэка выражений в текст происходит в обратном порядке (начиная с $st - 1$ и кончая a).

7. razn(a, b, c) - a, b суть десятичные числа в формате ЛОСа; модуль первого из них не меньше модуля второго. Возвращает разность модулей этих чисел, взятую со знаком "плюс" при $c = 0$ и со знаком "минус" при $c = 1$.

8. sravnmod(a, b) - a, b суть десятичные числа в формате ЛОСа. Если модуль первого из них не меньше модуля второго, возвращает 0, иначе возвращает 1.

9. summa(a, b) - a, b суть десятичные числа в формате ЛОСа. Возвращает их сумму.

10. nomer(a) - определяет десятичное число в формате ЛОСа, являющееся номером логического символа a.

11. lenman(a) - определяет длину мантиссы десятичного числа a, представленного в формате ЛОСа. Результат - обычное машинное 32-битное число.

12. dopnuli(a, b) - дописывает к концу мантиссы десятичного числа a, заданного в формате ЛОСа, b нулей, и возвращает смещение начала нового текста.

13. norman(a) - a есть указатель на ячейку стэка выражений, начиная с которой и до конца стэка (т.е. до $st-1$) расположена запись десятичного числа в формате ЛОСа. В ячейке a находится младший разряд этой записи. Предпринимается нормализация записи (отбрасывание нулей в конце мантиссы и т.п.), и возвращается указатель на начало в стэке выражений нормализованной записи.

14. udalmassiv(a) - a есть смещение в зоне задач начала текста. Если этот текст хранит ссылку на массив чисел машинного формата ("с плавающей запятой" либо "целое со знаком"), то происходит удаление этого массива.

15. udlin(a) - a есть смещение кадра продолжения, введенного при реализации операторного выражения "выписка" либо "перечисление" и имеющего ссылку на непустой накопитель набора, создаваемого указанным операторным выражением. Предпринимается удлинение этого накопителя на 10 разрядов.

16. novcadr(a, b) - создает в конце глобального стэка новый кадр оператора либо операторного выражения. a - тип кадра; в 7-ю ячейку этого кадра помещается значение b.

17. tekur() - возвращает текущий уровень последней (по глобальному стэку) задачи.

18. cadrprod(a) - создает в конце глобального стэка новый кадр продолжения длины a, используемый при реализации перечисляющего оператора.

19. prfx(a, b) - вводит текст, получающийся добавлением элемента a к началу набора b, и возвращает смещение его начала.

20. sffx(a, b) - вводит текст, получающийся добавлением элемента b концу набора a, и возвращает смещение его начала.

21. isklch(a, b) - вводит текст, получающийся из текста a исключением разряда по смещению b. Возвращает ссылку на этот текст.

22. cmntr(a, b) - a есть набор (возможно, "пустоеслово"); b - указатель на ячейку стэка выражений, начиная с которой и до конца стэка хранится текст комментария - набор либо единственный логический символ. Если этого комментария нет в наборе a, то возвращает 1, иначе возвращает 0.

23. spskprm() - в первых 16 ячейках массива R хранится маска переменных. Номера последовательно проходимых бит этой маски суть номера переменных. Создается текст списка тех переменных, позиции которых в маске помечены единицами, и возвращается смещение начала этого текста. Переменные в тексте упорядочены по возрастанию номеров.

24. svbvhg(a) - определяет, является ли свободным вохждение a переменной в некоторый терм. Если является, возвращает 1, иначе возвращает 0.

25. prmter(a) - регистрация в маске переменных всех свободных переменных терма a.

26. numkey(a) - a есть некоторый виртуальный код клавиатуры. Возвращает соответствующий ему номер строки в таблицах драйвера клавиатуры DK0[] - DK6[]. Если a не входит в таблицу DK0, возвращает 100.

27. tekfail(a) - a есть 32-битная ссылка на объект файла активного информационного блока: младшие 19 бит - смещение в файле, следующие за ними 12 бит - 4 восьмеричных цифры номера файла. Этот файл открывается; если его не было - вводится. В регистр infhndl заносится его handle, и корректируется номер файла в строке для его пути.

28. adrcod(a) - a есть ссылка на пару логических символов в зоне задач, кодирующую ссылку на объект файла информационного блока. Возвращает 32-битную ссылку на этот объект либо (при неудаче) 0.

29. codadr(a) - a есть 32-битная ссылка на объект файла информационного блока. Создает в зоне задач текст, состоящий из пары логических символов, кодирующих эту ссылку. Возвращает смещение начала этой пары. При неудаче возвращает 0.

30. readinf(a) - a есть 32-битная ссылка на объект файла информационного блока, для которого уже установлен регистр infhndl. Этот объект считывается в начало массива R. Возвращает скорректированную ссылку на считанный объект (если объект был достижим по цепочке косвенных ссылок через объекты, ранее помеченные как "мусор", то выдается прямая ссылка на конец этой цепочки). При неудаче возвращает 0.

31. poiskmetki(a) - в массиве R находится содержимое некоторого указателя - списка, считанного из информационного блока. a - логический символ либо переменная либо смещение начала терма в зоне задач. Предпринимается поиск метки a в указателе - списке, и если она найдена, то возвращается смещение в байтах в массиве R первой ячейки поля данной метки (эта ячейка хранит длину поля). При этом регистр $B1$ хранит смещение в байтах начала поля ссылок; $B2$ - длину поля метки. При неудаче возвращает 0.

32. sledmetka(a, b) - a есть число байт в поле указателя-списка, считанного в массив R; b - смещение в байтах поля метки этого указателя. Если метка не последняя, то возвращает смещение начала поля следующей метки. Иначе возвращает 0.

33. tekstmetki(a, b) - a есть смещение в байтах до текста метки указателя - списка, считанного в массив R. Если $b = 1$, то заносит в конец стэка выражений ссылку на эту метку - логический символ либо сформированный данной функцией в зоне задач текст терма. При переполнении зоны задач возвращает 0. Если $b = 0$, то ограничивается поиском конца текста метки и не меняет стэка выражений. В обоих случаях возвращает смещение в байтах до поля ссылок метки.

34. infterm(a, b) - a есть логический символ либо символ переменной либо смещение начала терма в зоне задач. b - смещение в байтах в массиве R той точки, начиная с которой нужно создать запись метки перехода a в формате указателя - списка. Такая запись создается, и возвращается смещение в R первого байта после нее.

35. novssilka(a, b) - a есть метка, по которой в заготовку указателя - списка, хранящуюся в массиве R, заносится ссылка на объект b (дано его смещение в файле информационного блока). Изменяет указатель числа занятых байт, но пока не корректирует общего числа байт в $R[0]$. Возвращает число использованных байт в R; при неудаче возвращает 0.

36. iskluchssilki(a, b) - a есть метка в указателе - списке, считанном в массив R. b - смещение в файле информационного блока объекта, на который происходит ссылка по метке a. Происходит сключение этой ссылки. Возвращает число занятых байт в R.

37. izmenenie(a, b) - a есть ссылка на объект в текущем информационном файле. Новая версия этого объекта хранится в массиве R, причем в $R[0]$ сохранено число байт старой версии. b - число байт новой версии в R. Предпринимается изменение версии объекта в информационном файле. Возвращает ссылку на новую версию либо, при неудаче, 0.

38. registracia(a, b, c) - происходит регистрация объекта текущего файла, ссылка на который есть c, по метке b в указателе, ссылка на который равна a.

39. poslednijfail() - устанавливает регистр infhndl для работы с последним файлом активного информационного блока. Возвращает шаблон ссылок на объекты этого файла (старшие 13 разрядов ссылки).

40. smfail(a, b) - a есть ссылка на объект файла информационного блока либо блока программ. b - 0 в случае блока программ, иначе - номер информационного блока. Реализует учет в массиве U и файле $U.los$ ссылки для уплотнения файла, содержащего a. В случае блока программ регистр proghndl, хранящий handle текущего файла блока программ, должен быть перед обращением установлен согласно a.

41. uplotninf(a, b) - уплотнение текущего файла активного информационного блока, считанного в массив $U2$. a - число байт в $U2$. Если информационный блок имеет указатель-каталог, то он считан в массив $U1$. Предпринимается коррекция ссылок в $U1, U2$. Возвращает число байт в $U2$ после уплотнения. b - шаблон ссылок на объекты рассматриваемого файла (старшие 13 разрядов).

42. bufsimv(a, b, c, d) - заносит в буфер текстов название логического символа либо обозначение переменной a начиная с ячейки номер b. c, d - цвет фона и символов, заданные символьными номерами. Возвращает номер первой ячейки в буфере текстов после занесенного фрагмента; при нехватке места возвращает 0.

43. prorisovka(a, b) - выдает на экран содержимое буфера текстов начиная с ячейки номер a и кончая ячейкой номер $b - 1$. Текст выдается с текущей позиции внутри текущей рамки. Возвращает номер ячейки буфера текстов, первой после выданной части текста (она может отличаться от b, если в рамке не хватило места).

44. bufbukva(a, b, c, d) - заносит в буфер текстов на позицию a букву b с цветом фона c и цветом символов d.

45. poiskslova(a) - a есть исходная позиция в буфере текстов. Если начиная с этой позиции расположено название логического символа, то возвращает этот символ и передает в регистр $B1$ смещение первой ячейки после названия символа, а в $B3$ - номер двух байт в начальной части сегмента словаря, хранящих номер логического символа; иначе возвращает 0. В последнем случае в регистр $B2$ передается номер первого сегмента словаря, имеющего неиспользованный логический символ, а в $B3$ - номер пары байт в начальной части сегмента, на которую следует вставить новый символ. Регистр $B2$ перед обращением устанавливается либо на $0xFFFFFFFF$, либо на номер сегмента словаря, в котором находится логический символ с изменяемым названием.

46. iskluchslova(a) - a есть исходная позиция в буфере текстов. В словаре ищется логический символ, название которого расположено начиная с данной позиции, и этот символ исключается из словаря. Возвращает данный логический символ, причем в регистр $B1$ заносится смещение в буфере текстов первой ячейки после названия символа. Если символ не найден, возвращает 0.

47. novprog() - устанавливает proghndl на последний файл блока программ. Возвращает шаблон ссылок на фрагменты этого файла (старшие 13 бит ссылки).

48. tekprog(a) - a есть ссылка на объект файла блока программ. Этот файл открывается, и proghndl становится равен его handl'у. Корректируется номер файла в строке для его пути.

49. isklprog(a) - a есть ссылка на фрагмент программы в блоке программ. Предпринимается исключение всей ветви этого фрагмента. Внешняя ссылка на фрагмент должна быть предварительно удалена.

50. uplotnprog(a, b) - уплотнение текущего файла блока программ, считанного в массив $U2$. a - число байт в $U2$. Каталог программ считан в массив K. Корректирует ссылки в $K, U2$. Возвращает число байт в $U2$ после уплотнения. b - шаблон ссылок на фрагменты уплотняемого файла (старшие 13 бит).

51. initprer(d) - инициализация входа в отладчик ЛОСа. d - 1 либо 2 - указывает значение регистра blokprer, которое должно быть установлено после выхода из отладчика.

52. confrag(a, b) - a, b суть указатель на начало фрагмента программы в формате блока программ (считанного в некоторый массив) и указатель на свободное место для его преобразования в формат зоны программ. Выполняется это преобразование. При удаче возвращает 1, иначе 0. Ссылки из фрагмента содержат только смещения в файле, без номера этого файла.

53. convchisl(a) - a есть натуральное десятичное число в формате ЛОСа. Возвращает его значение в 32-битном машинном представлении "целое без знака".

54. kodpodobija(a) - a есть смещение в зоне задач начала терма. Возвращает сумму кодов логических символов и переменных в этом терме (код переменной равен 1). Используется при поиске термов в списке, отличающихся друг от друга переобозначением переменных и перестановкой операндов.

55. dlinazoni(a,b) - a есть смещение в зоне задач ячейки, относящейся к некоторому тексту. Если $b=1$, то второй бит второго слова префикса всех достижимых из указанного текста текстов зоны задач изменяется на 1, и возвращается суммарная длина таких текстов (кроме тех, у которых рассматриваемый бит уже был равен 1). Если $b=0$, то все введенные при $b=1$ вспомогательные единицы заменяются нулями.

56. smslovo(a,b,c,n) - a есть номер позиции в буфере текстов, начиная с которой должен быть прочитан словарный фрагмент при анализе текста. b - указатель на начало сегмента внешнего словаря, в котором предпринимается поиск фрагмента. c - номер разряда слова, до которого включительно происходит сравнение слов (изначально равен максимальной длине 24 словарных фрагментов внешнего словаря, а в процессе перечисления фрагментов может принимать меньшие значения). Перед обращением в регистры $B1, B2$ заносятся смещения в нижней части сегмента внешнего словаря начала и конца отрезка поиска (в этой части хранятся 16-битные номера символов, кодирующих фрагменты, упорядоченные в соответствии с лексикографическим упорядочением фрагментов). Функция осуществляет поиск на данном отрезке логического символа, фрагмент которого является началом текста, расположенного в буфере текстов начиная с a. В случае неудачи возвращает 0, иначе - найденный логический символ. При этом $B1, B2$ указывают на новые границы отрезка поиска (если $B1 > B2$, то отрезок пуст). В регистр $B5$ заносится смещение в буфере текстов после считанного слова. При поиске словарного фрагмента используется метод деления отрезка пополам. Так как происходит не идентификация слов, а определение допустимого наложения на текст, то возможен пропуск более короткого слова, которое допускает наложение на текст. Чтобы предотвратить такие пропуски, используются регистры $B3, B4$. В $B3$ заносится смещение верхнего конца отрезка для повторного поиска фрагментов, длина которых ограничена величиной $B4$.

57. sledslovo(a,b) - a есть номер позиции в буфере текстов; b - номер сегмента внешнего словаря, с которого начинается поиск словарных фрагментов. Реализует цикл обращений к оператору smslovo, со сменой номера сегмента и с откатами для регистров $B3, B4$ при поиске более короткого наложения - до обнаружения первого допустимого фрагмента. Возвращает найденный логический символ либо (при неудаче) 0. Перед обращением в регистры $B1, B2$ заносятся смещения в нижней части текущего сегмента внешнего словаря начала и конца отрезка поиска; в регистр $B7$ - номер разряда слова, до которого включительно происходит сравнение слов.

58. kluchslova(a,b,c) - a,b суть начало и конец в буфере текстов ключевой последовательности K; c - номер сегмента внешнего словаря. Возвращает смещение в этом сегменте, по которому расположен логический символ, обозначающий первое слово с заданным началом K. Если таких слов нет, возвращает значение $0xFFFFFFFF$.

59. shagslova() - в массиве teksl[ ] хранятся исходные смещения по всем сегментам внешнего словаря. Предпринимается поиск лексикографически наименьшего словарного фрагмента начиная с этих смещений. Возвращает логический символ для этого фрагмента; при неудаче возвращает 0. Корректирует teksl[ ] для следующего шага.

16.5.3 Функции, используемые для перевода программы из формата зоны задач в формат зоны программ и обратно

1. konvblock(a, b, c) - преобразование подтерма оператора ЛОСа из формата зоны задач в формат зоны программ (если заголовок этого подтерма - непосредственно реализуемый оператор либо операторное выражение); результат накапливается в массиве R. a есть смещение начала текста обрабатываемого оператора в зоне задач; b - смещение вхождения в этот оператор текущего подтерма; c - смещение первой свободной ячейки в R. Используется массив ссылок на функции konv[ ], в котором содержатся выполняющие указанное преобразование процедуры для конкретных заголовков операторов и операторных выражений. Возвращает смещение первой свободной ячейки в R; при неудаче возвращает 0.

2. konvoper(a, b, c) - входные данные и преобразования - те же, что для konvblock, но для общего случая подтерма оператора (без условия непосредственной реализуемости).

3. konvert(a) - a есть смещение в зоне задач набора, представляющего фрагмент программы. В массиве R создается текст этого фрагмента в формате зоны программ. Ссылки из фрагмента - двух типов: либо 2-13 биты равны 0, и тогда младшие 19 бит суть смещение в файле блока программ, либо эти биты равны 1, и тогда младшие 16 бит - номер логического символа, являющегося меткой недоопределенного перехода. Префикс фрагмента размещается в R начиная с нулевой ячейки (заполняется частично).

4. konvperechisl(a, b, c, f) - общая часть функций массива konv[ ], обрабатывающих перечисляющие операторы. f - номер F-функции интерпретатора для процедуры, реализующей шаг продолжения перечисления.

5. konvssilka(a) - a есть смещение в зоне задач вхождения символа "ветвь" либо "иначе" для оператора перехода в фрагменте программы. Формирует в массиве F (где до перенесения в массив R накапливаются ссылки из фрагмента) очередную ссылку для этого перехода и корректирует регистр $B1$, указывающий на первую свободную ячейку в F.

6. fragfile() - в массиве R находится фрагмент программы, сформированный оператором konvert. Осуществляет преобразование этого фрагмента в формат блока программ и формирует результат в массиве F. Возвращает число байт, занятых данным результатом.

7. dekop(a, b, p) - шаг преобразования программы из формата зоны программ в формат зоны задач. a - указатель на первую ячейку фрагмента программы в зоне программ; b - смещение в этом фрагменте текущей ячейки, начиная с которой и до ячейки со смещением p (не включая последнюю) следует выполнить преобразование в формат зоны задач. Результирующие операторы считывает в массив R начиная с ячейки $B1$. Функция использует массив указателей на функции dekonv[ ], в котором содержатся процедуры для преобразования конкретных операторов и операторных выражений.

8. dekonvert(a) - a есть указатель на первую ячейку фрагмента программы в зоне программ. Этот фрагмент преобразуется в формат зоны задач и возвращается смещение в зоне задач списка операторов фрагмента. Корректируются регистры $FR1$, $FR2$, ссылающиеся на фрагмент в зоне программ и его версию в зоне задач. В массиве FR создается переходник: на позициях, соответствующих обращениям к процедурам интерпретатора в фрагменте программы из зоны программ, указываются смещения в зоне задач вхождений операторов и операторных выражений, реализуемых этими процедурами. При неудаче возвращает 0.

16.5.4 Отладочные функции

Несколько функций интерпретатора используются при отладке с помощью отладчика СИ.

1. pp() - создает в массиве $nn[\,]$ последовательность номеров переходов от корня программы символа symbol к текущему выполняемому ее фрагменту. Номера расположеы в обратном порядке (первый номер перехода - в конце последовательности).

2. contprog(a) - a есть номер файла блока программ (4x3 бит в младших разрядах). Предпринимается проверка корректности этого файла. При обнаружении ошибок в массиве CONTR перечисляются пары: тип ошибки - смещение в файле (либо в массиве $U1$, куда считан файл) места расположения ошибки. Возвращает смещение первой свободной ячейки в CONTR. Типы ошибок:

   1 - отсутствие внешней ссылки на фрагмент;

   2 - более одной ссылки на фрагмент, и тогда вместо пары в CONTR вводится тройка - в конце пары добавляется число ссылок на фрагмент;

   3 - фрагмент не преобразуется в формат зоны задач;

   4 - более 30 ссылок из фрагмента;

   5 - более 10000 байт в фрагменте;

   6 - ссылка из другого фрагмента оказалась внутри фрагмента, и тогда вместо пары в CONTR вводится тройка - в конце пары добавляется смещение места внутри фрагмента, на которое имеется внешняя ссылка;

   7 - ссылка на объект вне текущего файла; тогда в CONTR к концу пары добавляется смещение такой ссылки из фрагмента;

   8 - ссылка на фрагмент, занесенный в категорию "мусора";

   9 - неправильная обратная ссылка.

3. continf(a, b) - a есть номер информационного блока; b - номер его файла (4x3 бит в младших разрядах). Выполняет контроль корректности данного файла. Информация об ошибках сохраняется в массиве CONTR. Типы ошибок: 1 - нет внешней ссылки на объект; 2 - число ссылок на объект не равно числу его закреплений; 3 - ссылка не на начало объекта.

16.6 Отладка логической системы на уровне интерпретатора ЛОСа

Обычно для отладки оказывается достаточно средств, предоставляемых отладчиком ЛОСа. Однако, иногда может возникнуть необходимость в отладке на уровне интерпретатора ЛОСа - с помощью отладчика СИ. Например, это бывает необходимо при вводе новых операторов ЛОСа, реализуемых интерпретатором. Не исключена возможность обнаружения ошибки и в программах старых операторов. Наконец, ошибки в интерпретаторе может и не быть, а причиной неверного выполнения базисного оператора ЛОСа оказаться какая-либо ошибка в ЛОС-программе, прошедшая незамеченной в момент ее появления, и лишь косвенным образом (например, через изменение каких-либо регистров или массивов) сказавшаяся позднее.

Так или иначе, перед обращением к отладчику СИ необходимо локализовать момент неверного выполнения интерпретатором конкретного оператора или операторного выражения ЛОСа. Обычно для этого хватает средств отладчика ЛОСа (например, локализация момента ошибки путем деления отрезка пополам - по счетчику шагов интерпретатора). Чтобы перейти к работе с отладчиком СИ, далее следует вставить перед оператором, выполняемым с ошибкой, вспомогательный оператор "тест(0)". Этот оператор не выполняет никаких действий, но обращение к нему легко отследить из отладчика СИ. Иногда ошибочно выполняемый оператор может применяться многократно до момента ошибки. В этом случае нужно сформулировать на ЛОСе какое-либо условие P, истинное только перед ошибочным обращением (например, посмотреть входные данные оператора перед ошибочным обращением и взять в качестве P оператор, проверяющий их совпадение с текущими), и вместо оператора "тест(0)" вставить оператор "или(не(P) тест(0))". Наконец, можно обойтись и без вставки оператора "тест", войдя в отладчик непосредственно перед ошибочным выполнением оператора и нажав Ctr-F3 - это вызовет обращение к оператору "тест(1)" внутри программы отладчика, причем после выполнения данного оператора произойдет немедленный выход из отладчика.

Для обработки ошибки отладчиком СИ нужно (решив одним из указанных выше способов вопрос о локализации ошибки путем обращения к оператору "тест" непосредственно перед ее появлением) запустить программу логической системы из этого отладчика с прерыванием при обращении к функции $F214$ (она реализует оператор "тест"). После прерывания в отладчике СИ нужно либо установить прерывание при входе в программу Fi исполняемого с ошибкой оператора (предварительно установив номер i логического символа - заголовка этого оператора), либо дойти до начала этой программы пошаговой трассировкой. В обоих случаях далее - анализировать выполнение программы по шагам.

Заметим, что обычно при отладке интерпретатора наибольший интерес представляет участок программы рабочего цикла interp, предшествующий оператору inter[(a & T4)-1](). Здесь расположен оператор if(step == Step || symbol == Symbol || fragm == Fragm || Cadr == cadr) {pp();}, позволяющий делать прерывания при достижении регистрами step, symbol, fragm, cadr заданных значений, устанавливаемых в начале СИ-трассировки в регистры Step, Symbol, Fragm, Cadr. После прерывания можно посмотреть в массиве nn номера переходов к текущему фрагменту ЛОС-программы символа symbol от корня этой программы (взятые в обратном порядке). Еще один аналогичный оператор расположен после обращения к inter[...] - с его помощью можно устанавливать прерывания по обнаружении заданного значения в заданной точке

зоны задач, глобального стэка либо зоны программ. Наконец, еще дальше идут (занесенные в комментарии) фрагменты, содержащие обращения к функциям contprog и continf для контроля корректности файлов блока программ и информационных блоков.

Глава 17

Программы просмотра списков задач

Фактически в разделе описываются две процедуры - "списокзадач" и "цепьзадач", обеспечивающие просмотр задач, редактирование, запуск и пошаговый просмотр решения. Приводятся достаточно подробные описания программ, реализующих перечисленные операции, так как при дальнейшем развитии системы неоднократно может возникать необходимость модифицировать эти программы.

Задачник логической системы организован в виде оглавления, концевые пункты которого соответствуют отдельным задачам. Концевые меню задачника определяют линейные цепочки задач, причем для просмотра и редактирования всей цепочки используется отдельная процедура - "списокзадач". Эта же процедура позволяет запускать процесс решения задачи.

Если решение задачи выполняется с пошаговым показом преобразований (так называемая семантическая трассировка), то на каждом шаге, кроме информации о текущем действии решателя, для просмотра предлагается также вся текущая цепь задач. Последняя задача этой цепи (она размещена непосредственно перед информацией о текущем действии) - та задача, к которой, собственно, и относится текущее действие. Предпоследняя задача - та, для решения которой некоторый прием обратился к последней задачи, и т.д., вплоть до задачи, извлеченной из задачника, быть может, измененной в процессе решения. Исходный вариант этой задачи (в задачнике он сохраняется без изменений) в просматриваемый список задач не включается.

После информации о текущем действии могут быть размещены несколько вспомогательных задач, которые были решены в процессе выполнения данного действия. Таким образом, при просмотре текущего шага решения обеспечивается просмотр некоторой последовательности задач и кадров вспомогательной информации (которые для удобства рассматриваются как обобщенные задачи). Этот просмотр аналогичен просмотру списка задач из задачника, хотя и реализован другим оператором - процедурой "цепьзадач". Обращается к процедуре "цепьзадач" отладчик ЛОСа, и при описании ее будет разобрана ветвь отладчика, извлекающая из анализа глобального стэка необходимые для обращения данные.

17.1 Просмотр списка задач из задачника

Как уже было отмечено в подразделе "Обращение к оглавлению задачника"раздела, посвященного главному меню системы, для просмотр списка задач концевого меню задачника используется процедура "списокзадач(x1 x2 x3)". Она получает в качестве входной информации: набор x1 меток пути в оглавлении задачника к указателю-

списку x2, подпункты которого ссылаются на задачи списка, а также логический символ x3, по которому от x2 имеется переход к текущей задаче списка.

Прежде, чем перейти к программе оператора "списокзадач", приведем описание основных структур данных задачника. Оглавление задачника хранится во 2-м информационном блоке. Концевой логический терминал этого оглавления содержит терм "задача(A_1 A_2)", ссылающийся на узел задачи (терм такого вида в терминале один). Здесь A_2 - номер узла в статье логического символа A_1.

Для указания на режим серийного решения задач используется логический терминал, достижимый из корневого указателя-каталога 2-го информационного блока по метке "метка". Этот терминал содержит терм "задачи(R)". Если $R = 0$, то серийный режим решения задач отсутствует. Если $R = 1$, то имеет место режим серийного решения задач из заданного раздела задачника. Если $R = 2$, то при серийном решении дополнительно предпринимается анализ холостого хода приемов - накапливается статистика о трудоемкости неудачных попыток применения приема и о числе его применений. Если $R = 3$, то имеет место режим серийного решения лишь тех задач выбранного раздела задачника, у которых по ходу решения встречался хотя бы один из логических символов заданного списка $S_1, \ldots, S_n$. Данные о символах, встречавшихся при решении задачи, сохраняются в 10-м информационном блоке - архиве задачника. Список $S_1, \ldots, S_n$ содержится в логическом терминале, достижимом из корня 2-го информационного блока по метке "не". Случай $R = 5$ - вспомогательный; он возникает после получения ответа задачи и служит для вос-становления просмотра задачи в задачнике. Одновременно в логический терминал, достижимый из корня 2-го информационного блока по меткам "сброс", "время" передается информация о времени решения (минуты и секунды) для прорисовки ее после текста задачи.

Из узла задачи могут иметься переходы по следующим меткам:

1. "задача" - переход к указателю-списку, представляющему задачу. По метке "цели" от него - переход к логическому терминалу, содержащему логические символы и термы $A_0, \ldots, A_n$. Здесь A_0 - тип задачи; $A_1, \ldots, A_n$ - коды ее целей. По метке "посылка" - переход к группе логических терминалов, каждый из которых хранит посылку задачи. После посылки могут быть размещены комментарии к ней (те комментарии, которые суть наборы, преобразуются в формат термов). Аналогичным образом, по метке "условие" - переход к группе терминалов, каждый из которых хранит условие задачи. Если задача сопровождается чертежом, то по метке "точка" - переход к логическому терминалу, хранящему описание чертежа. Формат этого описания представлен в справочной информации логического символа "точка". По метке "комментарии" (если она есть) - переход к логическому терминалу, перечисляющему комментарии к задаче.

2. "оглавление" - переход к логическому терминалу, содержащему термы "набор($A_1 \ldots A_n$)", определяющие последовательности $A_1, \ldots, A_n$ меток перехода от корня оглавления задачника к концевым пунктам, ссылающимся на данную задачу.

3. "ответ" - переход к логическому терминалу, хранящему ответ либо отказ на задачу узла. Если после того, как система находила ответ на задачу, она перестала ее решать, то перед старой версией ответа помещается логический символ "отказ". В случае задачи на текстовый анализ по метке "ответ" - переход к

текстформульной ветви (см. описание текстформульного редактора), хранящей текст ответа.

4. "время" - переход к логическому терминалу, хранящему термы "время(A_1)", "вычитание(A_2)", "минимум(A_3)", где A_1 - число шагов, затрачиваемых системой на решение задачи; A_2 - приращение данного числа шагов в последнем цикле решения задачи; A_3 - наименьшее из зарегистрированных в процессе изменений системы значений A_1 для данной задачи.

5. "повторение" - переход к логическому терминалу, хранящему десятичную запись номера шага, на котором предпринимается прерывание при повторном решении задачи (используется при отладке).

6. "текстформ" - переход к текстформульной ветви, хранящей текст задачи. Используется для представления задачи на анализ текста, причем в этом случае переход по метке "задача" отсутствует.

7. "блок" - переход к логическому терминалу, хранящему ссылки "прием(A_1 A_2 A_3)" на приемы, применение которых при решении задачи блокируется. A_1 - логический символ; A_2 - номер узла этого символа; A_3 - заголовок приема. Заготовка отладчика процедур синтеза приемов, используемых в процессе решения задачи.

8. "битмэп" - переход к логическому терминалу, хранящему ссылку на рисунок. Используется для представления задачи на анализ рисунка, причем в этом случае переход по метке "задача" отсутствует. Битмэпы рисунков хранятся не во 2-м информационном блоке задачника, а в специально выделенном для них 13-м информационном блоке. Ссылкой на рисунок является терм "битмэп(A_1 A_2 A_3)". Здесь A_1 - логический символ, по которому из корневого указателя-каталога 13-го информационного блока имеется переход к указателю-списку, из которого по метке "битмэп" - переход к текстовому терминалу, хранящему битмэп рисунка. A_2 - ширина рисунка в 16-битных словах; A_3 - число 16-битных слов текстового терминала, загружаемых в буфер битмэпов при считывании рисунка из информационного блока.

9. "структура" - переход к указателю-списку, ветвь которого определяет задачу на моделирование процесса. В этой ветви выделяются узлы объектов (корень ветви сам является узлом объекта). От каждого узла объекта по метке "элемент(t)", где t - обозначение элемента либо серии однотипных элементов, имеется переход к узлу объекта, представляющему элемент (серию элементов) t. В случае серии выражение t имеет параметры, определяющие конкретный элемент серии. По метке "условие" из узла объекта - переход к группе логических терминалов, содержащих утверждения логического описания структуры (элементы и связи между ними) и параметров состояния объекта. Переменные в описаниях объектов - локальные; из внешнего контекста переносятся, если они есть, только параметры серии элементов. Если узел объекта представляет серию однотипных объектов, то по метке "параметры" от него - переход к логическому терминалу, хранящему условия на параметры, определяющие элемент серии. В случае задачи на моделирование процесса переход по метке "задача" отсутствует.

Начало программы символа "списокзадач" можно найти, например, через оглавление программ - пункт "Интерфейс просмотра и редактирования задач" - "процедура СПИСОКЗАДАЧ" - "Исходная точка".

После оператора "повторение", к которому будут происходить откаты при повторном входе в обработку списка задач (например, если он был изменен), инициализируется накопитель x4 троек, описывающих задачи списка и их прорисовку на экране. Каждая тройка (A_1 A_2 A_3) этого накопителя состоит из: логического символа A_1, являющегося меткой перехода к концевому терминалу оглавления задачника, ссылающемуся на задачу, задачи A_2, и набора A_3 информационных элементов для последовательно прорисовываемых фрагментов изображения задачи. Эти фрагменты далее называем полосами изображения. В наборе A_3 используются информационные элементы следующих типов:

1. (терм B_1 B_2). B_1 - очередной терм задачи (посылка либо условие); B_2 - либо 0 (в отсутствии прорисовки), либо пара (C_1 C_2) (в случае прорисовки текстовым редактором), где C_1 - исходная строка для прорисовки; C_2 - высота полосы прорисовки, либо тройка (C_1 C_2 C_3) (в случае прорисовки формульным редактором), где C_1, C_2 - те же, C_3 - корень дерева указателей формульного редактора для прорисовки терма.

2. (отрезок B) - указатель на горизонтальную линию, с которой начинается прорисовка задачи. B - либо 0, либо символьный номер строки для проведения горизонтальной линии.

3. (слово $B1$ B_2) - ссылка на текстовый терминал, достижимый из корневого указателя 2-го информационного блока по меткам B_1, "слово". B_2 - либо 0, либо пара (C_1 C_2), где C_1 - исходная строка для прорисовки текста, C_2 - высота полосы прорисовки.

4. (набор B_1 B_2) - указатель прорисовки последовательно расположенных в общей полосе текстовых фрагментов, термов и букв. B_1 - набор троек одного из следующих типов:

   а) (слово C_1 C_2) - ссылка на текстовый терминал, достижимый из корня 2-го информационного блока по меткам C_1, "слово". C_2 - либо 0, либо пара (столбец - строка) для начала прорисовки текста.

   б) (буква C_1 C_2) - C_1 есть логический символ, кодирующий прорисовываемую букву; C_2 - такое же, как и выше.

   в) (терм C_1C_2) - C_1 есть терм либо логический символ; C_2 - либо 0, либо пара (столбец - строка) для начала прорисовки терма текстовым редактором, либо тройка (столбец - строка - корень дерева указателей формульного редактора для прорисовки терма).

   B_2 - либо 0, либо пара (исходная строка полосы - высота полосы).

5. (геомредактор B_1 B_2) - ссылка на чертеж. B_1 есть тройка (описание чертежа в формате оператора "геомредактор" - верхняя строка рамки - нижняя строка рамки). B_2 - либо 0, либо пара (C_1 C_2), где C_1 - исходная строка для прорисовки чертежа; C_2 - высота полосы чертежа. До прорисовки чертежа в списке задач последние два элемента тройки B_1 могут отличаться от значений $C_1, C_1 + C_2$, а затем становятся им равны.

6. (пусто B). Указатель на пропуск горизонтальной полосы высотой в 19 пикселей. B - либо 0, либо набор (C), где C - номер строки, с которой начинается полоса.

7. (текстформ B_1 B_2). B_1 - текстформульная структура (см. описание текстформульного редактора); B_2 - либо 0, либо пара (исходная строка для прорисовки текстформульного элемента - высота тксткформульной полосы).

8. (шахматы B_1 B_2). B_1 - задача на исследование, описывающая текущую шахматную позицию; B_2 - либо 0, либо пара (исходная строка для прорисовки шахматной позиции - высота полоы прорисовки).

9. (битмэп B_1 B_2). B_1 - тройка (C_1 C_2 C_3, представляющая собой ссылку на рисунок в 13-м информационном блоке. C_1 - логический символ, такой, что по меткам C_1, "битмэп" от корня указанного блока - переходы к текстовому терминалу, хранящему битмэп. C_2 - ширина рисунка в 16-пиксельных единицах. C_3 - число 16-битных слов в битмэпе. B_2 - либо 0, либо пара (исходная строка рисунка - высота полосы рисунка).

После инициализации набора x4 программа "списокзадач" обращается к оператору "извлечениезадачи", который определяет в указанном выше формате тройку x5 для текущей задачи, после чего эта тройка заносится в x4. Основную часть работы по созданию информационных элементов полос изображения при этом выполняет оператор "элементызадачи", который будет описан в разделе, посвященном пошаговому просмотру решения (см. подраздел "Создание набора информационных элементов полос изображения обобщенной задачи"). Далее - переход через "ветвь 2".

Здесь, после проверки непустоты набора x4 (если он пуст, то - внутренний перезапуск), располагается оператор "повторение", к которому будут происходить откаты для перерисовки. Если нет комментария (транскоммент ...) - при его наличии перерисовка не нужна, так как сразу запускается решение текущей задачи, - то происходит расчистка экрана. Далее инициализируются переменные: x6 - вхождение в x4 текущей прорисовываемой задачи (изначально - левый край набора x4); x7 - номер первой свободной строки на экране; x8 - набор информационных элементов для прорисовки полос текущей задачи; x9 - сначала 0, а затем вхождение в x8 первого непрорисованного элемента. Просматриваются все вхождения x10 в x4, начиная с вхождения x6. Для текущего вхождения x10 определяется список x11 информационных элементов прорисовки полос соответствующей задачи. Просматриваются вхождения x12 в список x11 элементов x13. Если последний элемент набора x13 не равен 0, то он содержит всю необходимую для прорисовки полосы на экране, и тогда оператор "смполоса(...)" фактически выполняет данную прорисовку. При этом переменной x6 присваивается значение x10, переменной x9 - значение x12, переменной x7 - номер строки, расположенной после прорисованной полосы. Таким образом, реализуется цикл повторной прорисовки на расчищенном экране тех элементов, для которых информация о прорисовке сохранилась после выполнения предшествовавшей перерисовке операции. По завершении этого цикла - переход через "ветвь 2".

Если исходная задача имеет комментарий "элементызадачи", то вход в процедуру "списокзадач" произошел в результате внутреннего перезапуска после получения ответа задачи. В этой ситуации после прорисовки текущей задачи нужно указать время (в минутах и секундах) решения задачи. Комментарий "элементызадачи" удаляется, и проверяется наличие комментария "транзитпереход". Он возникает, если решалась задача на исследование, и прорисовка времени не нужна, так как такие задачи в задачнике - только фиктивные (например, используемые для представления исходной

игровой позиции). Если указанного комментария нет, то количества затраченных на решение минут и секунд извлекаются из логического терминала, достижимого из корневого указателя 2-го информационного блока по меткам "сброс","время". После последней полосы первой задачи набора x4 (она и будет только что решавшейся задачей) вставляются элементы, необходимые для прорисовки текста "Время решения: ... мин. ... сек.". Затем - переход через "ветвь 3", где происходит такая переустановка значения x9, чтобы вместо прорисовки всей задачи была прорисована лишь ее часть, идущая после слов "Ответ:" либо "Ответ утерян" (эти текстовые заготовки хранятся в текстовых терминалах 2-го информационного блока, достижимых из корня, соответственно, по меткам "суммавсех","слово" и "выписка","слово").

Далее - цикл дорисовки экрана до его заполнения оставшимися элементами задачи, прорисовка которой уже начата, и последующими задачами. Переход к нему - через оператор "ветвь 1", предшествующий оператору "исходнаязадача(x10)".

Цикл дорисовки начинается с оператора "повторение", откаты к которому будут выполняться после прорисовки очередной полосы. Затем идет оператор, заменяющий значение переменной x8 на набор информационных элементов, определяющих прорисовку полос изображения задачи, соответствующей вхождению x6 в набор x4. Если x9 равно 0 (прорисовка полос задачи еще не начиналась), причем набор x8 непуст, то x9 заменяется на вхождение первого разряда набора x8. Если x9 не равно 0, причем соответствующая x9 полоса уже прорисована, то находится следующее после x9 вхождение x10 в набор x8. Если x10 не равно x9, то x9 заменяется на x10, иначе - находится следующее после x6 вхождение x11. Если такого вхождения не было (т.е. x11 оказалось равно x6), то из задачника считывается задача, следующая за последней считанной в набор x4, и тройка x12, представляющая ее, добавляется в конец набора x4. x6 становится равно вхождению в x4 только что добавленной тройки, а x9 - вхождению первого из элементов прорисовки полос считанной задачи. Если же x11 было отлично от x6, то x6 присваивается x11, а x9 - вхождение первого из элементов прорисовки полос задачи по x11. Во всех перечисленных выше случаях далее - переход через "ветвь 1".

Проверяется отсутствие комментария (транскоммент ...), а также равенство нулю последнего разряда информационного элемента по вхождению x9. Предпринимается обращение к оператору "высотаполосы", который изменяет последний разряд этого информационного элемента на данные, необходимые для прорисовки полосы, и определяет высоту полосы x10. Если места для прорисовки полосы достаточно, то оператор "смполоса" выполняет эту прорисовку, корректируется номер x7 первой свободной строки, и откат к оператору "повторение" для прорисовки следующей полосы. Если же места для прорисовки полосы недостаточно, то последний разряд элемента по x9 снова заменяется на 0, и цикл дорисовки завершается (переход через "ветвь 1").

Если x9 не равно 0 и вхождение x6 в набор x4 - не последнее, то отбрасываются все идущие после него разряды, причем значением x6 становится вхождение последнего разряда набора x4. Далее - переход через "ветвь 1", и рассматривается первый элемент x10 набора x4. Если он отличен от последнего элемента и ни одна его полоса не прорисована, то отбрасывается, причем эта операция повторяется в цикле, пока набор x4 не окажется одноэлементным либо не будет прорисована хотя бы одна полоса его первой задачи.

После такой расчистки набора x4 - переход через "ветвь 1" к фрагменту, в котором начинается обработка команд просмотра списка задач. Альтернативный выход к данному фрагменту - через пункт "Цикл обращений к клавиатуре - Исходная точ-

ка" ветви процедуры "списокзадач" в оглавлении программ. Фрагмент начинается с оператора "повторение" (откаты к получению очередных команд), после которого идет оператор "автоменю(x10)", запрашивающий очередную команду x10. Если эта команда отлична от "тринадцать", "обл","мышь" (клавиши Enter, пробел и нажатие кнопки мыши), то перед обработкой команды предпринимается обращение к процедуре "коррекциязадач(0)". Эта процедура анализирует комментарий (коррекциязадач ...), в котором накапливаются указания на изменения задач, выполненные предыдущей командой и еще не зарегистрированные в файлах задачника, и регистрирует такие изменения. Далее идет последовательность фрагментов обработчика команд:

1. x10 = "Плюс" либо x10 = "циклвариантов". Соответственно, нажаты клавиши Esc либо "курсор влево". В этом случае происходит выход из программы "списокзадач" для возвращения в процедуру просмотра оглавления задачника. Так как в процессе прокрутки списка задач текущей задачей может оказаться не та задача, через которую первоначально произошел вход в просмотр списка, то перед выходом предпринимается переустановка оглавления задачника на новую "текущую" задачу. Ее роль теперь играет первая из задач, для которых на экране прорисована верхняя отделяющая горизонтальная линия. Для переустановки рассматриваются первая тройка x11 набора x4 и следующая за ней тройка x13. Если первый элемент полосы задачи, соответствующей тройке x11, не прорисован (условие равенства нулю конца первого информационного элемента списка x14), а первый элемент полосы задачи x13 - прорисован (отличие от нуля конца первого информационного элемента списка x15), то метка x12 перехода к текущей задаче берется из x13, иначе - из x11. Затем - переход через "ветвь 3" к оператору "путьвоглавлении", выполняющему переустановку текущего пути в оглавлении задачника.

2. x10 = "подстановка". Нажата клавиша "курсор вниз" для прокрутки списка задач вверх на минимально возможный шаг. При сдвиге в верхней части экрана убирается ровно одна полоса. Если снизу после этого оказывается достаточно места, и список задач не завершен, то добавляется некоторое количество новых полос.

   Прежде всего, проверяется, не осталась ли на экране единственная полоса, завершающая прорисовку всего списка задач. В этом случае сдвиг вверх блокируется. Иначе - переход через "ветвь 2".

   Здесь инициализируется нулем переменная x11 - накопитель величины сдвига вверх в пикселах. Просматривается список x4, и для текущей тройки x13 этого списка (x12 - текущее вхождение ее в x4) просматриваются информационные элементы x16 прорисовки полос. Находится первый такой элемент, у которого конец ненулевой - он определяет первую полосу изображения. x11 увеличивается на высоту этой полосы (используются символьные номера). Если элемент определял прорисовку разделяющей горизонтальной линии либо вертикального 19-пиксельного пробела, то просмотр и увеличение значения x11 продолжается. Иначе - откат к оператору "ветвь 1", перед которым проверяется, не оказалась ли первая задача списка x4 в результате прокрутки полностью устраненной с экрана; в этом случае она исключается из списка x4.

   Если величина x11 оказалась не равной 0, то предпринимается просмотр всех определяющих прорисовку полос информационных элементов и коррекция их

параметров, соответствующая сдвигу изображения на x11 пикселей вверх. Затем - прокрутка изображения вверх на x11 пикселей, коррекция номера x7 первой свободной строки, и откат к дорисовке.

3. x10 = "внешсумма". Нажата клавиша "курсор вверх" для прокрутки списка задач вниз на минимально возможный шаг. При этом сверху оказывается прорисована ровно одна дополнительная полоса изображения (не считая полос горизонтальной разделяющей линии и 19-пиксельного вертикального пробела), а снизу может быть удалено с экрана сразу несколько полос. Каждая полоса удаляется целиком, и каких-либо ее частей на экране не остается.

   Прежде всего, инициализируются значения переменных x11, x12, x13, x14. В x11 будут накапливаться пары (вхождение информационного элемента для прорисовки полосы, добавляемой при сдвиге вниз в верхней части экрана - номер метки перехода к задаче, в изображение которой входит данная полоса). x12 - накопитель величины сдвига вниз (символьный номер); x13 - вхождение в набор x4 тройки для той задачи, к которой относится первая полоса изображения до сдвига вниз; x14 - вхождение информационного элемента для прорисовки этой полосы. После инициализации предпринимается просмотр набора x4 и определение указанных значений x13, x14. Далее - переход через "ветвь 2".

   Выполняется цикл просмотра информационных элементов полос изображения, предшествующих элементу по x14, и заполнение набора x11 до тех пор, пока в нем не окажется ровно одна ссылка на информационный элемент для прорисовки полосы изображения, отличной от горизонтальной разделяющей линии и вертикального 19-пиксельного пробела. Если при этом просмотре оказывается, что в задаче по x13 такой полосы нет, то переход через "иначе 2", где из задачника извлекается предыдущая задача списка, и ее тройка x19 заносится в начало списка x4. По окончании цикла - переход через "ветвь 1".

   Если накопитель x11 оказался пустым (на экране уже было прорисовано начало списка задач), то откат к запросу очередной команды. Иначе просматриваются информационные элементы полос изображения, прорисованных на экране до сдвига вниз, и их параметры корректируются для сдвига вниз на x12 пикселей. Затем - переход через "иначе 1".

   Изображение сдвигается вниз на x12 пикселей. Последовательно просматриваются пары x16 из накопителя x11; для каждой такой пары оператор "высотаполосы" подготавливает ее информационный элемент полосы изображения для прорисовки, после чего оператор "смполоса" выполняет прорисовку. Далее - переход через "ветвь 1".

   Находится информационный элемент x19 последней прорисованной на экране полосы изображения и вычисляется номер x21 строки, следующей после полосы этого элемента. x7 присваивается значение x21; если строка x21 расположена выше нижнего края экрана, то экран под ней расчищается. Значения x6 и x9 корректируются так, чтобы они указывали на задачу, с которой будет нужно продолжать прорисовку, и на вхождение того информационного элемента полосы, с которого ее нужно продолжать. Если в наборе x4 имеются тройки, следующие после вхождения x6, то они удаляются. Затем - откат к запросу следующей команды.

4. x10 = "точкапривязки". Нажата клавиша PageDown для перехода к следующей

странице изображения списка задач. Если вхождение x9 - последнее в списке информационных элементов полос изображения последней прорисованной на экране задачи, то считывается тройка x11 для прорисовки следующей задачи, и x4 заменяется на одноэлементный набор, состоящий из x11. x6 и x9 корректируются для начала прорисовки новой задачи, экран расчищается, и откат к дорисовке. Если же вхождение x9 - не последнее, то в наборе x4 оставляется только его последний элемент, корректируется x6, и последние разряды всех информационных элементов полос изображения оставшейся задачи обнуливаются (так как эти элементы уже не являются прорисованными). Значение x9 сохраняется, так как дорисовка начнется с первого непрорисованного ранее элемента. Далее расчищается экран, указатель первой свободной строки x7 устанавливается на верхнюю строку экрана, и откат к дорисовке.

5. x10 = "контрольунификации". Нажата клавиша Page Up для перехода к предыдущей странице изображения списка задач. Последовательно просматриваются тройки x12 набора x4 и информационные элементы полос изображения x14 в этих тройках - пока не оказывается найден первый прорисованный на экране элемент x14. Инициализируется нулем счетчик x15 суммарной высоты полос изображения, которые будут выданы на предыдущей странице. Переменным x16 и x17 присваиваются значения x11, x13 - вхождения тройки x12 и информационного элемента x14 в свои списки. Эти переменные будут указывать на текущий анализируемый информационный элемент полосы изображения при перечислении элементов, которые будут составлять предыдущую страницу. Далее, начиная с оператора "повторение", идет цикл перечисления. Если вхождение x17 - не первое в своем списке, находится предшествующее ему вхождение x18. Определяется высота x19 полосы элемента по вхождению x18, и x15 увеличивается на x19. Если x15 оказывается меньше, чем x5, то-есть при добавлении полосы к предыдущей странице эта страница помещается на экране, то x17 присваивается значение x18, и продолжение цикла; иначе цикл обрывается. По завершении просмотра всех x17, относящихся к тройке по вхождению x16, проверяется, является ли x16 началом набора x4. Если нет, то находится вхождение x19, предшествующее x16, и переменной x20 присваивается вхождение последнего информационного элемента полосы изображения тройки по x19. Определяется высота x21 полосы элемента x20, и x15 увеличивается на x21. Если результат оказывается меньше x5, то x16 заменяется на x19, x17 - на x20, и продолжение цикла; иначе - обрыв цикла. Наконец, в случае, когда x16 было началом списка x4, находится тройка x21, представляющая задачу, предшествующую первой задаче в x4. Эта тройка добавляется к началу списка x4, и цикл просмотра информационных элементов полос изображения продолжается с последнего такого элемента тройки x21. По завершении цикла - переход через "ветвь 2".

   Просматриваются все информационные элементы полос изображения текущей страницы, и их последний разряд заменяется на 0. Затем переменным x6, x9 присваиваются значения x16, x17, экран расчищается, и откат к дорисовке.

6. x10 = "мышь". Нажата кнопка мыши для выделения задачи либо ее элемента. Выделение задач и элементов задач является сквозным для всего задачника - оно сохраняется при выходе из одного концевого списка задач и переходе в другой список. Для выделения используется комментарий (смполоса A) к

списку посылок исходной задачи. Каждый информационный элемент набора A выделяет задач у либо элемент задачи; в этом наборе используются информационные элементы следующих типов:

а) (B_1 B_2). Выделение целой задачи. B_1 - путь в оглавлении задачника к предконцевому указателю-списку; B_2 - метка перехода в этом указателе-списке к логическому терминалу, содержащему ссылку на задачу.

б) (B_1 B_2 B_3 B_4). Выделение целой полосы изображения в задаче. B_1, B_2 определяют ссылку на задачу - как и выше. B_3 - номер вхождения выделенной полосы в список полос. B_4 - информационный элемент данной полосы.

в) (B_1 B_2 B_3 B_4 B_5). $B_1 - B_4$ - те же, что и выше, причем B_4 имеет вид (набор ...). В этом случае полоса состоит из последовательности элементов; выделен один из этих элементов, и B_5 есть номер вхождения данного элемента.

г) (B_1 B_2 B_3 B_4 B_5). $B_1 - B_4$ - те же, что и выше, причем B_4 имеет вид (терм ...). В этом случае полоса образована изображением терма; выделен некоторый его подтерм, и B_5 - указатель вхождения выделенного подтерма в терм. Указателем вхождения одного терма в другой здесь и далее называется терм "фикс($i_1 \ldots i_n$)", где $i_1, \ldots, i_n$ - номера операндов, по которым нужно двигаться от корня терма в направлении к подтерму. Если рассматривается вхождение в кванторную импликацию "длялюбого(x1 ... xn если $A_1 \ldots A_m$ то A_0)", то в качестве i_1 берется номер j того A_j, в котором расположен подтерм.

д) (B_1 B_2 B_3 B_4 B_5 B_6). $B_1 - B_5$ - такие же, как в случае в). В одном из элементов полосы, представляющем собой терм, выделен подтерм. B_6 - указатель вхождения этого подтерма.

Определяются столбец x12 и строка x13 курсора мыши. Если нажата правая кнопка (x11 = 2), то должна быть выделена задача. Для нахождения этой задачи инициализируется нулем переменная x14, и просматриваются тройки x15 набора x4. Если строка полосы изображения для x15, определяющей верхнюю отделяющую линию задачи, расположена не ниже курсора мыши, то x14 заменяется на номер задачи x15 в списке задач. По окончании просмотра значение x14 оказывается равно номеру той задачи, в зоне которой расположен курсор мыши. Затем - переход через "иначе 3".

Создается информационный элемент x15, ссылающийся на выделенную задачу в формате комментария (смполоса A). Если такой комментарий уже был, причем элемент x15 не входил в A, то он заносится в A, иначе - исключается из A. Если же такого комментария не было, то он вводится, с одноэлементным набором A, состоящим из x15. Предпринимается перерисовка всех полос изображения, относящихся к задаче с номером x14 - при этой перерисовке они перекрашиваются в голубой цвет. Далее - откат к получению очередной команды.

Если была нажата левая кнопка мыши (x11 = 0), то должен быть выделен тот элемент задачи, на котором находится курсор мыши. Инициализируются нулями переменные x14, x15, и предпринимается просмотр всех полос изображения, прорисованных на экране и отличных от горизонтальной разделяющей линии либо 19-пиксельного вертикального пробела. Находится последняя из таких полос, для которых курсор мыши расположен не выше ее верхней строки, и переменной x14 присваивается номер задачи, к которой относится полоса, а

переменной x15 - вхождение информационного элемента данной полосы в свой список. Затем - переход через "иначе 1", где переменной x16 присваивается информационный элемент по вхождению x15.

Сначала рассматривается случай, когда x16 имеет заголовок "набор", т.е. определяет последовательность размещаемых друг за другом разнородных элементов изображения. Переменной x17 присваивается список троек, определяющих эти элементы. Просматриваются вхождения x18 в список x17 троек x19, имеющих заголовок "терм". Эти тройки определяют прорисовку терма текстовым либо формульным редактором. Отбирается та тройка, для которой курсор мыши попадает в вертикальную полосу прорисовки терма. Переменной x20 присваивается последний элемент тройки x19 - пара (столбец - строка начала прорисовки терма) в случае текстовой прорисовки либо тройка (столбец - строка - корень дерева указателей формульного редактора) в случае формульной прорисовки. Проверяется, что в случае текстовой прорисовки курсор мыши попадает на 19 - пиксельную горизонтальную полосу текста, а в случае формульной прорисовки - на горизонтальную полосу, в которой располагается формула. Создается информационный элемент x21, ссылающийся на выделенный элемент полосы изображения в формате комментария (смполоса ...). Как и в случае выделения всей задачи, x21 учитывается в таком комментарии (добавляется либо удаляется). Далее происходит перекраска выделенного элемента изображения, и откат к получению очередной команды.

Если x16 не имел заголовка "набор", то создается информационный элемент x17, ссылающийся на выделенную полосу изображения в формате комментария (смполоса ...), который учитывается в таком комментарии. После перекраски полосы - откат к получению очередной команды.

7. x10 = "подборнеизвестных". Нажата клавиша "з" для ввода пустой заготовки новой задачи. Если имеется комментарий (смполоса A), то информационные элементы x15 набора A просматриваются от конца к началу, пока не встретится элемент x15 длины 2. Он соответствует последней выделенной задаче; проверяется, что верхняя горизонтальная линия этой задачи прорисована на экране и что под ней достаточно места для ввода пустой заготовки новой задачи. Если удовлетворяющий таким условиям элемент x15 найден, то новая задача будет вводиться непосредственно перед его задачей.

   Прежде всего, здесь удаляется комментарий (смполоса ...). Затем в цикле просматриваются все задачи текущего списка задач, начиная с той выделенной задачи, переж которой осуществляется вставка. x20 - номер текущей просматриваемой в этом цикле задачи. При просмотре корректируется ссылка из узла задачи на путь в оглавлении задачника, ведущий к ее концевому терминалу - последние элементы этих путей увеличиваются на 1. По завершении цикла - переход через "ветвь 3".

   Здесь увеличиваются на 1 все метки перехода в указателе-списке рассматриваемого списка задач, начиная с метки перехода к выделенной задаче. Затем выбирается тот логический символ x23, в статье которого будет введен новый узел задачи. Для этого используется следующий простой датчик случайных чисел: находится номер x20 текущего шага работы интерпретатора; вычисляется остаток x22 от деления x20 на 1000, и в качестве x23 берется символ с номером x20 + 300. Вводится новый узел статьи x25 символа x23; x26 - номер этого уз-

ла. От этого узла создаются переходы: по метке "задача" - к новому пустому указателю-списку; по метке "оглавление" - к логическому терминалу, в который заносится терм "набор(...)", определяющий путь в оглавлении задачника к концевому логическому терминалу, ссылающемуся на новую задачу. Здесь же вводятся сам этот концевой терминал, а также сопровождающий его текстовый терминал, содержащий текст из трех тире. Затем - переход через "ветвь 2".

Расчищается область экрана, расположенная вниз от верхней горизонтальной линии выделенной задачи (сама эта линия теперь будет рассматриваться как верхняя отделяющая линия новой задачи). Позиция x16 набора x4, соответствующая выделенной задаче, изменяется на тройку, определяющую бланк новой задачи. Этот бланк представляет собой набор (0 пустоеслово пустоеслово набор(пустоеслово)), причем информационных элементов полос изображения новой задачи - два. Первый из них - для горизонтальной отделяющей линии, второй - для вертикального 19-пиксельного пробела. Все тройки набора x4, расположенные после x16, отбрасываются. Корректируются для дорисовки значения x6,x9,x7. Вводится комментарий (смполоса ...), выделяющий новую задачу (так как обычно сразу после ее ввода приходится с ней работать). Корректируется комментарий (списокзадач ...), указывающий путь в оглавлении задачника к изменявшейся в последний раз задаче. После этого - откат к дорисовке.

Если элемент x15, определяющий выделенную задачу, перед которой должна вставляться новая задача, не найден, то переход через "иначе 2" для ввода новой задачи в конце списка задач. Здесь проверяется, что последняя тройка набора x4 действительно определяет последнюю задачу списка и что прорисованы все ее полосы изображения. Кроме того, проверяется, что на экране достаточно места для ввода заготовки новой задачи. После этого переменной x15 присваивается номер новой задачи. Как и выше, с помощью номера текущего шага работы интерпретатора, рассматриваемого как случайное число, определяется тот логический символ x19, в статье которого будет введен узел новой задачи. Вводится этот узел x21; x22 - его номер. Дальнейшие действия аналогичны случаю ввода новой задачи перед выделенной; единственное небольшое отличие - дорисовка после отката начинается не с последнего элемента бланка новой задачи (вертикального пробела), а с первого элемента - еще не прорисованной горизонтальной отделяющей линии.

8. x10 = "обл". Нажата клавиша "пробел" для сброса всех выделений задач и их элементов. Удаляется, если он есть, комментарий (смполоса ...) к посылкам исходной задачи. Согласно этому комментарию, восстанавливается черный цвет прорисовки ранее выделенных задач и элементов задач, находящихся на экране.

9. x10 = "десчастное". Нажата клавиша Ctr-PageUp для перехода к просмотру первой задачи списка задач. Находится первая тройка x11 набора x4. Если она соответствует не первой задаче списка, либо верхняя горизонтальная линия этой задачи не прорисована на экране, то определяется тройка x12 для прорисовки первой задачи списка; x4 заменяется на одноэлементный набор, состоящий из x12, и откат к перерисовке всего экрана.

10. x10 = "отрезок". Нажата клавиша Ctr-PageDown для перехода к просмотру последней задачи списка задач. Находится символьный номер x12 последней

задачи списка задач. Если первая тройка x13 набора x4 соответствует другой задаче, либо верхняя горизонтальная линия последней задачи не прорисована на экране, то определяется тройка x14 для прорисовки последней задачи списка; далее - как в предыдущем случае.

11. x10 = "модули". Нажата клавиша Ctr-Del для удаления выделенной задачи либо выделенного элемента задачи. Находится комментарий (смполоса A) к посылкам исходной задачи, и этот комментарий удаляется. Проверяется, что набор A состоит из единственного элемента x14. Далее рассматриваются следующие случаи:

    а) Длина набора x14 равна 2, т.е. удаляется выделенная задача. Проверяется, что эта задача относится к текущему списку задач (началом пары x14 служит значение x1). Находится номер x15 выделенной задачи в данном списке задач, и в наборе x4 выбирается тройка x16, соответствующая номеру x15. Переменной x17 присваивается список информационных элементов полос изображения удаляемой задачи. Проверяется, что первая из этих полос (верхняя отделяющая линия) прорисована на экране. В концевом логическом терминале оглавления задачника, соответствующем удаляемой задаче, находится терм x21 вида "задача(A_1 A_2)", определяющий логический символ A_1 и номер узла задачи A_2. Находится ссылка x25 на узел задачи. Проверяется, что ссылка на него из оглавления задачника - единственная, после чего этот узел удаляется. Затем просматриваются все концевые логические терминалы рассматриваемого концевого меню задачника, номер x28 перехода к которым больше x15. Корректируются возвратные ссылки на оглавление из узлов задач, соответствующих этим терминалам, так как после удаления задачи номера переходов здесь уменьшатся на единицу. Далее - переход через "ветвь 3".

    Исключаются переходы от определяющего список задач указателя x2 оглавления задачника к логическому и текстовому терминалам удаленной задачи. Метки переходов от x2, номера которых больше x15, уменьшаются на 1. Находится вхождение x28 в набор x4 тройки x16 удаляемой задачи и создается список x29 всех предшествующих x28 элементов набора x4. Расчищается область экрана вниз от верхней отделяющей линии удаленной задачи. Если список x29 непуст, то x4 заменяется на x29; корректируются для дорисовки значения x7,x6,x9, и откат к дорисовке. Если же x29 пуст, то переход через "иначе 1".

    Если в текущем списке задач, после удаления из него задачи, остается задача с тем же номером, что и удаленная, то находится тройка x30, представляющая эту задачу, x4 заменяется на одноэлементный набор, состоящий из x30, и откат к перерисовке всего экрана. Иначе - переход через "иначе 1".

    Если номер x15 удаленной задачи был больше 1, то находится тройка x30, представляющая задачу с номером, на единицу меньшим, и далее - как в предыдущем случае. Иначе - переход через "иначе 1".

    Это - ситуация, складывающаяся при удалении единственной задачи списка задач из просмотра этого списка. В этом случае осуществляется ввод пустого бланка новой задачи взамен удаленной задачи (совершенно аналогично тому, как было описано ранее). Заметим, что при удалении единственной задачи списка не из просмотра задач, а из оглавления задачника такие действия не выполняются.

б) Длина набора x14 равна 4, то есть удаляется выделенная полоса задачи. Проверяется, что выделенная полоса относится к задаче рассматриваемого списка задач (x1 равно началу набора x14) и что она прорисована на экране. Находятся номер x15 задачи, в которой выделена полоса, и тройка x16 этой задачи в наборе x4. Определяются список x17 информационных элементов полос изображения данной задачи, а также вхождение x18 в этот список информационного элемента x19 удаляемой полосы. Далее рассматриваются следующие подслучаи:

б1) Элемент x19 имеет заголовок "терм", то есть удаляется терм задачи (посылка либо условие). Переменной x20 присваивается задача с номером x15, а переменной x21 - удаляемый терм.

Сначала рассматривается случай, когда либо x21 является посылкой задачи x20, либо x20 - задача на описание и x21 - ее условие. Переменной x22 присваивается вхождение x21 в список посылок либо условий, а x23 - 0 в случае посылки и 1 в случае условия. Проверяется, что число вспомогательных информационных элементов с заголовками "набор","слово", предшествующих x18, равно x23 (перед условиями задачи должна размещаться полоса с целевой установкой задачи, которая и определяется информационным элементом данного типа). Переменной x24 присваивается символьный номер вхождения удаляемого терма задачи x20 в его список. Затем находится узел x30 задачи x20, определяется список x33 ссылок на логические терминалы, хранящие посылки (при $x23 = 0$) либо условия (при $x23 = 1$) задачи, и удаляется терминал, хранящий терм x21. Далее - переход через "ветвь 3".

Происходит исключение терма x21 из списка посылок либо условий задачи x20 и соответствующее исключение разрядов в наборе весов и наборе списков комментариев. Информационный элемент x19 удаляется из списка x17, причем x18 устанавливается на вхождение в x17 элемента, следующего за удаленным. Полосы изображения, следующие за удаляемой, сдвигаются вверх на величину ее высоты. Предпринимается коррекция информационных элементов этих полос, соответствующая их сдвигу. Корректируются для дорисовки изображения значения x7,x6,x9, и откат к дорисовке.

Далее рассматривается случай, когда x21 - условие задачи на доказательство либо преобразование. Этот случай аналогичен предыдущему; основное отличие состоит в том, что условие задачи x20, вес условия и комментарии к задаче заменяются на нули.

б2) Элемент x19 имеет заголовок "набор". Здесь происходит удаление целевой установки задачи. Определяется ссылка x25 на узел задачи, и исключаются логические терминалы "цели","время","ответ", хранящие целевую установку и данные о предыдущих попытках решения. Условия и посылки задачи объединяются (в файле задачника) в один общий список посылок - в том же порядке, в каком они прорисованы на экране. Затем происходит повторное считывание задачи из задачника, и x4 заменяется на одноэлементный набор, состоящий из тройки x27 для считанной задачи.

б3) Элемент x19 имеет заголовок "геомредактор". Находится ссылка x25 на узел задачи; удаляется логический терминал "точка", хранящий чертеж задачи. Если имелся терминал "точки" со старой версией чертежа, то эта версия преобразуется в текущую. Затем откат к перерисовке.

12. x10 = "прогрпеременная". Нажата клавиша "ы" для смены режима полной либо сокращенной прорисовки задачи. В обычном режиме происходит сокращенная прорисовка задачи - отбрасываются простейшие посылки и условия, определяющие тип значения переменной. В полном режиме прорисовываются все посылки и условия. Указателем на наличие режима полной прорисовки служит комментарий "полный" к посылкам исходной задачи. В зависимости от того, имелся ли этот комментарий, он вводится либо удаляется. Затем происходит повторное считывание оператором "извлечениезадачи" первой задачи набора x4. При считывании используется подоператор "элементызадачи", учитывающий наличие комментария "полный". Переменной x4 присваивается одноэлементный набор, образованный тройкой для считанной задачи, и откат к перерисовке.

13. x10 = "тринадцать" либо x10 = "число". Нажата клавиша Enter либо "T" (кир.) для ввода нового условия либо новой посылки. В случае Enter для ввода используется формульный редактор, в случае "T" - текстовый. Инициализируются переменные x11 и x12. Первая из них будет указывать тройку набора x4 для задачи, в которую заносится новый терм (сначала x11 присваивается последняя тройка набора x4); вторая - номер этой задачи. Инициализируются нулями переменные x13 и x14. Первая из них - индикатор вставки нового терма в последнюю задачу списка (значение 0) либо перед какой-либо задачей списка (значение 1). Вторая определяет номер строки, с которой набирается новый терм.

    Находится, если он есть, комментарий (смполоса A), определяющий список выделенных задач и элементов задач. При наличии выделенной задачи либо элемента задачи вставка нового терма будет происходить непосредственно перед этой задачей либо элементом. Если выделена задача, последняя полоса изображения которой прорисована на экране, то x11 заменяется на тройку выделенной задачи из набора x4, x12 - на номер выделенной задачи. Значение x14 сначала изменяется на номер x7 первой свободной строки. Если после выделенной задачи на экране прорисована верхняя отделяющая линия следующей задачи, то x14 изменяется на номер строки этой линии, x13 изменяется на 1. Далее - откат к переходу через "ветвь 2".

    Если выделенных задач в списке A нет, то переход через "иначе 4", где рассматривается случай выделенной полосы изображения некоторой задачи. x18 - четверка из A, определяющая эту полосу. Находятся номер x19 задачи, в которой выделена полоса, и тройка x20 этой задачи в наборе x4. Переменной x21 присваивается вхождение информационного элемента x22 выделенной полосы изображения в список информационных элементов полос изображения тройки x20. Проверяется, что элемент x22 относится к целевой установке задачи либо к ее терму (посылке либо условию). Переменной x23 приваивается номер строки, с которой начинается выделенная полоса, и проверяется, что начиная с этой строки имеется минимально необходимое для ввода нового терма место. Переменной x24 присваивается набор информационных элементов полос изображения тройки x20. Находится число x25 предшествующих выделенной полос изображения, определяемых информационными элементами типа "набор" либо "слово", и проверяется, что оно не превосходит 1. Если оно равно 0, то водимый терм нужно относить к списку посылок, иначе - к условиям. Проверяется, что в случае ввода условия задачи на доказательство либо преобразование эта задача

еще не имеет условия (соответствующий разряд задачи есть логический символ "пустоеслово"). Номер выделенной полосы в наборе x18 увеличивается на 1 - чтобы после вставки посылки или условия выделенной оказалась прежняя полоса и можно было продолжить ввод новых термов.

Расчищается область экрана вниз от начала x23 выделенной полосы. Корректируются все информационные элементы полос изображения, начиная с выделенной полосы - их последний разряд обнуливается, чтобы при откате они рассматривались как непрорисованные. x6 и x9 корректируются для дорисовки начиная с полосы, в которой будет вводиться новый терм. Затем - переход через "ветвь 2", где отдельно рассматриваются случаи ввода терма формульным и текстовым редактором.

Формульным редактором вводится терм x27 (при отказе от его ввода - откат к дорисовке начиная с полосы редактирования и отмена выделений). Переменной x30 присваивается задача, в которую происходит вставка, а переменной x31 - вхождение списка посылок этой задачи при $x25 = 0$ и списка условий (условия) при $x25 = 1$.

Если вводится условие, и тип задачи отличен от "описать", то по вхождению x31 регистрируется терм x27, и откат к оператору "ветвь 3". Иначе - находится число x32 информационных элементов полос изображения, предшествующих выделенной полосе и содержащих тот же терм, что и выделенная полоса. Просмотр элементов обрывается, как только встречается элемент, не имеющий заголовка "терм". Это число позволит идентифицировать в списке посылок либо условий то конкретное вхождение терма выделенной полосы (если таких вхождений несколько), перед которым нужно добавлять новый терм. После определения x32 - переход через "ветвь 5".

Если выделенная полоса содержала терм x33, то просматривается список посылок либо условий, в котором происходит вставка, и в нем находится вхождение терма x33, перед которым имеется ровно x32 вхождений этого же терма. Затем x32 заменяется на номер найденного вхождения, и переход через "ветвь 1".

Переменной x33 присваивается тот список посылок либо условий, в который вставляется новый терм. Предпринимается собственно регистрация в этом списке терма x27, с соответствующим пополнением списков весов и комментариев. Затем - откат к переходу через "ветвь 3".

Создается информационный элемент x32 полосы, в которой был введен новый терм, и этот элемент регистрируется в списке информационных элементов тройки x20 перед информационным элементом выделенной полосы. В файле задачника пока никаких изменений задачи не происходит - такое изменение будет выполнено впоследствии, при выполнении операции, отличной от ввода нового терма задачи. Это позволяет уменьшить число перезаписей в файле и замедлить накопление в нем "пустот", заполненных старыми версиями текстов. Информация о необходимости коррекции в файле сохраняется в виде комментария (коррекциязадач ...), ссылающегося на задачу и измененный ее список ("посылка" либо "условие"). Такой комментарий вводится; корректируются для дорисовки значения x7,x9, корректируется комментарий (списокзадач ...), и откат к дорисовке.

Действия, предпринимаемые при вводе новой посылки либо условия текстовым редатором, аналогичны описанным выше. Перед обращением к текстовому ре-

дактору прорисовывается переходник для обозначений используемых в задаче переменных формульным и текстовым редакторами.

Выше рассматривался случай вставки перед выделенным элементом задачи. Если никаких выделений не было, причем на экране прорисован последний элемент последней задачи списка, то переменной x14 переприсваивается номер x7 первой свободной строки (см. фрагмент, достижимый по "ветвь 3" от начальной точки обработки данной команды x10). В этом случае, а также в рассмотренном ранее случае выделенной целиком задачи, далее - переход через "ветвь 2" в начальном фрагменте обработки команды.

Новый терм здесь вводится в конце изображения задачи и регистрируется в списке посылок, если еще не была введена целевая установка, иначе - в условиях задачи. Прежде всего, в случае пустого бланка задачи номер x14 строки для ввода нового терма уменьшается на 19 (отбрасывается вертикальный 19-пиксельный пробел после разделяющей линии, вводимый для удобства работы с пустым бланком задачи). Далее идут действия, аналогичные описанным выше: проверяется наличие свободного места для ввода терма; находится число x16 полос типа "набор" либо "слово", относящихся к задаче; если задача, в которой происходит ввод терма, не последняя, то вниз от строки x14 расчищается экран и все информационные элементы полос в последующих задачах помечаются как непрорисованные. Собственно ввод нового терма и его регистрация аналогичны случаю вставки терма перед выделенным элементом. В тех случаях, когда пополняется список посылок либо условий, терм заносится в конец этого списка.

14. x10 = "транслвыражения" либо x10 = "кортеж" либо x10 = "нормпроизведениевсех". Соответственно, нажаты клавиши "Ctr-ф" (изменение выделенного терма задачи формульным редактором) либо "Ctr-т" (изменение выделенного терма текстовым редактором) либо "ф" (продолжение редактирования выделенного терма задачи формульным редактором).

Находится комментарий (смполоса ...) к посылкам исходной задачи и проверяется, что он определяет выделение единственного объекта - полосы некоторой задачи. Переменной x15 присваивается информационный элемент выделенной полосы. Проверяется, что он имеет заголовок "терм" и что его полоса прорисована на экране. Переменной x17 присваивается номер задачи, к которой относится выделенная полоса, а x18 - определяющая задачу тройка набора x4. Переменной x19 присваивается список информационных элементов полос изображения данной задачи, а переменной x20 - вхождение в x19 информационного элемента выделенной полосы. Находится номер x21 строки, с которой будет происходить ввод новой версии изменяемого терма. В случае Ctr-ф эта строка - первая строка под полосой изменяемого терма, а в двух оставшихся случаях - первая строка данной полосы. Проверяется, что для ввода новой версии терма имеется минимально необходимое место. Переменной x22 присваивается число полос типа "набор" или "слово", предшествующих полосе изменяемого элемента. В случае равенства x22 нулю изменяется посылка, иначе - условие задачи. Проверяется, что в случае изменения условия задачи, не имеющей типа "описать", это условие уже имелось. Комментарий (смполоса ...) удаляется. Расчищается область экрана начиная со строки x21. Все информационные элементы полос изображения, идущие после изменяемой полосы, помечаются как

непрорисованные. x6, x9 переустанавливаются на дорисовку начиная с изменяемой полосы. Затем - переход через "ветвь 2".

Сначала рассматриваются два случая изменения терма формульным редактором ("Ctr-ф" либо "ф"). Во втором случае вводится комментарий "формоперанды" к посылкам исходной задачи. При обращении к формульному редактору в первом случае происходит обычный ввод новой версии терма под его старой версией; во втором случае - формульному редактору вместо указателя цвета символов передается старая версия терма. Это приводит к автоматическому перенабору старой версии, причем комментарий "формоперанды" переключает формульный редактор, по завершении автоматического перенабора, на обычный режим ручного набора формулы. В обоих случаях, по завершении редактирования, получаем новую версию терма x24. Переменной x27 присваивается задача, в которой происходит изменение, а x28 - вхождение в x27 списка посылок либо списка условий (условия), в зависимости от того, изменяется ли посылка или условие.

Если изменяется условие задачи на доказательство либо преобразование, то переход через "иначе 3", где сразу же старая версия, расположенная по вхождению x28, заменяется на x24. Такая же замена происходит в информационном элементе x15, а также в файлах задачника. Экран расчищается вниз от начала изменяемой полосы, корректируется комментарий (списокзадач ...), указывающий на последнюю измененную задачу, и откат к дорисовке.

Если изменяется посылка либо условие задачи на описание, то для идентификации нужного вхождения изменяемого терма (если задача имеет несколько одинаковых посылок либо условий) определяется число x29 полос изображения рассматриваемой задачи, предшествующих изменяемой полосе и содержащих тот же терм (сичтая изменяемую полосу). С помощью x29 определяется вхождение x31 изменяемого терма, и это вхождение заменяется на x24. Обнуливается вес измененного терма; новая версия его регистрируется в информационном элементе x15 и в файлах задачника. Экран расчищается вниз от начала измененной полосы, и откат к дорисовке.

Если терм изменяется текстовым редактором (Ctr-т), то действия аналогичны описанным выше. Перед обращением к текстовому редактору предпринимается прорисовка переходника, указывающего соответствие обозначений используемых в задаче переменных для формульного и текстового редакторов.

15. x10 = "подобл". Нажата клавиша "ч" для ввода либо изменения чертежа. Инициализируется нулем переменная x11. Если выделен единственный объект - задача, верхняя отделяющая линия которой прорисована на экране, то x11 заменяется на тройку набора x4, представляющую выделенную задачу, и комментарий (смполоса ...) удаляется. Иначе проверяется, что верхняя линия последней задачи списка прорисована на экране, и x11 заменяется на последнюю тройку набора x4. Далее - переход через "ветвь 2".

   Проверяется, что x11 не равно 0. Теперь x11 - тройка набора x4, представляющая ту задачу, для которой будет редактироваться чертеж. x12 присваивается эта задача.

   Сначала рассматривается случай изменения ранее введенного чертежа. Находится информационный элемент x13 полосы изображения задачи x12, определяющий чертеж. Переменной x14 присваивается тройка (описание чертежа в

формате оператора "геомредактор" - верхняя строка чертежа - нижняя строка чертежа). В верхней части экрана устанавливается меню редактирования чертежа, и происходит обращение к геометрическому редактору. После редактирования восстанавливается меню просмотра списка задач; определяются нижняя строка чертежа x15 и верхняя строка x16, и эти новые значения регистрируются в тройке x14 вместо старых. Описание чертежа модифицируется в файлах задачника и в комментариях к посылкам задачи x12. Переменной x4 присваивается одноэлементный набор, состоящий из тройки x11; все информационные элементы полос изображения в x11 помечаются как непрорисованные, и откат к перерисовке.

В случае ввода нового чертежа (переход через "иначе 1") - устанавливается меню геометрического редактора, расчищается область экрана вниз от верхней отделяющей линии задачи x12, и происходит обращение к геометрическому редактору. После редактирования восстанавливаетя меню просмотра списка задач; определяются нижняя и верхняя строки чертежа x16,x17; чертеж регистрируется в комментариях к посылкам задачи x12 и в файлах задачника; в тройке x11 вводится информационный элемент полосы чертежа (с удалением вертикального 19-пиксельного пробела, если чертеж вводится непосредственно на пустом бланке задачи); набор x4 преобразуется, как и в случае изменения чертежа, и откат к перерисовке.

16. x10 = "облнорм". Нажата клавиша "Ц" для просмотра и редактирования оглавления типов целевых установок. Переменной x12 присваивается тип текущего оглавления - согласно комментарию (вид ...). Находится одноэлементный набор x14, образованный ссылкой на корневой указатель оглавления типов целевых установок, и комментарий (вид ...) переустанавливается на данное оглавление. Текущее содержимое экрана сохраняется в буфере, и предпринимается обращение к оглавлению типов целевых установок. При отказе от выбора его концевого пункта восстанавливаются экран и исходный комментарий (вид ...), после чего - откат к запросу очередной команды. Если концевой пункт оглавления выбран, то расчищаются экран и буфер текстов; в верхней части экрана перерисовывается текст выбранного концевого пункта, под которым проводится горизонтальная линия. Находится содержимое логического терминала выбранного концевого пункта. Если оно отлично от терма "фикс(...)", то представляет собой логический символ, и этот символ прорисовывается под горизонтальной линией. Затем - вход в цикл обработки команд редактирования данного символа (этот символ используется как указатель процедуры, обрабатывающей ввод целевой установки данного типа).

17. x10 = "верхняяоценка". Нажата клавиша "ц" для выбора целевой установки задачи через оглавление типов целевых установок. Переменной x11 присваивается последняя тройка набора x4, переменной x12 - задача этой тройки, переменная x13 инициализируется нулем. Далее этим переменным будут переприсвоены, соответственно, тройка набора x4, определяющая задачу, в которой редактируется целевая установка; данная задача, и номер строки, с которой будет происходить редактирование.

Если имеется комментарий (смполоса A), то просматривается список A. Если в нем имеется пара x17, выделяющая задачу с номером x18, то находится тройка x19 набора x4, соответствующая данной задаче. Если в задаче еще не

была введена целевая установка (отсутствует информационный элемент полосы изображения с заголовком "набор"), то переменным x11, x12 присваиваются тройка x19 и задача этой тройки, а переменной x13 - первая строка изображения следующей задачи (при отсутствии таковой - первая свободная строка x7). Если же целевая установка имелась, то проверяется, что она прорисована на экране, и тогда переменной x13 присваивается первая строка полосы прорисовки целевой установки. Переменные x11, x12 корректируются так же, как и выше. В обоих случаях далее - откат к оператору "ветвь 2".

Отдельно рассматривается случай, когда в списке A имеется четверка, выделяющая полосу изображения некоторой задачи (переход к рассмотрению этого случая - через "иначе 4"). Определяются номер x18 этой задачи и ее тройка x19 в наборе x4. Проверяется, что целевая установка задачи еще не была введена (заголовок задачи есть 0, то есть это бланк задачи с не определенным пока ее типом). Данная ситуация означает, что целевая установка задачи вводится перед выделенной полосой, причем все термы, предшествующие полосе, будут отнесены к посылкам задачи, а все термы начиная с полосы - к ее условиям. Изначально все прорисованные на экране термы бланка задачи относились к ее списку посылок. Находится информационный элемент x21 выделенной полосы изображения; проверяется, что он имеет заголовок "терм". Удаляется комментарий (смполоса ...); текущий экран сохраняется в буфере; комментарий (вид ...) переустанавливается на оглавление типов целевых установок, и предпринимается обращение к этому оглавлению. После выбора концевого пункта оглавления указатель (вид ...) переустанавливается обратно на оглавление задачника, и находится содержимое x28 логического терминала выбранного концевого пункта. Экран восстанавливается и сразу же снова сохраняется. Значением x28 теперь является одноэлементный набор, состоящий из логического символа S, к программе которого нужно обращаться через справочник "смцель" для реализации диалога ввода целевой установки. После обращения к данному справочнику определяется тройка x29 (информационный элемент полосы изображения для целевой установки - список целей - тип задачи). Здесь определяется список x30 термов, располагавшихся в списке посылок задачи начиная с выделенной полосы, и переход через "ветвь 3".

Проверяется, что в случае неодноэлементного набора x30 рассматриваемая задача имеет тип "описать". В файлах задачника предпринимается коррекция списков посылок, условий и целевой установки задачи. Затем - переход через "ветвь 1".

Переменной x31 присваивается текущая (до изменения целевой установки) версия рассматриваемой задачи. Тип ее изменяется согласно тройке x29. Далее предпринимается доопределение задачи x31, в зависимости от ее типа. Заметим, что до этого x31 представляло собой лишь бланк задачи, то есть являлось набором длины 4: тип задачи, список посылок, список весов посылок и список комментариев к посылкам. Поэтому при формировании задачи конкретного типа к набору x31 добавляется необходимое число разрядов. По окончании преобразований набора x31 - переход через "ветвь 1" (измененная версия задачи сохраняется в тройке x19).

Здесь происходит вставка информационного элемента полосы изображения целевой установки в список информационных элементов полос изображения тройки x19; все информационные элементы полос изображения, начиная с полосы

целевой установки, помечаются как непрорисованные; корректируются для дорисовки x6, x7 и x9; расчищается област экрана вниз от начала полосы целевой установки; корректируется комментарий (списокзадач ...), и откат к дорисовке.

Выше рассматривался случай вставки целевой установки перед выделенной полосой. Если никаких выделений не было, то в исходном фрагменте обработки команды - переход через "ветвь 3", где проверяется, что на экране прорисована верхняя линия последней задачи списка. Если эта задача уже имела целевую установку, то проверяется, что она прорисована на экране и переменной x13 присваивается верхняя строка полосы целевой установки; если целевой установки еще не было, то проверяется, что последний элемент задачи прорисован, и переменной x13 присваивается номер x7 первой свободной строки. В обоих случаях (а также в разобранном выше случае выделенной задачи) далее - откат к оператору "ветвь 2".

Здесь проверяется, что x13 не равно 0. Если редактируется пустой бланк задачи, то x13 уменьшается на величину вертикального 19-пиксельного пробела. Затем - переход через "ветвь 1".

Текущий экран сохраняется в буфере. Далее, как и выше, реализуется обращение к оглавлению типов целевых установок и считывание содержимого x18 логического терминала выбранного концевого пункта. Экран восстанавливается и соазу же снова сохраняется. Если x18 содержало логический символ, относящийся к "шахматной ситуации" - инициализации партии либо вводу промежуточной игровой позиции, то организуется специальный интерфейс ввода задачи на выбор очередного хода в шахматах (она оформляется как задача на исследование). Переход к программе этого интерфейса - через "ветвь 3". Иначе - переход через "ветвь 2".

Предпринимается обращение к справочнику "смцель", которое приводит к появлению тройки x19 (информационный элемент для прорисовки целевой установки - список целей - тип задачи). Проверяется, что в случае уже определенного типа задачи этот тип совпадает с указанным в x19. Скорректированная либо новая целевая установка регистрируется в файлах задачника, и переход через "ветвь 1".

Если задача уже имела ненулевой тип, то ее список целей изменяется согласно x19. Иначе - в зависимости от определившегося типа ее набор пополняется необходимыми разрядами, как это делалось в разобранном выше случае. Затем - переход через "ветвь 1".

Просматриваются информационные элементы полос изображения редактируемой задачи. При обнаружении элемента типа "набор" он заменяется на информационный элемент полосы целевой установки из тройки x19. Далее расчищается область экрана вниз от верхней линии редактируемой задачи; все ее информационные элементы полос изображения помечаются как непрорисованные; удаляется комментарий (смполоса ...); корректируется для дорисовки x9, и откат к оператору "ветвь 1". Если же информационного элемента типа "набор" не обнаружено, то переход через "иначе 2".

Если редактируется пустой бланк задачи, то отбрасывается информационный элемент вертикального 19-пиксельного пробела. Затем к концу списка информационных элементов полос изображения добавляется элемент из тройки x19.

Особо рассматривается случай, когда редактируется задача на преобразование. Если ее целевая установка вводится после того, как было введено преобразуемое выражение, то предусмотрено автоматическое перенесение данного выражения в условия задачи (до ввода целевой установки оно попадало в список посылок). При отсутствии данного особого случая - как и выше, откат к оператору "ветвь 1" из предыдущего фрагмента.

Здесь завершаются необходимые для отката к дорисовке коррекции: все информационные элементы полос изображения задач, следующих после редактируемой, помечаются как непрорисованные; изменяются x7 и x6; комментарий (списокзадач ...) переустанавливается на редактируемую задачу. Затем - откат к дорисовке.

18. x10 = "приведение". Нажата клавиша Home для завершающей обработки введенной задачи. Такая обработка приводит к расширению списков посылок и условий простейшими утверждениями о типах значений переменных, однозначно извлекаемых "по умолчанию" из контекста, в котором они встречаются; коррекцией символов для неоднозначно расшифровываемых обозначений (использование знака суммы как для сложения чисел, так и для сложения векторов, и т.п.) на основе анализа контекста, и т.п. В принципе, эта обработка выполняется автоматически при каждом запуске решения задачи, так что данная команда почти избыточна.

Сначала определяется та задача, к которой относится команда. Здесь применяется стандартная процедура: сначала переменным x11 и x12 присваиваются последняя тройка набора x4 и номер задачи этой тройки, причем x13 инициализируется нулем. Затем проверяется наличие комментария (смполоса ...), определяющего выделенную задачу, верхняя линия которой прорисована на экране; x11 и x12 переустанавливаются на эту задачу, а x13 заменяется на 1. При отсутствии такой выделенной задачи проверяется, что последняя задача набора x4 является последней задачей списка и ее верхняя линия прорисована на экране. Тогда x13 тоже заменяется на единицу. Далее в обоих случаях - переход через "ветвь 2".

Здесь проверяется, что x13 равно 1; переменной x15 присваивается номер строки верхней линии модифицируемой задачи, и выполняется обращение к оператору "завершзадачи", модифицирующему задачу и выполняющему изменение записи ее в задачнике. Из набора x4 отбрасываются все тройки, следующие после тройки x11; повторно считывается из задачника модифицированная задача, и корректируется ее тройка в конце набора x4. Расчищается экран в области, расположенной ниже верхней линии модифицированной задачи; значения x6, x7, x9 корректируются для дорисовки, и откат к дорисовке.

Оператор "завершзадачи" выполняет следующие действия по модификации задачи:

а) Если список посылок задачи пуст, то в нем регистрируется единственное утверждение "истина".

б) Для каждой посылки и условия выполняются обращения к оператору "коррекцияобозначений", уточняющему расшифровку введенного символа с учетом типов объектов, к которым он относится (например, выбор одной из операций: умножение чисел, умножение вектора на число, умножение матриц; все эти

операции вводятся формульным редактором одинаково, но обозначаются разными символами).

в) Определение с помощью справочника "типданных" типов значений переменных, встречающихся в задаче и пополнение списков посылок и условий (в случае неизвестных задачи на описание) утверждениями, фиксирующими эти типы значений. Аналогичное пополнение утверждениями о типе значений переменных конструкций со связанными переменными.

г) В случае неодноэлементного списка посылок - удаление посылки "истина".

д) Ввод посылки "планиметрия" для тех геометрических задач, которые усматриваются как заведомо планиметрические.

е) Сопровождение задачи на доказательство комментарием "одз".

19. x10 = "помощь". Нажата клавиша Ctr-ц для просмотра и редактирования целевой установки текстовым редактором. Последовательность действий здесь мало чем отличается от случая ввода целевой установки через оглавление типов целевых установок. Сначала в наборе x4 определяется тройка x11 редактируемой задачи, и переменной x12 присваивается номер этой задачи, а переменной x13 - строка, с которой начинается редактирование списка целей. В качестве x11 берется сначала последняя тройка набора x4; если есть выделенная задача, то она заменяется на ее тройку. Если была выделена полоса изображения бланка задачи с не определенным пока типом, то далее выполняется ветвь программы, переход к которой - через оператор "иначе 4"; в противном случае - ветвь программы, переход к которой - через "ветвь 2". Перед обращением к текстовому редактору прорисовывается переходник для обозначений входящих в задачу переменных формульным и текстовым редакторами. Затем прорисовываются в скобочной записи цели задачи (если они уже были введены), причем в начале списка целей присоединяется тип задачи. При редактировании целей первым элементом списка должен быть введен тип задачи. Для определения информационного элемента полосы изображения целевой установки применяется оператор "элементызадачи", который будет описан в разделе, посвященном программам пошагового просмотра решения.

20. x10 = "внешдизъюнкция". Нажата клавиша Insert для вставки копий выделенных термов. Находится комментарий (смполоса A), и переменной x13 присваивается набор указателей выделенных объектов A. Проверяется, что этот набор непуст. Переменной x14 присваивается последний элемент набора A. Он определяет, куда нужно производить вставку копий: если выделена задача, то - в конце ее полос изображения; если выделена полоса изображения, то - перед ней. Элемент x14 исключается из набора x13. Далее предпринимается просмотр набора x13 и заполнение накопителя x15 вставляемых термов на основе тех элементов из x13, которые определяют выделенные термы либо подтермы. После этого - переход через "иначе 2".

Проверяется наличие комментария (Смполоса ...), аналогичного комментарию (смполоса ...), но сохраняющему выделенные объекты текущей задачи процесса пошагового просмотра решения. Формат этого комментария будет описан в разделе, посвященном программам пошагового просмотра. Здесь его следует учитывать потому, что из текущего кадра пошагового просмотра можно (не

обрывая решения) перейти к просмотру задач задачника и воспользоваться выделенными в текущей задаче термами для регистрации их в этих задачах. Если комментарий есть, то извлекаемые из него выделенные термы добавляются в начале списка x15. Далее - переход через "ветвь 1".

Проверяется, что список x15 непуст. Затем рассматриваются два случая: элемент x14 определяет выделенную задачу либо выделенную полосу.

Если x14 определяет задачу, то находится ее номер x16 в рассматриваемом списке задач. Находится узел x21 этой задачи, и предпринимается регистрация термов списка x15 в ее посылках (если еще не введена целевая установка задачи) либо в ее условиях (пока - только в файла задачника). Затем - переход через "ветвь 2".

Находится тройка x17 для выделенной задачи в наборе x4, и из нее извлекается сама задача x18. Предпринимается вставка термов набора x15 в задачу x18 - так же, как это было сделано в файлах задачника. В случае условия задачи на доказательство либо преобразование происходит замена старой версии условия на первый терм набора x15 (остальные термы из x15 здесь не используются). Переменной x19 присваивается список информационных элементов полос изображения задачи x18. Этот список просматривается для определения вхождения x21 в него того элемента, после которого будут вставляться термы. Переменная x20 при этом играет роль индикатора перехода через полосу целевой установки: до нее она равна 0, а затем 1. Далее - переход через "ветвь 2". Здесь проверяется, не имеет ли места работа с пустым бланком задачи. Если это так, то в x19 отбрасывается вертикальный 19-пиксельный пробел. Затем - переход через "ветвь 1".

Создается список x22 информационных элементов полос изображения для термов набора x15, и находится результат x23 вставки этого списка в набор x19 после вхождения x21. В тройке x17 набор x19 заменяется на x23. В зависимости от того, был ли прорисован элемент по вхождению x21, выполняется подготовка к дорисовке, и затем - откат к дорисовке. Перед откатом вводится комментарий (автоклавиатура ...), инициирующий нажатие после дорисовки клавиши пробела, сбрасывающей все выделения.

Если x14 определяет не задачу, а полосу изображения, то выполняется похожая последовательность действий. Для уточнения того, добавляются ли термы к условиям или к посылкам задачи, находится число x22 информационных элементов типа "набор" либо "слово", предшествовавших элементу выделенной полосы. При x22 = 0 термы относятся к посылкам, иначе - к условиям.

21. x10 = "транзитпереход". Нажата клавиша Ctr-Ins для перенесения выделенного терма на новое место. Находится комментарий (смполоса ...), определяющий список x13 выделенных объектов. Проверяется, что таких объектов ровно два, причем первый из них - полоса, в которой прорисован терм (его нужно перенести), а второй - задача либо полоса (куда будет перенесен терм). Находятся тройка x16 набора x4, соответствующая второму выделенному объекту, и список x17 информационных элементов полос изображения этой тройки. Проверяется, что если второй выделенный объект - полоса, то ей предшествует (в той же самой задаче) не более одной полосы типа "набор" либо "слово". Комментарий (смполоса ...) удаляется. Определяется ссылка x23 на узел той задачи, из

которой берется переносимый на новое место терм. Переменной x25 присваивается эта задача, и находится список x26 информационных элементов ее полос изображения. Проверяется, что выделенная полоса, определяющая переносимый терм, имеет не более одной предшествующей полосы типа "набор" либо "слово". Находится число x29 информационных элементов в x26, предшествующих элементу выделенной полосы и определяющих тот же самый терм. Это число нужно для идентификации удаляемого терма в списке посылок (x28 = 0) либо условий (x28 = 1). Далее предпринимается исключение переносимого терма из представления задачи x25 в задачнике. Затем - откат к оператору "ветвь 2".

Находится ссылка x22 на узел задачи, в которую переносится терм; переменной x24 присваивается сам этот терм. Если задача была выделена целиком, то предпринимается регистрация x24 в конце списка условий, а если целевая установка задачи еще не введена, то - списка посылок. Если был выделен терм задачи, то x24 регистрируется перед соответствующей посылкой либо условием (аналогично тому, как это делалось при вставке группы термов). Все изменения затрагивают здесь только файлы задачника. Затем - откат к оператору "ветвь 1".

В наборе x4 оставляется только тройка, к которой относится первый прорисованный на экране элемент, причем эта повторно считыается из задачника; экран расчищается; значения x6, x7, x9 переустанавливаются для дорисовки начиная с первого ранее прорисованного элемента, и откат к дорисовке.

22. x10 = "Стандплюс" или x10 = "Обобщподст". Нажата клавиша "к" либо "Ctr-к" (в обоих случаях кир.) для копирования выделенной задачи либо перенесения ее на новое место. Находится комментарий (смполоса ...) и проверяется, что он определяет выделение не более чем двух объектов, причем оба эти объекта - задачи. Комментарий удаляется, и находится ссылка x20 на узел первой из задач (она должна быть скопирована либо перенесена на новое место).

Сначала расматривается случай, когда выделен один объект (список x13 имеет длину 1). Тогда копирование либо перенесение происходит в конец списка задач. Проверяется, что последняя тройка набора x4 относится к последней задаче списка. Выбирается случайным образом логический символ x27, в статье которого будет введен новый узел задачи, и такой узел x29 вводится; x30 - его номер. В оглавлении задачника ссылка на новый узел заносится в конец рассматриваемого списка задач; создается встречная ссылка из узла на оглавление, и в новом узле копируется представление задачи узла x20 (перенесение задачи выполняется последовательным копированием и удалением). Предпринимается считывание только что созданной задачи, и переменной x4 присваивается одноэлементный набор из считанной тройки. Затем - откат к оператору "ветвь 2".

Если выделены две задачи, то переход через "иначе 3", где переменной x22 присваивается ссылка на вторую задачу, а переменной x23 - метка перехода к ней. Проверяется, что вторая задача относится к текущему списку задач. В этом списке увеличиваются на 1 номера меток перехода к задачам, начиная с метки x23. Сначала это делается в возвратных ссылках на оглавление из узлов задач, а затем (после перехода через "ветвь 1") - в самом оглавлении. Далее, как и выше, выбирается случайным образом логический символ x27, в статье

которого вводится новый узел задачи x29. Ссылка на новый узел задачи из оглавления делается по метке x23; прочие действия аналогичны действиям для одноэлементного x13. Далее, как и выше, переход к оператору "ветвь 2".

Здесь сначала рассматривается случай перенесения задачи, в котором, после регистрации копии задачи, нужно еще удалить исходный ее экземпляр. Если задача переносится на новое место в рамках одного и того же списка задач, то за счет вставки копии номер ее удаляемого экземпляра может увеличиться на 1. Поэтому сначала уточняется номер удаляемой задачи списка, и лишь затем выполняются действия по удалению. Далее - откат к перерисовке.

23. x10 = "Текзадача". Нажата клавиша Ctr-F4 для сброса информации об ответе на выделенную задачу, полученном при предыдущем запуске ее решения. Находится комментарий (смполоса ...) и проверяется, что в нем выделена единственная задача; определяется тройка x16 этой задачи в наборе x4. Комментарий (смполоса ...) удаляется. Проверяется, что верхняя линия задачи прорисована на экране. Находится ссылка x21 на узел задачи, и удаляются, если они есть, логические терминалы "ответ" и "время" данного узла. Из набора x4 удаляются все тройки, следующие за тройкой x16. Повторно считывается из задачника выделенная задача, и тройка x16 заменяется на новую свою версию. Фактически так из нее удаляются информационные элементы полос, в которых прорисовывались ответ и данные о трудоемкости решения. Экран расчищается начиная с верхней линии выделенной задачи, значения x6, x7, x9 корректируются для дорисовки, и откат к дорисовке.

24. x10 = "внешконъюнкция". Нажата клавиша "курсор вправо" для входа в выделение подтерма выделенного терма. Находится комментарий (смполоса A), и рассматривается последний элемент x14 набора A. Определяется тройка x15 набора x4, соответствующая задаче, к которой относится выделенный элемент x14.

Если длина набора x14 равна 4, то есть выделена полоса изображения задачи, то находитя информационный элемент x17 данной полосы и проверяется, что он имеет заголовок "терм". В случае прорисовки терма формульным редактором переменной x16 присваивается пара (терм - корень дерева указателей для прорисовки терма); в случае прорисовки его текстовым редактором переменной x16 присваивается тройка (терм - 0 - строка для начала прорисовки).

Если длина набора x14 равна 5, то находится информационный элемент x17 полосы изображения, к которой относится выделенный элемент, и проверяется, что он имеет заголовок "набор". В x17 находится тройка x18, соответствующая выделенному элементу полосы, и проверяется, что она имеет заголовок "терм". Если терм тройки x18 прорисовывается формульным редактором, то переменной x16 присваивается пара (терм - корень дерева указателей), иначе - тройка (терм - столбец начала прорисовки - строка начала прорисовки).

После того, как значение переменной x16 определилось для двух указанных выше случаев (изначально эта переменная была инициализирована нулем) - откат к оператору "ветвь 2".

Сначала рассматривается случай прорисовки терма формульным редактором; здесь длина набора x16 равна 2. Оператор "формоперанды" определяет максимальные по включению допускающие независимую прорисовку формульным

редактором собственные подтермы терма T из x16 и присваивает переменной x17 набор пар (корень дерева указателей для прорисовки подтерма - вхождение подтерма). В стандартных ситуациях данные подтермы суть просто корневые операнды терма T. Инициализируется стэк x18 просмотра подтермов терма T; в него заносится пара (выделенный подтерм списка x17 - список x17). Последовательность пар стэка x18 (просматриваемых с конца к началу) будет определять путь в терме T к текущему выделенному подтерму. Терм T при выделении был перекрашен в светло-синий цвет; теперь он обратно перекрашивается оператором "блокредактора" в черный цвет. Переменной x21 присваивается вхождение заголовка терма T (далее эта переменная будет указывать вхождение выделенного подтерма). Затем - оператор "повторение", к которому будут происходить откаты при смене выделенного подтерма.

Переменной x22 присваивается пара (корень дерева указателей для прорисовки подтерма - вхождение подтерма) для того подтерма, который выделен согласно первому элементу стэка x18; x23 становится равно первому элементу пары x22, а x24 - второму. Переменной x21 переприсваивается значение x24. Подтерм перекрашивается в светло-синий цвет. Переменной x27 присваивается вхождение пары x22 в список, перечисляющий такие пары для текущего слоя подтермов. Далее - оператор "повторение", к которому будут происходить откаты при обработке команд просмотра подтермов. Запрашивается очередная такая команда x28. Рассматриваются следующие подслучаи:

а) x28 - "внешсумма" либо "внешконъюнкция" либо "циклвариантов". Соответственно, нажаты клавиша "курсор вверх" для возвращения к надтерму, либо клавиши "курсор вправо"; "курсор влево" для смены подтерма в одном слое. Восстанавливается черный цвет ранее выделенного подтерма, корректируется необходимым образом стэк x18, и откат к перекраске нового выделенного терма. В том случае, когда надтерм отсутствовал, первая из указанных команд приводит к обрыву выделения подтерма и восстановлению изначального светло-голубого цвета всего терма T.

б) x28 - "подстановка". Нажата клавиша "курсор вниз" для перехода к следующему слою подтермов (обычно - операндов). Корректируется стэк x18; текущий выделенный подтерм перекрашивается обратно в черный цвет, и откат к перекраске нового выделенного подтерма (то есть первого из списка подтермов нового слоя).

в) Нажата какая-либо клавиша, отличная от перечисленных выше. Выделение подтерма обрывается и к определяющему выделенный элемент набору x14 добавляется указатель вхождения выделенного подтерма.

В случае прорисовки терма текстовым редактором для выделения подтерма используется оператор "просмотртерма", перед обращением к которому создается комментарий (указательвхождения 0). Через этот комментарий пересылается указатель выделенного вхождения. Кроме того, данный комментарий инициирует выход по истинностному значению "ложь" из оператора "просмотртерма" при нажатии любой клавиши, кроме клавиш выделения подтермов.

25. x10 = "усмчисло". Нажата клавиша "c" для включения либо выключения режима просмотра выделенного терма в скобочной записи. Находится информационный элемент (смполоса ...) и проверяется, что он выделяет единственный

объект, представляющий собой полосу изображения задачи. Находится тройка x16 набора x4, соответствующая этой задаче. Проверяется, что выделенная полоса изображения определяется информационным элементом x18, имеющим заголовок "терм" и что она прорисована на экране. Если выделенный терм был прорисован формульным редактором, то вводится комментарий "терм" к посылкам исходной задачи. Все информационные элементы полос изображения рассматриваемой задачи, начиная с x18, помечаются как непрорисованные. Предпринимается обращение к оператору "высотаполосы", корректирующему информационный элемент x18 для прорисовки терма с учетом наличия или отсутствия комментария "терм" (в первом случае в x18 вводится информация для прорисовки текстовым редактором, во втором случае - формульным). Комментарий "терм", если он был введен, удаляется. Экран расчищается вниз от выделенной полосы; оператор "смполоса" прорисовывает эту полосу согласно скорректированному элементу x18. Затем помечаются как непрорисованные все информационные элементы полос изображения задач, идущих после рассматриваемой задачи; x6, x7, x9 корректируются для дорисовки, и откат к дорисовке.

26. x10 = "блокприемов". Нажата клавиша "Ctr-курсор вверх" для просмотра частично прорисованной в верхней части экрана задачи с ее начала, либо (если экран уже начинается с верхней отделяющей задачу линии) для перехода к просмотру предыдущей задачи. Находятся первая тройка x11 набора x4 и список x12 ее информационных элементов полос изображения. Если первый из этих элементов не прорисован на экране, то экран расчищается; x6, x7, x9 корректируются на дорисовку начиная с первого элемента набора x12; все информационные элементы полос изображения в x4 помечаются как непрорисованные, и откат к дорисовке. Если же первый элемент набора x12 прорисован, то проверяется, что номер x13 задачи тройки x11 больше 1; из задачника считывается предыдущая задача, и переменной x4 присваивается одноэлементный набор, образованный тройкой x14 считанной задачи. Далее - откат к перерисовке.

27. x10 = "символтеоремы". Нажата клавиша "Ctr-курсор вниз" для перехода к просмотру задачи, следующей за той задачей, которая частично или полностью прорисована в верхней части экрана. Если набор x4 одноэлементный, то находится номер x11 задачи, представленной в этом наборе. Если в списке задач имеется задача с номером x12, на единицу большим номера x11, то эта задача считывается, и x4 заменяется на одноэлементный набор, образованный тройкой x13 считанной задачи. Затем - откат к перерисовке. Если же набор x4 имеет более одного элемента, то его первый элемент отбрасывается; экран расчищается; значения x6, x7, x9 корректируются для дорисовки начиная с верхней линии первой из оставшихся в x4 задач; все информационные элементы полос изображения помечаются как непрорисованные, и откат к дорисовке.

28. x10 - один из символов "деление" (клавиша "а"), "частичныйответ" (клавиша "о"), "числитель"(клавиша "р") , "внешнийквантор" (клавиша "г"), "минимум" (клавиша "л"), "область" (клавиша "Л"), "оператор" (клавиша "Ctr-р"), "натуральное" (клавиша "Ш"), "," (клавиша "Р"), "прообраз" (клавиша "П"), "новыесвязки" (клавиша "и"), "переобозначение" (клавиша "Ctr-п"), "числзначение" (клавиша "Ч"), "дробнаявеличина" (клавиша "Ctr-ч").

Все обозначения клавиш здесь взяты на основе кириллицы. Эти команды запускают процесс решения выделенной задачи (при отсутствия выделения - задачи, к которой относится первая сверху отделяющая горизонтальная линия) в различных режимах. Напомним эти режимы:

а) клавиша "а" - решение с выходом в отладчик ЛОСа на шаге, номер которого для данной задачи ранее был сохранен путем нажатия клавиши "щ" в процессе трассировки;

б) клавиша "о" - решение без трассировки;

в) клавиша "р" - решение с трассировкой;

г) клавиша "г" - переход в оглавление ГЕНОЛОГ'а для выбора приема прерывания и запуска решения;

д) клавиша "л" - переход в оглавление программ для выбора точки прерывания и запуска решения;

е) клавиша "Л" - указание логического символа, при обращении к программе которого следует выйти в отладчик ЛОСа, и запуск решения;

ж) клавиша "Ctr-р" - решение задачи на текстовый анализ с полным показом как синтаксического разбора, так и процесса решения.

з) клавиша "Ш" - указание номера шага работы интерпретатора ЛОСа, на котором следует выходить в отладчик ЛОСа, и запуск решения;

и) клавиша "Р" - двойное решение задачи: сначала в обычном режиме, с определением по окончании решения тех шагов, которые оказались ненужными; при повторном решении происходит пошаговый показ решения с пропуском этих шагов;

к) клавиша "П" - решение задачи с созданием технического протокола;

л) клавиша "и" - запуск игровой задачи с ручным вводом всех ходов (игра нескольких игроков без решателя либо пошаговый просмотр известной партии);

м) клавиша "Ctr-п" - запуск решения задачи до выявления отличия от технического протокола;

н) клавиша "Ч" - запуск решения геометрической задачи для построения чертежа после первых шагов решения задачи;

о) клавиша "Ctr-ч" - запуск решения геометрической задачи для построения чертежа по исходному условию задачи.

Программа обработки указанных выше команд прежде всего проверяет, не реализуется ли оператор "списокзадач" внутри внешней программы оператора "цепьзадач". При наличии такой программы выполнение команд блокируется, так как в этом случае обращение к просмотру списка задач имеет место из процесса трассировки ранее запущенного решения некоторой задачи.

Далее предпринимается определение той задачи x11, к которой относится команда запуска решения. Сначала переменным x11 и x12 присваиваются нули; впоследствии x12 станет номером выбранной задачи в списке задач.

При наличии комментария (транскоммент ...) задача x11 определяется сразу - это первая задача списка x4. Напомним ситуацию, в которой применяется

такой комментарий. При пошаговом просмотре решения задачи очередное преобразование сопровождается списком вспомогательных задач, которые были решены для выполнения данного преобразования. Интерфейс предоставляет две возможности для пошагового просмотра решения вспомогательной задачи - либо запуск этого решения в контексте прерванного на текущем шаге решения основной задачи, либо откат к повторному решению задачи из списка задач, с прерыванием перед решением данной вспомогательной задачи. Во втором случае, после внутреннего перезапуска, и возникает комментарий (транскоммент A_1 A_2) - он указывает номер A_1 шага, на котором следует устанавливать прерывание при обращении к решению задачи типа A_2 (она и будет рассматриваемой вспомогательной задачей).

Если комментария (транскоммент ...) нет, то проверяется наличие выделенной задачи. Если такая задача есть, то решаться будет она. Перед запуском решения эта задача обрабатывается оператором "завершзадачи", что делает избыточной указанную выше команду "Home". Исключение составляют обобщенные задачи с заголовками "текстформ" и "структура" - соответственно, анализ русского текста и численное моделирование процесса.

Далее, если задача не выделена, но имеется комментарий (списокзадач ...), указывающий на последнюю измененную задачу, то в случае, когда эта задача относится к текущему списку задач, решаться будет она. Наконец, во всех остальных случаях решается та задача, которая определяется первой прорисованной на экране верхней отделяющей линией. Во всех этих случаях к выбранной для решения задаче применяется оператор "завершзадачи".

После переопределения значений x11 и x12 проверяется, что x11 не равно 0 и не является недоопределенным бланком задачи - то есть что заголовок x11 отличен от 0. В случае команды "новыесвязки" проверяется, что x11 есть задача на исследование с целью "шахматы", причем вводится цель "смход" этой задачи, указывающая на режим разыгрываемой вручную партии двух игроков. Если не имеет место режим серийного решения задач из заданного раздела, то указатель текущего пути в оглавлении задачника переустанавливается на выбранную для решения задачу. Далее - переход через "ветвь 1" к фрагменту, в котором осуществляется разветвление дальнейших действий в зависимости от типа задачи.

Прежде всего, рассматривается случай обычной задачи, имеющей тип "преобразовать", "доказать", "описать" либо "исследовать".

Если обрабатываются команды построения чертежа "дробнаявеличина", "числзначение", либо имеется комментарий "чертеж", указывающий на серийное построение чертежей для заданного раздела задачника, то задача x11 снабжается целью (в случае задачи на доказательство - комментарием) "чертеж". Для команды усиленного построения чертежа "числзначение" вводится также цель либо комментарий "геомформат". Далее сбрасывается, если он был, комментарий (геомредактор ...) к посылкам задачи x11 - при построении чертежа старая его версия не используется. Продолжение действий по запуску решения - через оператор "ветвь 2".

Здесь задача x11 снабжается комментарием посылок (просмотрзадачи A_1 A_2), представляющим собой ее координаты в задачнике (A_1 - логический символ; A_2

- номер узла статьи символа A_1, являющегося узлом задачи). Затем - переход через "ветвь 1".

Инициализируется нулем переменная x13 - она будет преобразована в номер шага работы интерпретатора, по достижении которого нужно выйти в отладчик ЛОСа. Переменной x14 присваивается исходная задача. Если x10 = "," (решение с пропуском показа неиспользуемых шагов), то вводится комментарий "см-источник" к посылкам задачи x14, указывающий на режим двукратного решения с расчисткой выдаваемых на экран шагов. Затем переход через "ветвь 1".

Сбрасываются все ранее введенные установки на прерывание. Далее рассматриваются следующие случаи:

а) x10 = "числитель" (решение с трассировкой). Вводятся комментарий (прерывание решить) к задаче x11, который в начале ее решения инициирует ввод установки на трассировку по шагам решения данной задачи. Кроме того, вводится комментарий (буфер пустоеслово) к посылкам исходной задачи, в котором будут накапливаться данные о вспомогательных задачах, решенных перед срабатыванием очередного приема. Подробнее о нем будет сказано в следующем разделе, посвященном пошаговому просмотру решения. Сразу заметим, что ряд связанных с этим комментарием действий выполняется на уровне интерпретатора ЛОСа. При наличии комментария (буфер ...) включаются процедуры отладчика, обеспечивающие обращение к пошаговому показу решения.

б) x10 = "внешнийквантор" (установка прерывания через оглавление базы приемов). Активируется 8-й информационный блок; текущий экран сохраняется в буфере, и находится корень оглавления базы приемов. Затем - оператор "повторение", к которому будут происходить откаты при возвращении из просмотра концевого пункта оглавления базы приемов в это оглавление. Комментарий (вид ...) к посылкам исходной задачи переустанавливается на оглавление базы приемов. Огранизуются обращение к оглавлению базы приемов, и вслед за этим - обращение к редактору приемов ГЕНОЛОГа (процедура "блокприемов") для просмотра выбранного концевого пункта оглавления. Если был выбран прием прерывания, то по выходе из редактора приемов появится комментарий "продолжение" к посылкам исходной задачи. Кроме того, будет введен комментарий (левыйкрай A_1 A_2 A_3) либо (начало A_1 A_2 A_3) к посылкам той же задачи. Здесь (A_1, A_2, A_3) - ссылка на прием прерывания. Этот комментарий передается от исходной задачи задаче x11, причем заголовок "начало" заменяется на "контрольприема" и в начале набора добавляется логический символ "прерывание", который становится новым заголовком комментария. В начале решения задачи x11 на основе данного комментария будет создана установка на прерывание при обращении к указанному приему (для второго комментария - только в стэковом кадре задачи x11).

в) x10 = "минимум" (установка прерывания через оглавление программ). Активируется 9-й информационный блок; комментарий (вид ...) переустанавливается на оглавление программ; находится корень этого оглавления, и текущий экран сохраняется в буфере. Далее предпринимается обращение к оглавлению программ, и после выбора концевого пункта находится содержимое x18 его логического терминала. По x18 определяются логический символ S и номер N оператора "прием(N)", при достижении которого следует делать прерывание.

Далее вводится комментарий (текоператор S N) к посылкам задачи x11; если N - цифра, то она в комментарии преобразуется в одноэлементный набор. Этот комментарий в начале решения задачи x11 инициирует ввод установки на прерывание при достижении указанной контрольной точки.

г) x10 = "прообраз" (решение с созданием технического протокола). Если отсутствует 4-й информационный блок (он используется для хранения технического протокола), то этот блок вводится. Его корневым объектом служит не каталог, а указатель-список. 4-й информационный блок активируется, и хранящийся в нем технический протокол (если он есть) сбрасываается. Затем предпринимаются уплотнение 4-го информационного блока и повторная его активация. Вводятся комментарий (прерывание решить) к посылкам задачи x11 и комментарий (техпротокол (новый) ...) к посылкам исходной задачи. Первый из них инициирует установку на прерывания по шагам решения задачи x11, но в сочетании со вторым, используемым как вспомогательный регистр процесса создания технического протокола, эта установка приведет лишь к сохранению в техническом протоколе кадров решения, без вывода их на экран.

д) x10 = "переобозначение" (решение до выявления отличия от технического протокола). Проверяется наличие 4-го информационного блока и наличие технического протокола в этом блоке. Затем вводятся комментарий (прерывание решить) к посылкам задачи и комментарии (техпротокол (различны) ...), (буфер пустоеслово) к посылкам исходной задачи.

е) x10 = "область" (решение с прерыванием при обращении к программе заданного логического символа). Текущий экран сохраняется в буфере; в левом нижнем углу экрана прорисовывается текст "Логический символ:", и под ним реализуется обращение к текстовому редактору для ввода названия логического символа x21. Затем сбрасываются все установки на прерывания и водится установка на прерывание при обращении к программе символа x21. Экран восстанавливается.

ж) x10 = "натуральное" (решение с прерыванием при достижении заданного шага работы интерпретатора). Если имеется комментарий (транскоммент A_1 A_2) к посылкам исходной задачи, то номер шага прерывания берется из него: этот номер полагается равным уменьшенному на 100 числу A_1 (поправка на возможные рассогласования в подсчете шагов при использовании режима трассировки и без этого режима). Номер шага прерывания передается переменной x13 (установка прерывания по x13 будет выполнена непосредственно перед началом решения, чтобы сделать ее не зависящей от трудоемкости дальнейших процедур, предваряющих это начало). При отсутствии комментария (транскоммент ...) для ввода номера шага прерывания предпринимается обращение к текстовому редактору; введенное значение передается переменной x13.

з) x10 = "деление" (решение с прерыванием по достижении точки, зафиксированной на предыдущем запуске с помощью клавиши "щ"). Находится ссылка x19 на узел задачи x11, и переменной x21 присваивается содержимое логического терминала "повторение" этого узла (если он есть). Находится номер x23 шага, сохраненный в данном терминале, и переменной x13 присваивается результат вычитания из x23 единицы (именно при такой коррекции обеспечивается точное восстановление ситуации, имевшей место при нажатии "щ").

После рассмотрения перечисленных выше случаев - откат к оператору "ветвь 1", расположенному непосредственно перед оператором "равно(x10 числитель)".

Если исходная задача имеет комментарий "внешконтроль", указывающий на режим серийного решения задач из некоторого раздела задачника, сопровождаемый анализом холостого хода приемов, то проверяется наличие комментария (выписка ...) к посылкам исходной задачи, и если такого комментария нет, то вводится комментарий (выписка пустоеслово). Включается режим учета холостого хода приемов: при откате к оператору "контрольприема(...)", обозначающему начало некоторого приема, счетчик холостого хода данного приема, создаваемый в комментарии (выписка ...), увеличивается на число тысяч шагов работы интерпретатора, затраченных на попытку применения приема.

Если исходная задача имеет комментарий "задачи", указывающий на наличие серийного решения задач из некоторого раздела задачника, то находится список x15 всех логических символов, встречающихся в посылках и условиях задачи x11, и создается комментарий (задачи x15 пустоеслово) к посылкам исходной задачи. В этом комментарии будет накапливаться информация о всех логических символах, возникающих в процессе решения задачи, а также о всех приемах, которые применяются хотя бы однажды. По окончании решения эта информация будет зарегистрирована в архиве задачника; она используется при поиске в задачнике.

Если в буфере задачника введены разделы, перечисляющие задачи, в которых срабатывают заданные приемы, то оператор "буферзадачника" создает комментарий (список ...) для учета срабатываний этих приемов в буфере задачника.

По комментарию (просмотрзадачи ...) к посылкам задачи x11 находится ссылка x19 на узел этой задачи. Если имеется логический терминал "блок" данного узла, то переменной x21 присваивается его содержимое - список термов "прием(...)", перечисляющий те приемы ГЕНОЛОГа, применение теорем которых должно быть заблокировано при решении задачи x11. Эта блокировка достигается с помощью оператора "трассировка(фильтр ...)". После ее установки интерпретатор ЛОСа отслеживает попытки обращения к оператору "контрольприема" для заблокированных приемов и делает результат обращения ложным.

Инициализируется пустым словом переменная x15, которой затем переприсваивается содержимое логического терминала, достижимого из корня 2-го информационного блока по меткам "трассировка", "трассировка". Это содержимое представляет собой набор установок на режим трассировки решения задачи (ручной либо автоматический вход в подпроцессы; пропуск приемов некоторых типов и т.п.). На его основе создается либо корректируется комментарий (трасснабор x15) к посылкам исходной задачи, который учитывается отладчиком ЛОСа при пошаговом показе решения.

Если x10 = "," (двойное решение задачи с отбрасыванием ненужных действий при пошаговом показе), то создается комментарий (Копия ...) к посылкам исходной задачи, в котором сохраняется копия исходной версии задачи x11 для повторного запуска решения.

Если логический терминал, достижимый из корня 2-го информационного блока по меткам "приемы", "стоп", непуст, то для каждого находящегося в нем терма "прием(A_1 A_2 A_3)", ссылающегося на прием, применение которого блокируется, создается комментарий (приемы A_1 A_2 A_3) к посылкам исходной задачи. Этот

комментарий обеспечивает блокировку при обращении приема к процедуре, реализующей собственно изменение задачи.

Если в серийном решении задач встречается задача на исследование, то она пропускается (такие задачи обычно определяют игровые процессы).

Наконец, начинающийся с оператора "прием(3 8)" фрагмент, в котором собственно происходит обращение к решению задачи. Если не имеет места серийный режим решения задач и не происходит построение чертежа, то в верхней части экрана вводится меню интерфейса пошагового просмотра решения. Определяется начальный момент запуска решения (x15 - часы, x16 - минуты, x17 - секунды, x18 - сотые доли секунды). Устанавливается максимальный уровень 16 для решения задачи x11 (в процессе развития базы приемов этот уровень несколько раз повышался, и при необходимости его можно будет повышать и далее; впрочем, вряд ли он будет увеличен сколь- нибудь значительным образом). Определяется номер x19 текущего шага работы интерпретатора и устанавливается прерывание по достижении шага с номером x13 + x19 (случай x13 = 0 означает, что прерывание данного типа отсутствует, так как шаг x19 уже достигнут). Вводится комментарий (трассировка x19 0), сохраняющий номер x19 начального шага запуска задачи и указывающий, что при работе с отладчиком и трассировке будут предприниматься попытки прорисовки термов в стандартной математической записи. После этого предпринимается обращение к решению задачи x11, и переменной x20 присваивается ответ (либо "отказ"), полученный при решении.

Определяется момент окончания решения: переменным x21 - x24 присваиваются, соответственно, часы, минуты, секунды и сотые доли секунды. Предпринимается обращение к оператору "учетответа", выполняющему регистрацию результатов решения (ответ, трудоемкость, данные для архива задачника) в задачнике. Оператор "буферзадачника" регистрирует решавшуюся задачу в буфере задачника, если при ее решении сработал прием, для которого в этом буфере был введен подраздел.

Если исходная задача имела комментарий "смисточник", указывающий на режим двойного запуска решения с исключением из пошагового показа неиспользованных приемов, то реализуется повторный запуск решения задачи - на этот раз с фактическим показом шагов решения. Сохраненный после первого запуска комментарий (повтор ...) позволяет при этом определять, было ли использовано введенное при срабатывании приема утверждение. По завершении трассировки происходит откат к просмотру списка задач. Для этого в логический терминал "метка" корневого каталога задачника заносится терм "задачи(5)", обеспечивающий вход в задачник при перезапуске, и осуществляется внутренний перезапуск системы. Предварительно текущий путь задачника устанавливается на решавшуюся задачу.

Если исходная задача имеет комментарий (техпротокол ...), то предпринимается уплотнение 4-го информационного блока. Затем прорисовываются извлекаемые из указанного комментария данные о 5 наибольших замедлениях на промежутках между срабатываниями приемов текущего решения по сравнению с решением, хранящимся в техническом протоколе.

Если имеется комментарий "задачи", указывающий на режим серийного решения, то происходит внутренний перезапуск.

Если решалась задача на преобразование, имеющая цель "график", то есть после выполнения преобразований условия задачи нужно войти в интерфейс построения и просмотра графика, то реализуется обращение к выполняющей эти действия процедуре "график". По окончании работы с графиком - переменной x4 присваивается одноэлементый набор, соответствующей текущей задаче построения графика, и откат к перерисовке.

В логический терминал "метка" корневого каталога задачника заносится терм, обеспечивающий возвращение к просмотру задачника (в случае задачи на исследование - к предоставлению хода игроку) после внутреннего перезапуска, и текущий путь оглавления задачника устанавливается по решавшейся задаче. В случае задачи на исследование (игровая задача) после этого предпринимается внутренний перезапуск.

Определяется время, затраченное на решение - в минутах и секундах, - и десятичные записи этих величин регистрируются в логическом терминале, достижимом из корневого указателя задачника по меткам "сброс", "время". Они будут использованы для прорисовки ответа после внутреннего перезапуска. Далее - внутренний перезапуск.

Кроме описанных выше действий, выполняемых при решении обычной задачи, предусмотрены действия для решения извлекаемых из задачника "обобщенных" задач.

Прежде всего, это задачи на текстовый анализ. Они представляют собой тройки (текстформ A_1 A_2), где A_1 - пара (логический символ - номер узла статьи этого символа), ссылающаяся на узел задачи в задачнике; A_2 - текстформульная структура, определяющая считанный из задачника текст задачи. В зависимости от команды ("Ctr-p" либо "p"), вводится комментарий (схемафразы 0) либо (схемафразы 3), определяющий режим трассировки. Собственно решение задачи на анализ текста выполняется процедурой "текстоваязадача". После получения ответа и регистрации его в задачнике - экран расчищается, и откат к дорисовке решавшейся задачи начиная с текста ответа.

Наконец, для работы с обобщенной задачей на моделирование процесса (набор с заголовком "структура") используется оператор "структура". Он реализует интерфейс просмотра и редактирования логического описания моделируемого объекта (это описание имеет иерархический характер и требует специального интерфейса; в просматриваемом списке задач на экран выводится лишь текст концевого пункта оглавления, являющийся названием объекта), а также запускает процесс моделирования.

29. x10 = "схемаидентификации". Нажата клавиша "ш" для перехода к просмотру следующего элемента списка просматриваемых задач. Такой список формируется заранее различными интерфейсами (например, для просмотра задач с измененным ответом либо имеющих наибольшие замедления, и т.п.) и хранится в комментарии (подборнеизвестных ...). Программа создает комментарий (автоклавиатура ...) для повторного нажатия клавиши "ш" и выводит во внешнюю программу оглавления задачника, где нажатие "ш" обеспечивает переход к следующему элементу списка.

30. x10 = "старшийчлен". Нажата клавиша "э" для просмотра и редактирования комментариев к задаче либо ее выделенному терму. Находится комментарий

(смполоса ...), определяющий выделенные объекты; проверяется, что такой объект единственный, и переменной x14 присваивается набор, выделяющий этот объект.

Сначала рассматривается случай, когда x14 - набор длины 2, выделяющий некоторую задачу. Переменной x19 присваивается ссылка на узел этой задачи в задачнике; переменной x21 - список комментариев к задаче. Экран сохраняется в буфере, и предпринимается обращение к оператору "просмотрсписка", которому передается список комментариев. С помощью интерфейса указанного оператора осуществляется редактирование списка комментариев. При выходе из оператора "просмотрсписка" - восстанавливается экран и изменения комментариев регистрируются в задачнике.

Если набор x14 имеет длину 4, то проверяется, что информационный элемент x18 выделенной полосы изображения имеет заголовок терм. Для определения того, относится ли этот терм к посылкам или условиям задачи, подсчитывается число x19 предшествующих выделенной полосе полос изображения той же задачи, имеющих тип "набор" либо "слово"(0 - посылки, 1 - условия). Для идентификации вхождения выделенного терма в соответствующий список задачи подсчитывается число x20 предшествующих выделенной полос изображения, в которых находится терм, совпадающий с выделенным; просмотр обрывается при обнаружении полосы, тип которой отличен от "терм". Находится узел задачи, к которой относится выделенный терм; дальнейшие действия аналогичны первому случаю.

31. x10 = "правмногоугольник". Нажата клавиша "я" для инициализации задачи на анализ текста. Предварительно был введен пустой бланк задачи.

    Сначала переменной x11 присваивается последняя тройка набора x4, переменной x12 - номер задачи этой тройки в списке задача. Переменные x13 и x14 инициализируются нулями. Если выделена единственная задача списка, то переменной x11 переприсваивается ее тройка, x12 - соответствующий номер; x13 становится равно 1, а x14 - номеру строки, следующей за последней использованной для прорисовки выделенной задачи. Если никаких задач не выделено, причем последняя задача списка целиком прорисована на экране, то переменной x14 переприсваивается номер первой свободной строки.

    Далее - переход через "ветвь 2", где сначала проверяется, что x14 отлично от 0, т.е. имел место один из указанных выше двух случаев. В случае пустого бланка задачи (он и должен здесь быть) x14 уменьшается на высоту вертикального 19-пиксельного пробела. Проверяется, что рассматриваемая задача x15 представляет собой пустой бланк. Переменной x16 присваивается пустая текстформульная структура, и предпринимается обращение к оператору "текстформ" для ввода условия задачи на анализ текста. Затем находится узел x22 рассматриваемого бланка задачи; отбрасывается ветвь "задача" этого узла, и вводится ветвь "текстформ", содержащая введенное условие. Наконец, корректируется список информационных полос изображения тройки x11, и откат к перерисовке.

32. x10 = "учетобоснований". Нажата клавиша Ctr-я для редактирования выделенной задачи на анализ текста. Находится тройка x15 набора x4, соответствующая выделенной задаче, и проверяется, что ее текстформульная полоса прорисована на экране. Эта полоса повторно обрабатывается оператором "текстформ", и изменения регистрируются в задачнике.

33. x10 = "группировки". Нажата клавиша End для возвращения в главное меню либо (при просмотре списка задач из трассировки) для возвращения в кадр трассировки. Если имеется внешняя (по цепочке обращений) процедура "цепь-задач", т.е. происходит трассировка, то выход из текущей процедуры "список-задач". Иначе - текущий путь в оглавлении задачника переустанавливается на первую из задач, верхняя линия которой прорисована на экране, и внутренний перезапуск.

34. x10 = "заменазнака". Нажата клавиша "С" (кир.) для ввода либо отмены режима просмотра всех задач в скобочной записи. Соответственно, вводится либо удаляется определяющий такой режим комментарий "терм" к посылкам исходной задачи.

35. x10 = "извлечениеварианта". Нажата клавиша "т" (кир.) для входа в диалоговый блок установки режима трассировки. Сначала переменной x11 присваивается список логических символов - установок на трассировку, содержащийся в логическом терминале, достижимом из корня 2-го информационного блока по меткам "трассировка","трассировка". Затем создается структура данных x12 блока диалога, обеспечивающего просмотр и изменение установок режима трассировки, и прорисовывается его пустая рамка. В начало буфера текстов заносится символ "+", который далее прорисовывается против тех пунктов блока диалога, для которых в x11 имеется соответствующая установка. Происходит обращение к процедуре "диалог" для ввода команд диалога. Эти команды либо корректируют список x11 (с соответствующей прорисовкой либо удалением плюсов), либо, по завершении диалога по Enter, модифицируют содержимое логического терминала "трассировка","трассировка" корня 2-го информационного блока.

36. x10 = "редакторответа". Нажата клавиша "Б" для просмотра и редактирования списка приемов, заблокированных при решении задачи. Переменной x11 присваивается номер в списке задач выделенной задачи, либо, если ее нет, номер первой из задач, верхняя линия которой прорисована на экране. Находится ссылка x18 на узел x11-й задачи и определяется содержимое x20 ее логического терминала "блок". В этом терминале перечисляются ссылки "прием(A_1 A_2 A_3)" на приемы, применение теорем которых должно быть заблокировано. Переменной x21 присваивается список этих теорем. Список x21 передается для организации просмотра оператору "просмотрсписка". Если при просмотре выбирается некоторая теорема, то далее предпринимается обращение к просмотру всех ее приемов оператором "блокприемов". Если при просмотре некоторые теоремы заносятся в буфер путем нажатия клавиши "б", то после отмены просмотра их приемы исключаются из логического терминала "блок". Заметим, что команда "Б" может только отменить ряд ранее введенных блокировок; ввод новых блокировок осуществляется через редактор приемов.

37. x10 = "заменагруппы". Нажата клавиша "б" для перехода к основной версии задачи, скопированной в буфер задачника. Сначала определяется номер x11 той задачи, к которой относится команда (так же, как в предыдущем пункте). Определяется узел этой задачи; находится возвратная ссылка на оглавление (она указывает только на основную версию задачи); текущий путь в оглавлении задачника устанавливается по этой возвратной ссылке, и выход по истинностному значению "ложь" для отката к процедуре оглавления задачника.

38. x10 = "стандменьше". Нажата клавиша F1. Осуществляется обращение к процедуре "Помощь", реализующей интерфейс справочника по логической системе.

39. x10 = "верхняягрань". Нажата клавиша "И" для инициализации либо редактирования задачи на анализ рисунка. Значения x11 - x14 определяются аналогично случаю инициализации задачи на анализ текста. Рисунок создается либо корректируется при помощи оператора "рисунки". Он формируется в специальном массиве интерпретатора ЛОСа, называемом буфером рисунков. Чтобы не перегружать информационный блок задачника, битмэпы задач на анализ рисунков хранятся в отдельном, тринадцатом, информационном блоке. Из узла задачи по метке "битмэп" - переход к логическому терминалу, хранящему терм "битмэп(A_1 A_2 A_3)", являющийся ссылкой на битмэп рисунка в 13-м информационном блоке. Здесь A_1 - логический символ, по которому от корневого каталога 13-го блока имеется переход к указателю-списку, текстовый терминал "битмэп" которого содержит битмэп. A_2, A_3 - ширина рисунка и длина его двоичного кода в 16-битных словах.

40. x10 = "значение". Нажата клавиша "М" для инициализации задачи на моделирование процесса. Редактирование этой задачи можно выполнить после этого, нажав клавишу "р" и войдя в просмотр иерархической структуры данных, описывающей объект моделирования. Инициализация состоит в отбрасывании ветви "задача" узла пустого бланка задачи и создании пустой ветви "структура" этого узла.

17.2 Пошаговый просмотр решения

Пошаговый просмотр решения инициируется установкой прерывания по преобразованиям выбранной задачи из задачника и вводом комментария (буфер пустоеслово) к посылкам исходной задачи. При применении очередного преобразования этой задачи происходит обращение к отладчику ЛОСа, который отображает на экране информацию о преобразовании и о контексте, в котором оно применено. Очередной шаг решения реализуется после нажатия клавиши Enter. При этом, в зависимости от установленного режима трассировки, отладчик может самостоятельно принять решение о показе шагов решения какой-либо вспомогательной задачи, или же такое решение принимает пользователь, нажимая клавишу Ctr-Enter.

Отображение информации об очередном шаге сначала подготавливается отладчиком ЛОСа, который анализирует текущий контекст, а затем полученные данные передаются процедуре "цепьзадач", которая, собственно, и реализует интерфейс просмотра.

17.2.1 Структура данных для описания текущего шага решения

Информация о текущем шаге решения представляется в виде так называемой цепи обобщенных задач. Обобщенная задача - либо обычная задача, либо описание обращения к некоторому макрооператору, либо вспомогательная конструкция, описывающая текущее преобразование или вводящая другие пояснения.

Сначала в цепи перечисляются обобщенные задачи $Z_1, \ldots, Z_n$, где каждое Z_{i+1} возникло как подзадача для $Z_i; i = 1, \ldots, n-1$. Эта часть цепи определяет текущий контекст решения; после нее размещается фиктивная задача, описывающая собственно преобразование. Если для выполнения преобразования были решены какие-либо вспомогательные задачи либо осуществлены обращения к макрооператорам, то некоторые из них отбираются отладчиком, и соответствующие обобщенные задачи помещаются в конце цепи. Кроме указанных элементов цепи обобщенных задач, могут вводиться элементы, указывающие на наличие разбора случаев и перечисляющие результаты уже разобранных подслучаев. Такие элементы размещаются между задачей, инициировавшей разбор случаев, и задачей, соответствующей текущему подслучаю.

Перечислим типы обобщенных задач и используемые для их представления структуры данных:

1. Обычные задачи на доказательство, преобразование, описание и исследование.

2. Задача на анализ текста - тройка (текстформ A_1 A_2), где A_1 - пара (логический символ - номер узла статьи этого символа), ссылающаяся на узел задачи в задачнике; A_2 - текстформульная структура, определяющая условие задачи.

3. Задача нормализатора - набор (нормализатор A_1 A_2 A_3 A_4). Эта задача представляет собой информацию об обращении к операторному выражению ЛОСа, называемому нормализатором. A_1 - заголовок этого оператора; (A_3, A_2, A_4) - значения входных переменных. Первая входная переменная нормализатора имеет своим значением некоторый терм; вторая - список утверждений, называемых посылками. Нормализатор выполняет тождественные либо (если указанный терм является утверждением) эквивалентные преобразования терма в предположении истинности посылок. A_4 - набор вспомогательных информационных элементов, используемых для передачи нормализатору управляющей информации и для получения такой информации от него. Эти элементы называются комментариями.

   Нормализатор аналогичен задаче на преобразование, однако он использует для преобразований лишь ограниченное число приемов, и это существенно ускоряет получение результата. Его приемы, как и приемы решения задач, обычно (хотя не всегда) задаются на ГЕНОЛОГе; часто они дублируют друг друга. Возникает некоторый "пакет" отобранных в нормализатор приемов, из-за чего нормализаторы, а также операторные аналоги задач на описание, доказательство и исследование называются пакетными операторами. Эти операторы оказались чрезвычайно эффективным средством для ускорения вычислений и организации локального перебора. Подробнее об их устройстве будет рассказано в разделах, посвященных ГЕНОЛОГу.

4. Задача проверочного оператора - набор (легковидеть A_1 A_2 A_3 A_4). Здесь A_1 - название оператора; A_3 - проверяемое утверждение; A_2 - список утверждений, в предположении истинности которых предпринимается проверка (посылок); A_4 - список комментариев. Проверочный оператор является пакетным оператором, аналогичным задаче на доказательство.

5. Задача синтезатора - набор (синтезатор A_1 A_2 A_3 A_4). Здесь A_1 - название оператора; A_3 - утверждение, для обеспечения истинности которого выполняется подбор значения заданной его переменной; A_2 - список утверждений, в

предположении истинности которых предпринимается подбор; A_4 - список комментариев. Синтезатор является пакетным оператором, аналогичным задаче на описание.

6. Задача анализатора - набор (анализатор A_1 A_2 A_3 A_4). Здесь A_1 - название оператора; A_3 - задача на доказательство либо на исследование, для которой анализатор предпринимает вспомогательный вывод следствий. Эти следствия, однако, не заносятся непосредственно в список посылок задачи, а регистрируются в буфере анализатора A_2. Этот буфер представляет собой тройку (набор полученных следствий - набор наборов комментариев к этим следствиям - конкатенация набора следствий и списка посылок рассматриваемой задачи). Лишь специально отобранные следствия переносятся при работе анализатора в список посылок задачи. A_4 есть список комментариев анализатора.

7. Фиктивная задача для выдачи информации о примененном приеме - (прием A). Здесь A - набор информационных элементов следующих типов:

   а) (замена B_1 B_2) - применяется преобразование замены посылки либо условия задачи, либо замены преобразуемого нормализатором терма, либо замены следствия, выведенного анализатором. B_1 - указатель на заменяемый терм, представляющий собой в случае задачи пару (вхождение заменяемой посылки либо условия в свой список - 0 в случае посылки и 1 в случае условия); в случае нормализатора равный 0, а в случае анализатора представляющий собой заменяемое утверждение из буфера анализатора. B_2 - заменяющий терм.

   б) (вывод B_1 B_2) - вывод новой посылки (при $B_1 = 0$) либо нового условия ($B_1 = 1$). B_2 - выведенное утверждение.

   в) (исключение B_1 B_2) - исключение посылки либо условия задачи. B_1 - вхождение исключаемого утверждения; B_2 - указатель посылки (0) либо условия (1).

   г) (прием B) - B есть ссылка на примененный прием - тройка (логический символ - номер узла приема в статье этого символа - заголовок приема) для приема, заданного на ГЕНОЛОГе, и пара (логический символ - номер узла) для приема, реализованного непосредственно на ЛОСе.

   д) (ответ B) - выдается ответ B. Если B - логический символ "легковидеть", то выдается ответ "истина" проверочного оператора.

   е) (текзадача B) - B есть вхождение текущей обобщенной задачи (обычной задачи либо обращения к пакетному оператору) в цепь обобщенных задач.

   ж) (замечание B_1 B_2 B_3 B_4) - вводится (при $B_1 = 1$) либо удаляется (при $B_1 = 0$) комментарий B_4. B_2 - указатель посылок (0) либо условий (1). B_3 - 0 в случае общего комментария, иначе - тот терм, к которому относится комментарий.

   з) (титр B_1 B_2 $C_1 \ldots$ C_n) - информационный элемент, сформированный программой приема и определяющий текст сопровождающих срабатывание приема пояснений. Здесь B_1 - тройка (логический символ - номер узла - заголовок приема), ссылающаяся на примененный прием. Сопровождающий текст создается на основе текстового шаблона, хранящегося в текстовом терминале "титр" узла приема. Этот шаблон распадается на несколько независимых фрагментов, каждый из которых начинается с символа ¬, вслед за которым идет номер фрагмента - последовательность цифр, завершающаяся пробелом. Нумерация

начинается с 1; если фрагмент всего один, то заголовок ¬1 отбрасывается. Для указания мест в шаблоне, на которые должны вставляться формулы, используются пары скобок (). B_2 есть номер (десятичное число) фрагмента шаблона, используемого в сопровождающем тексте (в зависимости от ситуации, прием выбирает один из нескольких фрагментов для сопровождения; сопровождающий текст может также быть составлен из нескольких фрагментов). $C_1, \ldots, C_n$ - термы, вставляемые в шаблон вместо его пар скобок.

и) (попыткаспуска B_1 B_2) - предпринимается попытка замены условия задачи на описание либо доказательство на новое условие, следствием которого оно является. B_1 - вхождение заменяемого условия; B_2 - заменяющее условие.

к) (Замена B_1 B_2 B_3) - замена группы утверждений B_2 на группу утверждений B_3. B_1 - указатель посылки (0) либо условия (1).

л) (разборслучаев B) - инициируется разбор случаев в пакетном нормализаторе по дизъюнкции B.

м) (подслучаи B) - переход к разбору подслучая в пакетном нормализаторе, определяемого конъюнкцией утверждений списка B.

н) (сборкатекста B) - формирование поясняющего текста, составляемого при помощи стандартных заготовок, хранящихся во 2-м информационном блоке. B - последовательность определяющих последовательные элементы текста пар следующих типов: (слово C) - текстовая заготовка, достижимая из корня 2-го информационного блока по меткам C, "слово"; (буква C) - логический символ C является кодом выдаваемой буквы; (терм C) - выдается терм C. Обычно такой способ создания поясняющего текста используется, если прием не создал указанного выше элемента (титр ...).

8. (разборслучаев A_1 A_2 A_3) - фиктивная задача для отображения информации о ранее рассмотренных подслучаях при разборе случаев по дизъюнктивному условию либо дизъюнктивной посылке задачи. A_1 - дизъюнктивный член, соответствующий подслучаю; A_2 - найденный для него ответ; A_3 - задача, для которой проводится разбор случаев.

9. (подслучаи A) - фиктивная задача для отображения информации о текущем рассматриваемом подслучае при разборе случаев в нормализаторе. A - набор утверждений, конъюнкция которых определяет подслучай.

10. (битмэп A_1 A_2) - задача на анализ рисунка. A_1, A_2 - логический символ и номер узла этого символа, определяющие узел задачи в задачнике.

11. (структура A_1 A_2) - задача на моделирование процесса. A_1, A_2 - логический символ и номер узла этого символа, определяющие узел задачи в задачнике.

17.2.2 Определение цепи обобщенных задач отладчиком ЛОСа

Цепь обобщенных задач создается отладчиком ЛОСа в процессе анализа текущего контекста. При описании отладчика мы обозначили лишь точку входа в ветвь его программы, выполняющую эти действия; теперь вернемся к рассмотрению данной ветви. Выйти на ее начало можно, например, через оглавление программ - пункт

"Отладчик ЛОСа - Просмотр текущей информации с помощью оператора ЦЕПЬЗАДАЧ - Исходная точка".

Напомним прежде всего, что значением переменной x5 на момент входа в нашу ветвь служит набор ссылок на кадры глобального стэка, начинающийся с того кадра, информация о программе которого должна быть выдана на экране (обычно это кадр текущей задачи либо текущего пакетного оператора), и заканчивающийся кадром прерывания (т.е. кадром программы отладчика). Ссылка на кадр прерывания присваивается переменной x15, после чего инициализируется пустым словом накопитель x16 цепи обобщенных задач. Инициализируемая пустым словом переменная x17 является накопителем ссылок на кадры этих обобщенных задач в глобальном стэке; для фиктивных задач в x17 заносится 0. Инициализируется нулем переменная x18 (в случае приема, обращающегося к вспомогательной задаче либо к пакетному оператору, значение этой переменной будет изменено на 1).

После оператора "прием(7 1)" размещается оператор "повторение", к которому будут происходить откаты при просмотре кадров глобального стэка, предшествующих кадру прерывания. x15 - ссылка на текущий такой кадр (напомним, что изначально это ссылка на сам кадр прерывания). Переменной x20 присваивается ссылка на источник кадра x15 (в случае не имеющего источника кадра прерывания - ссылка на непосредственно предшествующий кадр); переменной x21 - тип этого кадра. Типы кадров глобального стэка были перечислены в начале раздела, посвященного отладчику ЛОСа.

Кадр x20, непосредственно предшествующий кадру прерывания, может оказаться кадром продолжения перечисления, не представляющим интереса для построения цепи обобщенных задач; для отсечения рассмотрения его в этих случаях предусмотрен оператор "(или(длинатекста(x5 1) ... равно(x20 правпозиция(x5 1))". При откате к оператору "ветвь 3" происходит замена x15 на x20, т.е. происходит очередной шаг вдоль цепи кадров глобального стэка.

Далее сначала рассматривается случай $x21 = 1$ - кадр x20 оказался кадром задачи. Определяется набор x22, несущий информацию об этом стэковом кадре. Последний разряд набора x22 - набор значений определенных в кадре программных переменных. Переменной x23 присваивается задача кадра x20. Если она совпадает с исходной задачей, то цикл просмотра глобального стэка обрывается. Иначе - находится значение x24 в кадре x20 программной переменной "x2" и проверяется, что оно представляет собой вхождение символа дизъюнкции. В этом случае проверяется, что накопитель x16 обобщенных задач непуст и его первая задача - того же типа, что и x23, причем этот тип - "описать" либо "преобразовать". Эти признаки указывают на возможное применение приема разбора случаев по дизъюнктивному условию либо дизъюнктивной посылке - тогда в данной точке нужно будет вставлять фиктивные обобщенные задачи, выводящие на экран результаты уже разобранных подслучаев.

По ссылке x15 на кадр, источником которого служит x20, определяется ссылка x26 на прием, который реализуется в кадре x20. Проверяется, что она имеет длину 2 - то есть прием реализован на ЛОСе и выделяется своей контрольной точкой "прием(N)". Пара x26 состоит из логического символа, к программе которого относится прием (этот символ расположен по вхождению x24, то есть является символом "или"), и номера N. Если N есть 33, то прием осуществляет разбор случаев по дизъюнктивному условию задачи на описание. Его несложно найти через пункт "Приемы решателя - Общие приемы - дизъюнкция - разбор случаев по дизъюнктивному условию задачи на описание - рассмотрение подслучая для дизъюнктивного условия" оглавления программ. Известно, что на момент рассмотрения подслучая значением

переменной x8 служит набор наборов конъюнктивных членов ответов уже рассмотренных подслучаев, а значением переменной x14 - набор дизъюнктивных членов рассматриваемой дизъюнкции. Эти значения находятся в наборе x22 и присваиваются, соответственно, переменным x28 и x29. Далее в начале набора x16 вставляются обобщенные задачи (разборслучаев ...), определяющие результаты разбора случаев, а в начале набора x17 добавляется такое же количество нулей.

Аналогичным образом обрабатывается случай, когда $N = 24$ (разбор случаев по посылке задачи на описание) и $N = 38$ (разбор случаев по посылке задачи на преобразование).

Мы видим из данного примера, что программа формирования цепи обобщенных задач использует конкретные номера контрольных точек и переменных отдельных приемов, реализованных на ЛОСе. Это обстоятельство следует учитывать при внесении в такие приемы каких-либо изменений.

После вставки фиктивных задач для разбора случаев (если она нужна) - переход через оператор "ветвь 6", где к началу набора x16 присоединяется задача x23, а к началу набора x17 - ссылка на кадр x20.

Если x21 $\neq 1$, то происходит переход через "иначе 4", где переменной x22 снова присваивается информация о стэковом кадре x20. Проверяется, что этот кадр является кадром оператора (тип 2) либо операторного выражения (тип 3). Если x20 оказывается кадром прерывания, предшестующим последнему кадру прерывания (тогда в наборе x22 на втором месте расположена переменная), то просмотр глобального стэка обрывается. Такая ситуация возможна при запуске из текущего кадра трассировки решения одной из вспомогательных задач, в процессе которого возникает прерывание. Если x20 не является кадром прерывания, то переход через "ветвь 1".

Переменной x23 присваивается заголовок оператора либо операторного выражения кадра x20. Если этот заголовок - название вспомогательного оператора, размещаемого в начале программы пакетного оператора ("контрольнормализации", "контрольбуфера", и т.п.), то переменной x18 присваивается 1 (см. переход через "ветвь 2"), чтобы отметить тот факт, что выводимое на экран преобразование - обращение к пакетному оператору. Предварительно анализируется случай оператора "контрольбуфера". Если он расположен в программе синтезатора, то вывод на экран информации об обращении блокируется, так как это уже сделано предшествующим вспомогательным оператором "синтезатор", располагаемым в начале программ синтезаторов. Такая же блокировка предпринимается для внешней программы проверочного оператора, если установка на трассировку отключает показ обращений к этим операторам. Блокировка достигается выходом из отладчика (оператор "ответ(1)"). При отсутствии блокировки - переход через "ветвь 1", где переменной x24 присваивается ссылка на источник кадра x20.

Далее сначала рассматривается случай, когда x20 - кадр операторного выражения. Проверяется, что x23 - заголовок некоторого пакетного нормализатора (используется справочник "быстрпреобр", определяющий тип нормализатора). Затем переменной x25 присваивается набор значений переменных, определенных в кадре x20, а переменной x26 - обобщенная задача нормализатора для кадра x20. Если в комментариях к нормализатору имеется набор (разборслучаев ...), указывающий на наличие режима разбора случаев (подробнее об этом режиме будет сказано в разделах, посвященных ГЕНОЛОГу), то создается вспомогательная обобщенная задача (подслучаи A), у которой A - список утверждений, определяющий текущий рассматриваемый подслучай. Эта задача регистрируется в списке x16, а к началу списка x17 добав-

ляется 0. Затем в начале списка x16 регистрируется задача нормализатора x26, а в начале списка x17 - ссылка на кадр x20. Далее - продолжение просмотра глобального стэка.

Если x20 - кадр оператора, то после "заголовок(x22 3)" - переход через "иначе 1", где переменной x25 присваивается список значений программных переменных, определенных в этом кадре.

Если справочник "очевидно" определяет, что x23 есть заголовок проверочного оператора, присваивая переменной x26 пару (набор входных переменных проверочного оператора - вид проверяемого утверждения), то далее создается задача проверочного оператора x30; она регистрируетсяв списке x16, к началу списка x17 присоединяется ссылка на кадр x20, и продолжение просмотра глобального стэка. Аналогичным образом, используя справочники "синтезатор" и "анализатор", программа усматривает обращения к пакетным операторам этих типов и добавляет в список x16 их обобщенные задачи.

По окончании просмотра глобального стэка в наборе x16 возникает начальная часть цепи обобщенных задач - она предшествует фиктивной задаче, описывающей текущее шаг решения. Дальнейшие действия выполняются ветвью программы, переход к которой осуществляется через оператор "ветвь 2", расположенный после "равно(x18 0)".

Здесь нам понадобится уточнить структуры данных, используемые для сохранения информации о вспомогательных обобщенных задачах, решенных для выполнения текущего приема. При поиске того приема, который будет осуществлять очередной шаг решения, реализуется достаточно большое число вспомогательных процедур, в том числе обобщенных задач. Однако, далеко не все они завершаются срабатыванием приема. Отображение на экране всех таких попыток загромождает картину решения и в обычной ситуации (за исключением некоторых отладочных случаев) нежелательно. С другой стороны, отображение хода решения вспомогательных задач в большинстве случаев необходимо для понимания того, что произошло на очередном шаге. Иногда основная задача вообще сводится к вспомогательной, и на очередном шаге сразу выдается ответ. Так как непосредственно во время решения вспомогательной задачи еще неясно, нужно ли будет ее отображать, приходится сохранять до срабатывания приема информацию о всех таких задачах - чтобы выдать ее после срабатывания приема. Разумеется, можно было бы сохранять здесь вообще полный протокол решения вспомогательных задач, и как только прием сработал, отображать сначала его, а лишь затем само срабатывание приема. Однако, этот вариант технически сложен, и в настоящее время реализован более простой вариант - сохранять лишь информацию об обращениях к вспомогательным обобщенным задачам, а ход их решения отображать при повторном запуске такого решения. Повторный запуск инициируется непосредственно из текущего кадра прерывания, причем на каждом шаге повторного решения можно запускать повторное решение своих вспомогательных задач, и т.д.

Для сохранения информации о вспомогательных задачах служит комментарий (буфер A) к посылкам исходной задачи. Здесь A - накопитель наборов (B_1 B_2 B_3 B_4 B_5), хранящих информацию об отдельных обращениях к решению вспомогательных задач. B_1 есть первая программная переменная, значение которой в момент обращения было не определено. B_2 - копия вспомогательной задачи либо пара (название пакетного оператора - набор значений его входных переменных); B_3 - число на счетчике шагов интерпретатора в момент обращения; B_4 - сначала 0, а затем число на

счетчике шагов в момент выхода. B_5 - сначала 0, а затем результат обращения - ответ задачи, либо выданный нормализатором терм, либо набор значений переменных, полученный синтезатором, либо логический символ "истина" в случае обращения к проверочному оператору, либо символ "отказ". Наборы размещаются в накопителе A в том же порядке, в котором происходили обращения.

Если вспомогательная задача зарегистрирована в комментарии "буфер", а затем произошел откат к точке, предшествующей обращению, то она автоматически исключается из данного комментария. Это делает при откатах интерпретатор ЛОСа, используя номер первой не определенной при обращении переменной B_1 - после отката такой номер почти всегда уменьшается (в особых точках, где при откате номер может сохраниться, используется дополнительная коррекция).

Вернемся к рассмотрению указанной выше ветви программы отладчика. В тех случаях, когда текущий шаг решения заключается в обращении к пакетному оператору либо к решению задачи, отладчик должен принять решение - перевести ли трассировку вглубь процесса решения этой вспомогательной задачи, либо зарегистрировать обращение в комментарии "буфер". Сначала рассматривается случай обращения к пакетному оператору. После проверки того, что отсутствует режим ручного выбора входа в подпроцессы и имеет место обращение к пакетному оператору (x18 = 1), переменной x19 присваивается извлекаемая из набора x16 обобщенная задача пакетного оператора, к которому происходит обращение (с отбрасыванием последней фиктивной задачи разбора подслучаев в нормализаторе, если она есть, - чтобы для каждого очередного подслучая процесс преобразований нормализатора отображался независимо). Если в установке на режим трассировки заблокирован вход в любые подпроцессы (элемент "обращение" в комментарии "трасснабор"), то выход из отладчика. Иначе - переход через "ветвь 2".

При наличии комментария (титр стоп) к пакетному оператору, блокирующему показ обращения к нему - тоже выход из отладчика.

Если пакетный оператор имеет комментарий "обращение", то отладчик осуществляет вход в пошаговый показ его действий. Оператор "трассировка(процедура ...)" переустанавливает для этого трассировку так, чтобы обращение к отладчику происходило в кадре рассматриваемого пакетного оператора при инициализации его действий операторами "контрольнормализации", "контрольбуфера", "синтезатор", "подслучаи", либо при изменении значения переменной x1 (т.е. при изменении обрабатываемого нормализатором терма).

Если же пакетный оператор не имеет комментария "обращения", то осуществляется регистрация в комментарии (буфер ...) факта обращения к нему. Для этого определяется первая не определенная при обращении к оператору переменная x20; переменной x21 присваивается заголовок оператора; переменной x23 - список значений переменных программы оператора, определенных на момент прерывания. Для проверочного оператора и синтезатора, имеющих выходные переменные, в наборе x23 их значения заменяются на логический символ "неопред". Все наборы из x23 заменяются их копиями - чтобы возможные изменения этих наборов при работе оператора не изменили сохраненных первоначальных версий. Для сохранения в комментарии (буфер ...) вводится, согласно указанному выше формату, набор x25 с информацией о данном обращении, и далее - выход из отладчика.

После ветви программы, анализирующей обращение к пакетному оператору, располагается похожая ветвь, анализирующая обращение к вспомогательной задаче. Она начинается с того же оператора "не(существует(x19 и(ключ(комментариипосылок(x14)трасснабор x19(входит(полный конец(x19)))))))", что и предыдущая ветвь

- для отсечения случаев ручного выбора входа в подпроцессы. Затем проверяется, что имеет место установка на прерывания типа 2 (преобразования задачи заданного кадра) либо 17 (действия пакетного оператора), причем заголовок оператора, при обращении к которому произошло обращение к отладчику - "ответзадачи" либо "прямойответ". Если просмотр подпроцессов заблокирован (элемент "обращение" в установке на режим трассировки), то выход из отладчика, иначе - переход через "ветвь 2".

Здесь значение переменной x18 изменяется на 1 (эта переменная, как уже говорилось выше, является индикатором шага решения, заключающегося в обращении к вспомогательной задаче либо к пакетному оператору). Переменной x21 присваивается смещение последнего кадра задачи - это и есть вспомогательная задача, к которой произошло обращение. Находится набор x22, описывающий содержимое кадра x21, и по нему определяется сама задача x23. Если она имеет комментарий либо, в случае задачи на исследование, комментарий посылок (титр стоп), то выход из отладчика. Иначе - переход через оператор "ветвь 1".

Если задача x23 имеет комментарий (в случае задачи на исследование - комментарий посылок) "обращение", то происходит переустановка трассировки на преобразования кадра x21. Иначе - проверяется, что обращение к вспомогательной задаче произошло непосредственно либо через посредство вспомогательных процедур "вспомописание", "быстроприсание", "попыткаспуска", "преобразование", "вспомпреобразование", "следствие" из того кадра x20, по преобразованиям в котором была установлена трассировка. Если это не так, то выход из отладчика. В противном случае определяется первая не определенная перед обращением переменная x24 кадра x20; находится копия x25 исходной версии вспомогательной задачи; эта копия снабжается комментарием посылок (максимальныйуровень ...), сохраняющим значение максимального уровня обращения к задаче; список комментария (буфер ...) пополняется набором x27, характеризующим обращение, и выход из отладчика.

После указанных выше действий, включающих регистрацию в комментарии (буфер ...) данных об обращении, если таковое имело место, - переход к фрагменту, начинающемуся с контрольной точки "прием(7 8)". Здесь начинается заполнение набора x19 информационных элементов, используемого при обращении к оператору "цепьзадач" (выйти на данный фрагмент можно через пункт оглавления программ, в котором упоминается x19). На основе набора x19 будет также создаваться фиктивная задача цепи обобщенных задач, описывающая текущее преобразование. Сначала в x19 заносятся информационные элементы (смкадр x17) и (Текзадача A), где A - вхождение текущей задачи в цепь обобщенных задач x16. Если x18 равно 1 (очередной шаг решения состоит в обращении к вспомогательной задаче либо к пакетному оператору), то в x19 заносится символ "обращение". Далее при x18 = 0 рассматриваются следующие случаи А) - Д), связанные с шагом выдачи ответа задачи либо завершения работы пакетного оператора, а при x18 = 1 - случай Е), связанный с инициализацией разбора случаев в нормализаторе:

А) Случай, когда на текущем шаге был выдан ответ некоторой задачи (либо x7 = 1, т.е. имелась установка на прерывание при выдаче ответа, либо x7 = 2 - установка на прерывание при преобразовании задачи, причем реализуется оператор "ответ").

Проверяется наличие внешнего оператора "редакторответа". Если он есть, то находится первый внешний кадр задачи либо пакетного оператора, предпринимается переустановка трассировки на этот кадр, и выход из отладчика. Сразу после оператора "редакторответа" в программах приемов идет оператор выдачи ответа в том

кадре, на который произошла переустановка трассировки, и отображение на экране полученного ответа будет выполнено при его реализации.

Если внешнего оператора "редакторответа" нет, то из стэка выражений извлекается значение х20 - объект, выданный в качестве ответа. Если х20 - терм либо логический символ, то в х19 заносится информационный элемент (ответ х20); иначе х20 - ответ задачи на исследование, совпадающий с текущей версией этой задачи, и тогда в х19 заносится элемент (ответ исследовать). В обоих случаях предпринимается переустановка трассировки на первый внешний кадр задачи либо пакетного оператора, и далее - откат к оператору "ветвь 1", расположенному перед контрольной точкой "прием(7 9)", где продолжается заполнение набора х19. К этому же оператору происходят откаты по завершении остальных случаев, кроме случая Г).

Б) Случай, когда на текущем шаге выдается результат применения пакетного оператора (переход через "иначе 3" в фрагменте, содержащем контрольную точку "прием(7 8)"). В зависимости от того, как выдается результат (операторы "ответ", "результат", "стоп"либо выход по откату), создается заготовка х20 ответа: символ "отказ" при выходе по оператору "стоп" либо по откату; содержимое последней заполненной ячейки стэка выражений при выходе по оператору "ответ", и константа "истина" при выходе по оператору "результат". Если реализуется пакетный синтезатор, определяющий значения $t_1, \ldots, t_n$ своих выходных переменных $x_1, \ldots, x_n$, то х20 заменяется на терм "и(равно(x_1 t_1) ... равно(x_n t_n))". Затем в накопитель х19 заносится элемент (ответ х20) и предпринимается переустановка трассировки на первый внешний кадр задачи либо пакетного оператора. Специально отслеживается случай попадания на кадр оператора "цепьзадач", которое может произойти, если трассировка происходит при повторном запуске вспомогательной обобщенной задачи. Выход за рамки данного кадра блокируется, и трассировка переустанавливается по тому кадру, источником которого он является.

В) Случай выдачи отказа на задачу по исчерпании средств (текущий уровень становится больше максимального уровня задачи). Тогда в х19 заносится пара (ответ отказ), а трассировка переустанавливается так же, как и выше.

Г) Случай завершения применения пакетного анализатора по оператору "выход". Тогда просто осуществляется выход из отладчика.

Д) Случай завершения решения задачи, сведенной при помощи оператора "попыткаспуска" к некоторой вспомогательной задаче. Здесь обращение к отладчику инициируется оператором "смпрер", размещенным в программе "попыткаспуска" после того, как вспомогательная задача оказалась решенной. При идентификации такого контекста отладчик заносит в набор х19 элемент (попыткаспуска A B), определяющий вхождение A заменявшегося при сведении к вспомогательной задаче условия основной задачи и заменяющее условие B. Убедившись в том, что при обращении к оператору "попыткаспуска" не был введен информационный элемент (титр стоп), блокирующий показ результата обращения, отладчик пересылает в х19 все информационные элементы (титр ...) указанного обращения; с их помощью будет определяться сопровождающий текст. Далее переменной х26 присваивается ответ вспомогательной задачи, и в х19 заносится пара (ответ х26).

Е) Реализуется оператор "подслучаи", инициирующий разбор случаев в пакетном нормализаторе. Определяется утверждение х21, по которому происходит разбор случаев. Если оно не имеет заголовка "или", то выход из отладчика. Иначе - к набору х19 присоединяется элемент (разборслучаев х21).

Далее происходит откат к ветви оператора "ветвь 1", расположенного перед "при-

ем(7 9)". Здесь, после контрольной точки "прием(8 0)", начинается анализ текущего преобразования, если оно отлично от рассматривавшихся выше случаев обращения к вспомогательной задаче (оператору) либо выдачи результата. Переменной x20 присваивается логический символ, определяющий тип преобразования - это либо заголовок оператора, вызвавшего обращение к отладчику, либо, в специальных случаях, название текущей процедуры ("внутрвывод", "внутрпреобр", "контрольнормализации", "контрольтитра", "контрольбуфера"), к которой относится данный оператор.

Сначала рассматривается случай преобразования замены в задаче либо нормализаторе - здесь x20 есть "замена" либо "заменаусловия" либо "заменапосылки". Переменным x21 и x22 присваиваются значения предпоследней и последней входных переменных реализуемого оператора x20, извлекаемые из стэка выражений.

Если x20 = "замена", то переменной x23 присваивается обобщенная задача нормализатора, в которой происходит обращение к оператору x20.

Если заменяющий терм x22 совпадает с текущим термом нормализатора (переход через "ветвь 6"), то выход из отладчика. Если нормализатор допускает режим разбора случаев, причем реализуется его прием с заголовком "окончание" и предпринимается извлечение из комментария (разборслучаев ...) очередного случая для преобразований, то при инициализации новой серии альтернатив в x19 заносится набор (подслучаи ...), определяющий рассматриваемую альтернативу, иначе - выход из отладчика.

В остальных случаях замены в нормализаторе - переход через "ветвь 5". Здесь предпринимается регистрация в x19 информационных элементов (титр ...), определяющих сопровождающий текст замены. Такие элементы создаются приемом нормализатора и присваиваются с помощью вспомогательного оператора "титр" последней определенной перед выполнением замены программной переменной. Если имелся единственный элемент (титр стоп), то выход из отладчика.

Если x20 отлично от символа "замена", то переход через "иначе 4". Здесь располагается ветвь программы, усматривающая случаи, в которых отображение шага замены нецелесообразно, а в остальных случаях пополняющая набор x19 сформированными приемом элементами (титр ...). Прежде всего, блокируется выдача на экран замены утверждения "A - число" на логическую константу "истина". Затем переменной x23 присваивается ссылка на контрольную точку "прием(...)", предшествующую замене.

Сначала рассматривается случай, когда эта точка расположена внутри оператора "заменавхождения". Если ее номер отличен от 27 и 42, то есть замена терма задачи - не основная, реализуемая указанным оператором, а сопровождающая основную, то выход из отладчика. Если оператор "заменавхождения" реализуется внутри другого обращения к оператору "заменавхождения", то в набор x19 заносится элемент (титр 0).

Если контрольная точка x23 расположена внутри программы "равно" и номер ее - 16, 12, 55, либо 57, то реализуется прием, усматривающий в задаче на исследование систему из N уравнений с N числовыми неизвестными ($N = 2, 3, 4$) и обращающийся к вспомогательной задаче для ее решения. Обращение к этому приему на экран не выводится (в случае удачной попытки отображается лишь достигнутый результат) - здесь предпринимается выход из отладчика.

Если контрольная точка x23 расположена внутри программы "и" и номер ее равен 4, то есть реализуется прием замены условия "и($A_1 \dots A_n$)" задачи на описание на группу условий $A_1, \dots, A_n$, то в x19 заносится элемент (Замена 1 A B) , определяющий заменяемый и заменяющий наборы термов A, B.

Наконец, если точка расположена внутри программы "описать" и номер ее равен 34, то применяется прием перехода к новой несущественной неизвестной. При его выполнении реализуется серия замен условий задачи. Чтобы избежать повторений, вся информация о приеме выдается при первой из этих замен (в x19 заносится элемент (Замена ...), а также элемент (сборкатекста ...), определяющий поясняющий текст), а при последующих заменах происходит выход в отладчик.

После рассмотрения подслучаев. связанных с контрольной точкой x23, происходит откат к оператору "ветвь 1", расположенному перед оператором "приемкадра(конец(x5)x23)". Здесь анализируется случай, когда оператор замены терма задачи расположен внутри процедуры "заменавхождения". В случае режима двойного решения задачи с блокировкой показа неиспользованных действий отменяется выдача на экран преобразования посылки, снабженной комментарием "следузел". Затем в x19 переносятся из информационных элементов оператора "заменавхождения" все наборы (сборкатекста ...) и (титр ...), определяющие сопровождающие тексты. Если оказывается, что здесь имеется единственный элемент (титр стоп), то выход из отладчика.

Завершает действия по обработке преобразования замены для задачи либо нормализатора регистрация в наборе x19 элемента (замена A_1 A_2). Здесь A_1 - указатель на преобразуемый терм: 0 для нормализатора и пара (вхождение посылки либо условия - указатель посылки (0) либо условия (1)) для задачи; A_2 - заменяющий терм.

Далее рассматривается случай занесения новой посылки либо условия - x20 = "занесениеусловия" либо x20 = "занесениепосылки". Переменной x22 присваивается ссылка на кадр, в котором предпринимается обращение к оператору x20, а переменной x23 - тип этого кадра. Сначала рассматриваются случаи, в которых отображение преобразования на экране блокируется либо организуется специальным образом; общий случай будет обрабатываться ветвью оператора "ветвь 2".

Прежде всего, рассматривается случай x23 = 1 - кадр задачи. Переменной x25 присваивается набор значений программных переменных в этом кадре задачи, а переменной x26 - значение переменной "x2" этого кадра.

Если значением x26 служит вхождение символа "равно", то определяется ссылка x27 на идущую перед выполнением оператора x20 контрольную точку "прием(...)". Если номер этой точки - 24 либо 56 (приемы перехода к новым неизвестным при решении числовых уравнений и решения уравнения из списка посылок задачи на исследование относительно числовой неизвестной), то выход из отладчика. В этих случаях реализуются группа операторов занесения и оператор удаления; чтобы вся информация о срабатывании приема выдавалась за один шаг, она связывается с оператором удаления. Если номер контрольной точки равен 34, то реализуется прием решения системы уравнений относительно параметров, если она линейна по этим параметрам. Здесь снова реализуется группа операторов занесения и группа операторов удаления. Информация отображается только для первого оператора занесения; при этом в x19 заносится соответствующий элемент (Замена ...).

Если значением x26 служит вхождение символа "и" и контрольная точка указывает на упоминавшийся выше прием замены условия "и($A_1 \dots A_n$)" на группу условий $A_1, \dots, A_n$, то выдача на экран информации о преобразовании - добавлении условия $A_i; i \geq 2$ - блокируется, так как она уже была выдана при применении оператора замены указанного условия на A_1.

Если значением x26 служит вхождение символа "меньше" либо "меньшеилиравно" и номер контрольной точки равен 6, то реализуется прием перехода к новым

неизвестным при решении строгого либо нестрогого неравенства. В этих случаях - выход из отладчика, так как отображение информации о приеме происходит при реализации первого из операторов удаления.

Если x26 равно 0, причем реализуется уже встречавшийся выше прием символа "описать", номер контрольной точки которого равен 34, то выход из отладчика - информация о применении этого приема выдается при выполнении оператора замены.

Далее рассматривается случай x23 = 2 - кадр оператора.

Сначала выделяется подслучай заголовка "вывод" этого оператора. Проверяется, что внутри программы "вывод" реализуется оператор "занесениепосылки". Находится контрольная точка "прием(...)", локализующая последний оператор, и проверяется, что ее номер равен 2 - т.е. выполняется основное преобразование занесения выведенной посылки. Переменной x26 присваивается список значений программных переменных в кадре оператора "вывод". Если выводимое утверждение имеет вид "и($A_1 \ldots A_n$)", то оператор "вывод" последовательно заносит в список посылок утверждения $A_1, \ldots, A_n$. Отображение информации о преобразовании предпринимается только при занесении первого из этих утверждений: в набор x19 добавляются элемент (список ...) и извлекаемые из обращения к оператору "вывод" элементы (титр ...); в остальных случаях - выход из отладчика.

Далее рассматривается случай заголовка "заменагруппы" внешнего оператора. После проверки соответствия значения программной переменной "x2" этого оператора типу выполняемого действия, в x19 заносится элемент (Замена ...), описывающий преобразование замены. К нему добавляются извлекаемые из обращения к оператору "заменагруппы" элементы (титр ...).

Наконец, переход через оператор "ветвь 2", расположенный после оператора "или (равно(x20 занесениеусловия)равно(x20 занесениепосылки))", приводит к ветви, в которой рассматривается общий случай преобразования вывода посылки либо условия. Переменной x21 присваивается извлекаемое из стэка выражений выводимое утверждение. Если оно имеет один из заголовков "актив", "истина", "число", "точка", то выход из отладчика, так как показ данного шага на экране малоинформативен. Аналогично, блокируется показ шага вывода, если он происходит внутри оператора "заменавхождения" (занесение утверждений, сопровождающих основное преобразование замены) либо внутри оператора "одз" при режиме трассировки, отключающем показ действий данного оператора. Далее в набор x19 заносится элемент (вывод ...), описывающий выполняемое преобразование. Если внешний оператор имеет заголовок "вывод" либо "выводусловия", то в x19 заносятся также извлекаемые из обращения к этому оператору элементы (титр ...). В первом случае предварительно проверяется, не имеет ли место режим двойного решения задачи с блокировкой показа неиспользуемых выводов, и если имеет, то используется ли данный вывод. Это делается с помощью комментария (повтор ...), перечисляющего все использованные на первом запуске решения выводы.

Следующий случай - удаление посылки либо условия; x20 = "удалениеусловия" либо x20 = "удалениепосылки". Как и выше, переменной x22 присваивается ссылка на кадр, в котором предпринимается обращение к оператору x20, а переменной x23 - тип этого кадра.

Сначала рассматривается случай кадра задачи - x23 = 1. Переменной x26 присваивается значение переменной "x2" в кадре x22.

Если x26 - вхождение символа "равно", то определяется предшествующая преобразованию контрольная точка "прием(...)". Если ее номер равен 56 (указанный

выше прием решения числового уравнения в посылках задачи на исследование), то в x19 заносится элемент (Замена ...), описывающий выполняемую приемом групповую замену. Аналогичным образом обрабатываются случаи номеров 16,12,55,57,24 (эти приемы также были указаны выше), причем в случае, когда удаляются несколько утверждений, показ преобразования происходит только для первого удаления.

Если x26 - вхождение символа "меньше" либо "меньшеилиравно", а номер контрольной точки "прием(...)" равен 6, то для первого удаления в набор x19 также заносится элемент (Замена ...), а остальные удаления игнорируются.

Если x23 = 2 и внешний оператор имеет заголовок "заменагруппы", то выход из отладчика - в этом случае отображение действий происходит при обращении к оператору занесения посылки либо условия.

Как и при занесении утверждения, общий случай при удалении обрабатывается в ветви оператора "ветвь 2". Блокируется показ удаления утверждения "истина"; кроме того, показ любого удаления (кроме перечисленных выше специальных случаев, где удаление являлось одним из элементарных действий в составе большой процедуры) блокируется, если такая блокировка имеется в установке на режим трассировки. Наконец, показ удаления не происходит, если оно реализуется внутри оператора "заменавхождения" при режиме блокировки показа изменений сопровождающих утверждений. В остальных ситуациях набор x19 пополняется элементом (исключение ...), описывающим преобразование удаления.

Далее идет случай преобразований, изменяющих общие комментарии. Здесь x20 - один из символов "замечание", "примечание", "удалениезамечания", "удалениепримечания". Переменной x21 присваивается последний занесенный в стэк выражений элемент. Для первых двух случаев (ввод комментария) этот элемент есть символьное число операндов оператора x20, для последних двух - удаляемый комментарий. В первых двух случаях переменной x23 присваивается вводимый комментарий (если он представлял собой логический символ, то преобразуется к виду односимвольного набора), после чего происходит переприсвоение переменной x21 значения x23. Затем в набор x19 заносится элемент (замечание ...), определяющий рассматриваемое преобразование комментариев.

Аналогичным образом обрабатывается случай занесения либо удаления комментария к отдельному терму задачи. Здесь x20 - один из символов "замечусловие", "примечпосылки", "удалениепримечпосылки", "удалениезамечусловия".

Если x20 = "внутрвывод", то реализуется прием вывода в пакетном анализаторе. Здесь x19 пополняется элементами (вывод ...) и (титр ...); последние извлекаются из входных данных оператора "внутрвывод". Аналогичным образом, x20 = "внутрпреобр" означает применение приема замены подтерма в анализаторе. Здесь в x19 вместо элемента (вывод ...) заносится элемент (замена ...).

Для дальнейших действий выполняется откат к оператору "ветвь 1", расположенному в том же фрагменте, что и контрольная точка "прием(8 0)". Прежде всего, оператор "поискприема" определяет ссылку x20 на текущий примененный прием. В случае приема ГЕНОЛОГа она представляет собой тройку (логический символ - номер его узла - заголовок приема); в случае приема, реализованного на ЛОСе - одноэлементный набор, образованный номером контрольной точки "прием(...)", предшествующей точке выхода в отладчик. К началу этого одноэлементного набора присоединяется символ, в программе которого расположена контрольная точка. Затем в набор x19 заносится элемент (прием x20).

Наконец, предпринимается добавление к цепи обобщенных задач фиктивной зада-

чи, описывающей текущее действие. Это происходит после контрольной точки "прием(8 2)". Проверяется, что набор x19 содержит элемент, заголовок которого указывает на подлежащее отображению на экране действие, отличное от обращения к вспомогательной задаче либо пакетному оператору - "замечание", "ответ", "замена", "вывод", и т.д. После этого из набора x19 в поднабор x20 отбираются те элементы, которые войдут в описание фиктивной задачи; небольшая коррекция здесь происходит лишь для элемента (ответ ...) в случае проверочного оператора. Далее к концу цепи обобщенных задач x16 присоединяется фиктивная задача (прием x20), и переход через "ветвь 1".

Переменной x20 присваивается вхождение в x16 фиктивной задачи, описывающей текущее действие. В случае шага решения, на котором предпринимается обращение к вспомогательной задаче либо пакетному оператору, переменной x20 оказывается присвоено вхождение в x16 такой новой задачи. Если это действие связано со вводом либо удалением комментариев, то добавление к цепи обобщенных задач тех вспомогательных задач, решение которые ему предшествовало, блокируется. Здесь сразу предпринимается переход через оператор "ветвь 1". Иначе - переход через "ветвь 2" для присоединения к x16 вспомогательных задач.

Инициализируется пустым словом накопитель x21 вспомогательных обобщенных задач, и переменной x22 присваивается комментарий (буфер ...) к посылкам исходной задачи. Последовательно просматриваются наборы x23 из данного комментария, описывающие имевшие место обращения к вспомогательным задачам и пакетным операторам. Переменной x24 присваивается копия вспомогательной задачи либо, для пакетного оператора, пара (заголовок оператора - набор его входных данных).

Сначала рассматривается случай, когда x24 - задача. Если она имеет тип "преобразовать" и ее ответ совпадает с исходным видом преобразуемого терма, то эта задача игнорируется. В противном случае задача x24 снабжается комментарием посылок (буфер A), ссылающимся на вхождение A первого разряда набора x23, и заносится в список x21.

Если x24 - указатель обращения к пакетному оператору, то переменной x25 присваивается его заголовок, а переменной x26 - набор входных данных.

В случае пакетного нормализатора (справочник "быстрпреобр" имеет на заголовке оператора ненулевое значение) проверяется, что результат нормализации отличен от исходного терма. Отбрасывается также, кроме особых случаев, отображение преобразований константного терма и терма, имеющего логические константы "истина", "ложь". Проверяется, что в x21 не было уже обобщенной задачи нормализатора с теми же обрабатываемым термом, посылками и комментариями. Затем создается обобщенная задача нормализатора x27, к комментариям которой добавляется, как и в случае обычных задач, ссылка (буфер A) на набор x23. Предпринимается обращение к справочнику "титр", определяющему элемент (титр ...) для сопровождающего обращение к вспомогательной задаче текста. Однако, этот элемент нигде не регистрируется, а нужен здесь лишь для проверки того, что отображение обращения к нормализатору не заблокировано элементом (титр стоп). После этого x27 заносится в накопитель x21.

В случае проверочного оператора (ненулевое значение справочника "очевидно" на заголовке оператора) из набора x26 входных данных выделяется поднабор x28 значений переменных утверждения, подлежащего проверке. Этот поднабор подставляется вместо самих переменных в шаблон проверяемого утверждения, после чего возникает проверяемое утверждение x29. Отбрасываются случаи, когда это утверждение не имеет свободных переменных либо представляет собой указатель на

тип объекта. Затем создается обобщенная задача проверочного оператора x31, и эта задача заносится в накопитель x21.

Аналогичным образом в накопитель x21 заносятся обобщенные задачи синтезатора и анализатора. По окончании просмотра комментария (буфер ...) к концу списка x16 добавляется содержимое списка x21; значение переменной x20 при этом изменении набора x16 корректируется так, чтобы оно по-прежнему указывало на вхождение фиктивной задачи, описывающей текущее действие. Далее - откат к оператору "ветвь 1", расположенному после "равно(x20 правыйкрай(x16))".

В случае повторного просмотра процесса применения проверочного оператора на экране сначала будет прорисовываться обращение к этому оператору, и лишь затем - кадр фиктивной задачи, поясняющий, каким образом выполнена проверка. Для этого в указанной ситуации переменной x20 переприсваивается вхождение, предшествующее фиктивной задаче.

После контрольной точки "прием(8 4)" идет обращение к процедуре "цепьзадач", реализующей собственно просмотр цепи обобщенных задач x16; эта процедура описывается в следующем подразделе. Если при просмотре цепи задач была нажата клавиша для продолжения трассировки, то оператор "цепьзадач" присваивает выходной переменной x21 набор, содержащий символ "трассировка". В этом случае перед выходом из отладчика выполняются следующие действия:

а) Удаляется комментарий (Смполоса ...) к исходной задаче, в котором содержатся выделения задач и их элементов, сделанные при просмотре цепи задач.

б) Расчищается комментарий (буфер ...), кроме случая приемов, преобразующих комментарии.

в) Удаляется комментарий "обращение" к нормализатору либо задаче, инициировавший вход в трассировку внутри этого нормализатора либо задачи. Чтобы такое удаление не повлияло на работу счетчика шагов в отсутствии трассировки, фактически комментарий "обращение" просто заменяется на другой логический символ - "нормимп", с которым не связано никакой специальной функциональной нагрузки.

г) Восстанавливается активация того информационного блока, который был активирован до обращения к отладчику.

Если же при просмотре цепи задач не была нажата клавиша, продолжающая трассировку, то экран расчищается, и откат к обработчику команд отладчика ЛОСа. В этом случае комментарий (автоклавиатура ...) хранит код команды, обеспечивающей прорисовку при откате текущей программы.

17.2.3 Просмотр и редактирование цепи обобщенных задач

Интерфейс просмотра и редактирования цепи обобщенных задач, созданной отладчиком при анализе текущего шага решения, осуществляется оператором "цепьзадач(x1 x2 x3 x4)". Значением входной переменной x1 служит цепь обобщенных задач; x2 - вхождение в x1 фиктивной задачи, описывающей текущее действие, либо вхождение той обобщенной задачи, к которой происходит обращение; x3 - набор информационных элементов, характеризующих контекст срабатывания приема. Фактически, это формируемый программой отладчика набор x19 из предыдущего подраздела. Список типов элементов в наборе x3 находится в справочной информации для символа "цепьзадач". Оператор "цепьзадач" реализует интерфейс просмотра, а при необходимости и изменения задач набора x1 (вспомогательные задачи изменениям не подлежат). Он позволяет также сохранить выделенные элементы задач в файлах задачника. Выходной переменной x4 присваивается набор информационных элементов, переда-

ваемых отладчику. В частности, наличие в x4 логического символа "трассировка" означает, что после выхода из оператора нужно будет продолжить трассировку решения (т.е. выйти из отладчика).

Создание набора информационных элементов полос изображения обобщенной задачи

Определение набора информационных элементов полос изображения обобщенной задачи осуществляется оператором "элементызадачи(x1 x2 x3)". Этот же оператор использовался и в случае просмотра задач из задачника, причем там были перечислены типы создаваемых им информационных элементов полос изображения. Описание его было отложено до данного момента по той причине, что обобщенные задачи возникли лишь в связи с пошаговым просмотром решения. Значением входной переменной x1 является обобщенная задача; значением x2 - набор элементов, содержащий дополнительную информацию для прорисовки. Выходной переменной x3 присваивается набор информационных элементов полос изображения.

В наборе x2 могут встречаться элементы следующих типов:

1. (ответ A) - A есть ответ на задачу, который должен быть прорисован после ее текста;
2. (отказ A) - A есть утерянный ответ на задачу;
3. (изменение A) - A есть старая версия изменившегося ответа на задачу;
4. (число A) - A есть десятичная запись числа шагов, затраченных на решение задачи;
5. (время A_1 A_2) - A_1 и A_2 суть четверки (часы, минуты, секунды, сотые доли секунды) для обращений к часам на моменты запуска и окончания задачи;
6. (вычитание A) - A есть десятичная запись величины замедления при последнем решении задачи.

Эти элементы вводятся при просмотре задачи из задачника; при пошаговом просмотре решения входной набор x2 пуст.

Войти в просмотр программы "элементызадачи" можно, например, из раздела "Интерфейс просмотра и редактирования задач - Процедура ЭЛЕМЕНТЫЗАДАЧИ" оглавления программ.

Накопителем результата служит переменная x4, которая инициализируется одноэлементным набором, содержащим информационный элемент для прорисовки верхней отделяющей линии задачи.

Сначала рассматривается случай, когда x1 - обычная задача либо заготовка такой задачи (если набор x1 начинается с 0). В случае задачи на выбор хода в шахматах к списку x4 добавляется информационный элемент (шахматы ...), и процедура завершается. Иначе - проверяется наличие комментария (геомредактор ...), и к x4 присоединяется информационный элемент для прорисовки чертежа.

Начиная с контрольной точки "прием(2)", происходит создание элементов для прорисовки посылок задачи. Переменной x6 присваивается список посылок. Если отсутствует комментарий "полный" к посылкам исходной задачи, то из x6 удаляются вспомогательные посылки "актив(...)", "фиктпосылка(...)", логические константы

"истина" и простейшие утверждения о типе значения переменных. Затем - переход через "ветвь 2".

В случае задачи на преобразование, имеющей цель "преобразование" либо "вспомпреобразование", прорисовка посылок блокируется. Такие цели указывают на вспомогательные задачи, посылки которых малоинформативны - здесь нужно в первую очередь показывать преобразуемый терм. В прочих случаях к набору x4 добавляются элементы, определяющие последовательную прорисовку утверждений набора x6. Затем - откат к оператору "ветвь 1".

Если в x4 нет элементов для прорисовки посылок, то вводится элемент (пусто 0) для вертикального 19-пиксельного пробела. Далее начинается определение информационных элементов для прорисовки целевой установки задачи. Если список целей укладывается в рамки одного из стандартных случаев, то для прорисовки целевой установки используется соответствующий текстформульный шаблон. Иначе цели просто перечисляются, будучи предварительно преобразованы к виду термов.

Прежде всего, рассматриваются задачи на описание. Если задача имеет цель "редакция" и непустой список неизвестных X, то к x4 добавляются элементы, определяющие текст "Упростить ответ для X". Если в указанной ситуации список неизвестных пуст, а целями задачи, кроме указанной выше, являются только символы "полный", "прямойответ", то задается текст "Переформулировать условие". Далее - переход через "ветвь 4", где переменной x6 присваивается тройка логических символов "полный", "явное", "прямойответ".

Если x6 включается в список целей, то ищется цель x7, перечисляющая неизвестные. Если такой цели нет, то проверяется, что список целей, кроме набора x6, может иметь лишь цели "одз", "проверка". Тогда задается текст "Определить истинность или ложность". Если же цель x7 имеется, то находится список x8 всех остальных (кроме x6, x7) целей задачи, за исключением ряда целей, наличие либо отсутствие которых не влияет на создание стандартного текста целевой установки. Наличие в x8 символа "учетответа" приводит к созданию текста вида "Редактирование параметрического описания относительно ...".

Если x8 не имеет никаких целей, кроме, быть может, цели "упростить", то проверяется наличие цели (параметры ...). Если она есть и все неизвестные задачи являются несущественными, то создается текст "Определить условие существования для ...". Если же некоторые неизвестные не являются несущественными, то создается текст "Найти ..., для которых существуют ...". При отсутствии цели (параметры ...) создается текст "Найти ...".

Если x8 имеет цель "числрешение", а кроме нее - никаких целей, кроме, быть может, цели "упростить", то создается текст "Найти приближенное значение ...".

Если x8 имеет цель вида (известно ...) и не содержит других целей, кроме, быть может, целей "упростить", "вспомпараметр", то создается текст вида "Найти значения ..." либо "Выразить ... через ...". При наличии цели "вспомпараметр" к нему добавляются слова "Предпочтительно параметрическое описание".

Если x8 имеет цель (связка ...) и не имеет других целей, кроме, быть может, "упростить", то создается текст "Найти ...", в котором перечисляются выражения $y(x)$ для неизвестных функций задачи.

Наконец, если x8 имеет цель "исследовать", то создается один из стандартных текстов для задачи на исследование свойств функции либо (при отсутствии цели "функция") задачи на исследование типа кривой или поверхности.

Если x6 не включается в список целей задачи, то проверяется, что задача имеет

цели "полный", "прямойответ", "пример", и создается текст "Найти пример для ...".

Далее рассматривается случай задачи на преобразование.

Сначала разбирается подслучай, когда задача имеет единственную цель, отличную от целей "упростить", "одз", "видУмножение". В случае цели (рядтейлора ...) к набору x4 присоединяются элементы, образующие текст "Разложить в ряд Тейлора по переменной ...". Аналогичным образом, для цели (рядфурье ...) создается текст "Разложить в ряд Фурье по переменной ..."; для цели (формулатейлора ...) - текст "Разложить в ряд Тейлора по переменной ... до членов степени ..."; для цели (числоценка ...) - текст "Вычислить с точностью до ..."; для целей "разложитьнамножители", "простейшиедроби", "раскрытьскобки" - тексты "Разложить на множители" (при наличии дополнительной цели "видУмножение" - "Разложить на комплексные множители"), "Представить в виде суммы простейших дробей", "Раскрыть скобки".

Затем (переход через "ветвь 2") рассматривается подслучай, когда задача имеет цель "упростить". Если число целей равно 2, то при наличии цели "одз" создается текст "Упростить в о.д.з. выражение ", а при наличии цели "число" - "Вычислить". В случае трех целей - "одз", "класс" и "упростить", - создается текст "Найти явное задание класса". Если, кроме цели "упростить", задача имеет цель "нормИнтеграл", то создается текст "Преобразовать выражение к виду, удобному для интегрирования". В прочих случаях, если имеется цель "упростить", то проверяется, является ли условие задачи утверждением либо выражением. В первом случае создается текст "Упростить утверждение", во втором - "Упростить выражение".

Если задача имеет цель "график", то создается текст "Построить график"; в случае цели "выч" - текст "Найти приближенное значение"; в случае двух целей - "учетрезультата" и "длина" - текст "Упростить утверждение" либо "Упростить выражение".

Для задачи на доказательство к набору x4 добавляется элемент, определяющий текст "Доказать"; для задачи на исследование, имеющей непустой список неизвестных - вводится элемент вертикального 19-пиксельного пробела и элементы, образующие текст "... - не известны".

Если список целей задачи не укладывается ни в один из перечисленных выше стандартных случаев, то предпринимается откат к оператору "ветвь 2" (он расположен в одном фрагменте с контрольной точкой "прием(3)"). Здесь создаются элементы для прорисовки списка целей в виде термов. Прежде всего, проверяется, что x1 не имеет типа "доказать" и не является заготовкой задачи с недоопределенным пока типом. Переменной x6 присваивается список целей; он преобразуется в список x7 термов, обозначающих эти цели. Здесь применяется переход от цели $(A\ b_1 \dots b_n)$ к терму $A(b_1 \dots b_n)$, причем делается это только в тех случаях, когда каждое b_i - символ либо терм (иначе цель в списке x7 не регистрируется). Если список x7 непуст, то в набор x4 вводится элемент для текста "Цели задачи:", после которого идут элементы для термов набора x7. В случае пустого списка x7 после текстового элемента вводится элемент для прорисовки логического символа "пусто".

После прорисовки целевой установки - откат к оператору "ветвь 1", где к набору x4 добавляется вертикальный 19-пиксельный пробел. Затем начинается создание элементов для прорисовки условий задачи.

Если тип задачи - "описать", то переменной x6 присваивается ее список условий. При отсутствии комментария "полный" к посылкам исходной задачи из x6 исключаются вспомогательные либо малоинформативные условия (например, указатели

типа значения неизвестных, кроме указателей "целое", "натуральное"). Затем к x4 добавляются элементы, задающие прорисовку термов списка x6.

В случае задач на доказательство либо преобразование к x4 добавляется элемент, определяющий прорисовку единственного условия. Блокируется прорисовка условия "пустоеслово" задачи на доказательство - это случай недоопределенной задачи, еще не имеющей никакого условия.

Завершают формирование списка x4 информационных элементов полос изображения обычной задачи действия по присоединению в конце ее текста сведений о ранее полученном, утерянном либо изменившемся ответе (согласно входному набору x2); сведений о трудоемкости решения задачи, а также сведений о времени, затраченном на решение. Если набор x4 оказался одноэлементным (пустой бланк задачи), то к нему добавляется элемент вертикального 19-пиксельного пробела - чтобы на экране бланк задачи был достаточно различим.

Далее рассматриваются случаи обобщенных задач.

Для задачи на текстовый анализ (заголовок "текстформ") происходит регистрация в x4 ее текстформульной структуры, после которой, как и в случае обычной задачи, добавляется информация об ответе и времени решения.

В случае задачи нормализатора список посылок x6 обрабатывается, как и для обычной задачи - при отсутствии комментария "полный" из него отбрасываются малоинформативные элементы. Регистрация в списке x4 элементов, обеспечивающих показ термов списка x6, происходит только при наличии комментария "смпосылка" к нормализатору. После отката к оператору "ветвь 2" начинается прорисовка обращения к нормализатору. Переменной x5 присваивается его название. Сначала программа символа x5 справочника "титр" создает по задаче нормализатора x1 комментарий (титр ...), определяющий текстформульный фрагмент, объясняющий выполняемое нормализатором преобразование. Затем процедура "тексттитра" определяет по этому комментарию (он присвоен переменной x6) текстформульную структуру x7 данного фрагмента, и в x4 заносится элемент полосы прорисовки структуры x7. Для создания текстформульных заготовок, объясняющих действия нормализатора A, нужно ввести на ГЕНОЛОГе прием справочника "титр" с "теоремой" данного приема "титр(A x1)", а текстформульные шаблоны связать с описанием этого приема. Если прием справочника "титр" для нормализатора A не создан, либо в нем не предусмотрен шаблон для текущей ситуации, то в x4 заносятся элементы, создающие текст "Применить нормализатор "A" к выражению:". После прорисовки обращения к нормализатору в набор x4 вводится элемент, определяющий преобразуемый терм.

Случаи задач проверочного оператора и синтезатора аналогичны случаю нормализатора. Если справочник "титр" для этих операторов не определяет специальное текстформульное сопровождение обращения к ним, то в x4 заносятся элементы, создающие текст "Проверить утверждение" для проверочного оператора и "Подобрать значение переменной" для синтезатора.

Для задачи анализатора A сначала предпринимается попытка определить текстформульное сопровождение обращения к нему при помощи справочника "титр"; если это не удается, то выдается стандартный текст "Применить анализатор A". Далее в x4 заносится элемент вертикального пробела, и за ним - элементы для прорисовки утверждений, выведенных в процессе применения анализатора.

В случае фиктивной задачи, описывающей текущий шаг решения (заголовок ее - логический символ "прием"), прежде всего проверяется, является ли текущая обобщенная задача обычной задачей и имеет ли она сопровождающий чертеж; если имеет,

то в x4 заносится элемент, определяющий прорисовку этого чертежа. Затем рассматриваются следующие типы выполняемых на текущем шаге действий:

1. Преобразование замены. В списке информационных элементов задачи x1 находится набор (замена A_1 A_2). Если A_1 - набор, то имеет место замена условия либо посылки задачи, причем A_1 - пара (вхождение заменяемого терма задачи - указатель посылки (0) либо условия (1)); A_2 - заменяющий терм. Тогда в x4 заносятся элементы для выдачи текста "Посылка (либо условие) ... заменяется на ...". Если A_1 - терм, то имеет место замена в анализаторе этого терма на терм A_2. В x4 заносятся элементы для текста "... заменяется на ...". Наконец, если A_1 равно 0, то имеет место замена в нормализаторе, и создается текст "Условие ... заменяется на ...".

2. Замена группы утверждений. В x1 находится элемент (Замена A_1 A_2 A_3). Здесь A_1 - указатель посылок (0) либо условий (1); A_2 - набор заменяемых утверждений; A_3 - набор заменяющих утверждений. Создается текст "Условия (либо посылки; условие; посылка) ... заменяются (заменяется) на ...".

3. Вывод новой посылки либо условия. В x1 находится элемент (вывод A_1 A_2). Здесь A_1 - указатель посылки либо условия; A_2 - выводимое утверждение. Создается текст "Добавление посылки (условия) ...".

4. Вывод группы новых посылок. В x1 находится элемент (список A), где A - выводимые посылки. Создается текст "Добавление посылок ...".

5. Попытка замены условия задачи на описание либо доказательство на новое условие, следствием которого оно является. В x1 находится элемент (попытка-спуска A_1 A_2). Здесь A_1 - вхождение заменяемого условия; A_2 - заменяющее условие. Создается текст "Для получения условия ... реализуем далее условие ...".

6. Удаление условия либо посылки задачи. В x1 находится элемент (исключение A_1 A_2). Здесь A_1 - вхождение удаляемого утверждения; A_2 - указатель посылки либо условия. Созадется текст "Удаление условия (посылки) ...".

7. Выдача ответа. В x1 находится элемент (ответ A), где A - ответ либо (в случае положительного результата применения проверочного оператора) символ "легковидеть" либо (при выдаче ответа задачи на исследование) символ "исследовать". Пополнение списка x4 предпринимается только в первой ситуации. Если ответ распадается на подслучаи, то для организации прорисовки схемы этих подслучаев используются специальные операторы "случаи" (для задач на описание) и "варианты" (для задач на преобразование). Затем создается текст "Ответ: ...".

8. Ввод либо удаление комментария. В x1 находится элемент (замечание A_1 A_2 A_3 A_4). Здесь A_1 - указатель ввода (1) либо удаления (0); A_2 - указатель посылок либо условий; A_3 - 0 в случае общего комментария либо тот терм, к которому относится комментарий; A_4 - сам комментарий. Переменной x8 присваивается логический символ, определяющий текст для названия выполняемой операции - "Удаляется комментарий посылок"; "Удаляется комментарий"; "Вводится комментарий посылок"; "Вводится комментарий". Для прорисовки комментария

применяется текстформульная структура; те разряды комментария, которые представляют собой символы, выдаются непосредственно, а прочие - заменяются на символ "в" (вхождение) либо "н" (набор).

9. Инициализация разбора случаев в нормализаторе. В x1 находится элемент (разборслучаев A). Здесь A - утверждение "или(...)", определяющее рассматриваемые подслучаи. Создается текст "Будем рассматривать альтернативные случаи .... Пусть сначала выполнено ...".

10. Переход к рассмотрению очередного подслучая в нормализаторе. В x1 находится элемент (подслучаи A). Здесь A - набор утверждений, конъюнкция которых определяет новый подслучай. Создается текст "Переход к рассмотрению подслучая ...".

По завершении приведенного выше рассмотрения вариантов - откат к оператору "ветвь 1", где предпринимается создание элементов поясняющего текста, расположенного после описания выполненного действия. Инициализируется нулем переменная x7, которая используется в качестве указателя наличия специальных пояснений. Такие пояснения определяются элементами (титр ...) из x1. Если последний разряд такого элемента отличен от 0, то оператор "текстtitра" преобразует его в текстформульную структуру x9, регистрируемую в наборе x4. Если же последний элемент равен 0, то создается текст "(Простейшая стандартизация)". Специальные пояснения могут определяться также элементами (сборкатекста ...) из x1. Отличие их от элементов (титр ...) состоит в том, что первые используют для создания текста текстовые заготовки из 2-го информационного блока, закрепленные за различными логическими символами, а вторые - текстформульные шаблоны, связанные с приемами. Преобразование элементов (сборкатекста ...) в текстформульную структуру осуществляется оператором "сборкатекста".

Если специальные пояснения имелись, то x7 заменяется на 1, и тогда достижимый через "ветвь 3" оператор "обрыв" отменяет откат к оператору "ветвь 2". Иначе - откат к "ветвь 2". Здесь в качестве поясняющего текста вводится текст того концевого пункта оглавления базы приемов, к которому относится применяемый прием. Отбрасываются случаи, когда пояснения представляются избыточными (преобразования, изменяющие комментарии, и т.п.).

Далее - к концу набора x4 добавляется элемент вертикального 19-пиксельного пробела, и этот набор выдается в качестве результата обращения к оператору "элементызадачи".

Однако, в случае пошагового показа решения нужно не просто отображение задачи в цепи обобщенных задач, но еще и пояснение того, с какой целью она введена. Эти и некоторые другие пояснения создаются оператором "инфзадачи(x1 x2 x3 x4)", обращение к которому предпринимается сразу после получения от оператора "элементызадачи" заготовки x3 набора информационных элементов полос изображения. Значением x1 является здесь вхождение рассматриваемой задачи в цепь обобщенных задач; x2 - набор информационных элементов, характеризующих контекст срабатывания приема (т.е. набор x19 отладчика). Выходной переменной x4 присваивается окончательная версия набора информационных элементов полос изображения. Оператор "инфзадачи" извлекает дополнительную информацию для пояснений из анализа стэковых кадров интерпретатора ЛОСа, начатого отладчиком ЛОСа при формировании цепи обобщенных задач.

Выйти в начало программы оператора "Инфзадачи" можно, например, через пункт "Интерфейс просмотра и редактирования задач - Процедура ЦЕПЬЗАДАЧ - Оператор ИНФЗАДАЧИ" оглавления программ.

Для вставки пояснений о внешнем приеме, обратившемся к обобщенной задаче, прежде всего проверяется, что заголовок этой обобщенной задачи отличен от символов "разборслучаев", "подслучаи", "прием". Затем переменной x5 присваивается цепь обобщенных задач, а переменной x6 - элемент (смкадр A), где A - набор ссылок на стэковые кадры для обобщенных задач цепи x5, предшествующих фиктивной задаче, хранящей информацию о текущем действии. Проверяется, что вхождение x1 расположено в цепи задач до указанной фиктивной задачи, и переменной x7 присваивается ссылка на стэковый кадр рассматриваемой задачи. Оператор "приемкадра" находит ссылку x8 на прием, реализация которого начата в кадре x7. Напомним, что такая ссылка - либо тройка (логический символ - номер узла статьи этого символа - заголовок приема), определяющая прием ГЕНОЛОГа, либо пара (логический символ - номер контрольной точки "прием(...)" программы этого символа) в случае приема, реализованного непосредственно на ЛОСе. Для ссылки второго типа - x8 есть пара (s, N) - рассматриваются следующие случаи:

1. s = "или"; $N = 33$. Это - прием разбора случаев по дизъюнктивному условию задачи на описание. Определяется набор x11 значений программных переменных того кадра, в котором происходит реализация приема. Переменной x12 присваивается значение переменной "x16" в наборе x11; этим значением является тот дизъюнктивный член условия, который определяет текущий подслучай. Затем создается текстформульная структура x13 для поясняющего текста "Рассмотрение подслучая: ...", и информационный элемент полосы изображения для этой текстформульной структуры вставляется в набор x3 непосредственно после первого элемента, задающего верхнюю отделяющую линию задачи.

2. s = "или"; $N = 24$. Это - прием разбора случаев по дизъюнктивной посылке задачи на описание. Отличие от предыдущего пункта здесь лишь в том, что дизъюнктивный член текущего подслучая определяется не переменной "x16", а переменной "x9"из набора x11.

3. s = "или"; $N = 38$. Это - прием разбора случаев по дизъюнктивной посылке задачи на доказательство либо на преобразование. Аналогично предыдущим двум случаев, но текущий дизъюнктивный член определяется переменной "x8".

4. s = "равно"; $N = 48$ либо $N = 24$. Это - приемы решения числовых уравнения путем перехода к новым неизвестным, инициируемые в различных ситуациях (первый - в задаче на исследование, второй - в задаче на описание). Переменной x11 присваивается набор значений переменных в кадре приема; переменной x12 присваивается значение переменной "x22" в первом случае и "x20" - во втором. Это значение представляет собой четверку (набор уравнений, в которых предпринимается замена неизвестных - набор выражений с неизвестными, для которых вводятся новые неизвестные - набор уравнений после перехода к новым неизвестным - список новых неизвестных). Переменной x13 присваивается список новых неизвестных, а переменной x14 - набор обозначаемых ими выражений. Затем создается текстформульная структура x15 для текста "Вводим вспомогательные неизвестные ... для выражений ...", которая вставляется в набор x3.

5. s = "меньше" либо s = "меньшеилиравно"; $N = 6$. Это - прием перехода к новым неизвестным при решении строгого либо нестрогого неравенства. Создается текстформульная структура, аналогичная предыдущей.

В остальных случаях (кроме обращения к задаче из оператора "списокзадач", которое происходит при первоначальном запуске решения) в качестве пояснения того, откуда возникла задача, используется текст того концевого пункта оглавления базы приемов либо оглавления программ, в котором зарегистрирован прием. Текстформульная структура для него создается оператором "титрприема".

В том случае, когда рассматривается фиктивная задача, описывающая текущее действие, предпринимается переход из корневого фрагмента оператора "инфзадачи" через оператор "иначе 3". Если в фиктивной задаче содержится указание на то, что предпринимается выход из задачи на исследование (элемент (ответ исследовать)) либо получен положительный результат при обращении к проверочному оператору (элемент (ответ легковидеть)), причем не предусмотрено специального поясняющего текста (отсутствует элемент (титр ...), то находится ссылка x6 на прием, выполнение которого привело к данному действию. Если этот прием определяется парой ("равно", 14), то-есть имеет место возвращение к условиям задачи на описание после того, как анализ объединенного списка условий и посылок привел к явному выражению для одной из неизвестных, то создается текстформульная структура для текста "Возвращение к условиям исходной задачи после получения выражения ... для неизвестной ...". В прочих случаях для пояснения действия используется текст того концевого пункта оглавления приемов либо программ, в котором зарегистрирован примененный прием.

Для фиктивной задачи, указывающей на найденный результат обращения к пакетному синтезатору, в качестве пояснения также используется текст соответствующего концевого пункта оглавления приемов либо программ.

После всех перечисленных выше пунктов - откат к оператору "ветвь 2", где рассматривается случай вспомогательной задачи, расположенной в цепи обобщенных задач после фиктивной. Напомним, что пока информационные элементы полос изображения вспомогательной задачи не содержат данных об ответе, полученном при ее решении. Для нахождения этих данных переменной x6 присваивается комментарий (буфер A) к посылкам вспомогательной задачи (для обобщенной задачи пакетного оператора - просто комментарий этого оператора). Здесь A - вхождение в набор B комментария (буфер B) к посылкам исходной задачи той пятерки, которая сохраняет информацию о решении данной вспомогательной задачи.

Однако, перед добавлением элементов прорисовки ответа, анализируется источник данной вспомогательной задачи, и после верхней разделяющей линии вставляются необходимые пояснения о цели ее решения. Для этого сначала проверяется наличие комментариев (титр ...), сформированных при обращении к задаче использующим ее приемом; если они есть, то оператор "тексттитра" создает текстформульную структуру x8 поясняющего текста и вставляет ее в набор x3. Если таких комментариев нет, но есть комментарий (сборкатекста ...), то текстформульная структура x8 создается оператором "сборкатекста". Если список информационных элементов x2 имеет элемент (попыткаспуска ...), то создается элемент для текста "(Попытка решить задачу после перехода к новому условию)". В остальных случаях вставляется пояснение "(Вспомогательная задача)".

Далее - откат к оператору "ветвь 2", где предпринимаются действия по регистрации в x3 ответа задачи. Проверяется, что рассматриваемая обобщенная задача не

имеет типа "исследовать" и не является обращением к пакетному анализатору. После этого переменной x7 присваивается ответ задачи - он хранится в последнем разряде той пятерки, вхождение v которой определяется парой (буфер v), присвоенной ранее переменной x6. Если рассматриваемая обобщенная задача является обращением к проверочному оператору и x7 отлично от символа "отказ", то оно заменяется на утверждение "истина". Если обобщенная задача является обращением к синтезатору, то x7 - набор термов $t_1, \ldots, t_n$, найденных в качестве значений для переменных $x_1, \ldots, x_n$. Тогда x7 заменяется на утверждение "и(равно(x_1 t_1) ... равно(x_n t_n))". Далее x7 обрабатывается для выдачи в стандартном виде оператором "нормформ" (в частности, здесь нормализуется упорядочение слагаемых и сомножителей). Если ответ представляет собой перечисление подслучаев, то информационные элементы для отображения схемы этих подслучаев создаются операторами "случаи" (для задач на описание) либо "варианты" (для задач на преобразование и нормализаторов). Перед информационными элементами, обеспечивающими собственно прорисовку ответа, вставляются элементы для текста "Ответ:". После этого - выход из оператора "инфзадачи".

Прорисовка цепи обобщенных задач

Оператор "цепьзадач" начинает свою работу с активации второго информационного блока и прорисовки в верхней части экрана меню просмотра цепи задач. Далее идет оператор "повторение", к которому будут происходить откаты при восстановлении исходного изображения просмотра цепи задач. Инициализируется пустым словом накопитель x5 пар (вхождение обобщенной задачи в цепь x1 обобщенных задач - набор информационных элементов полос изображения данной задачи). В x5 заносится пара для задачи, указанной по вхождению x2 в цепь задач - это либо фиктивная задача, поясняющая текущее действие, либо новая задача, в обращении к которой состоит это действие. Для этого применяются операторы "элементызадачи" и "инфзадачи". Затем - переход через "ветвь 2".

Здесь располагается оператор "повторение", к которому будут происходить откаты при перерисовке цепи задач согласно набору x5. Переменной x7 - указателю вхождения в x5 текущей прорисовываемой задачи - присваивается левый край набора x5. Инициализируется нулем номер x8 первой свободной строки. Инициализируется нулем переменная x9, которая будет использоваться для указания на вхождение текущего прорисовываемого информационного элемента полосы изображения.

После контрольной точки "прием(2 9)" располагается ветвь, обеспечивающая выделение голубым цветом отличающихся частей заменяемого и заменяющего термов при преобразовании замены. Для идентификации такого преобразования среди информационных элементов полос изображения первой пары набора x5 ищется элемент (слово смисточник), определяющий прорисовку текста "заменяется на:". Предшествующий и последующий элементы (терм ...) определяют, соответственно, заменяемый и заменяющий термы задачи; эти элементы присваиваются переменным x15 и x16. Переменной x17 присваивается пара (вхождение корня заменяемого терма - вхождение корня заменяющего терма). Далее предпринимается цикл коррекций пары x17. Если на очередном шаге первый и второй элементы этой пары определяют вхождения термов $f(A_1 \ldots A_n)$ и $f(B_1 \ldots B_n)$, имеющих одинаковые заголовки и одинаковое число операндов, причем для некоторого i термы A_i, B_i различаются, а при всех $j \neq i$ термы A_j, B_j совпадают, то x17 заменяется на пару вхождений A_i, B_i. По окончании цикла коррекций - переход через оператор "ветвь 3", где проверяется, что

x17 выделяет два подтерма, отличных от самих термов. Для выделения подтермов используется комментарий (Смполоса ...) к посылкам исходной задачи, аналогичный комментарию (смполоса ...), выделявшему элементы и задачи при просмотре задач в задачнике. Выделенные таким комментарием задачи и их элементы будут прорисовываться в цепи обобщенных задач голубым цветом.

Для дальнейшего определим формат комментария (Смполоса A). Здесь A - набор выделений в цепи обобщенных задач, имеющих один из следующих типов:

1. (B) - B есть вхождение выделенной задачи в цепь обобщенных задач;
2. (B_1 B_2 B_3) - B_1 есть вхождение задачи, в которой выделена полоса; B_2 - номер вхождения выделенной полосы в список полос изображения задачи; B_3 - информационный элемент, определяющий эту полосу;
3. (B_1 B_2 B_3 B_4) - $B_1 - B_3$ те же, что и выше; B_4 имеет вид (набор ...), а B_4 определяет номер вхождения выделенного элемента полосы изображения.
4. (B_1 B_2 B_3 B_4) - $B_1 - B_3$ те же, что и выше; B_3 имеет вид (терм ...), а B_4 - указатель вхождения в этот терм выделенного подтерма.
5. (B_1 B_2 B_3 B_4 B_5) - $B_1 - B_4$ те же, что и выше для случая B_3 вида (набор ...); выделен элемент полосы - терм, и B_5 - указатель вхождения в этот терм выделенного подтерма.

По завершении выделения различающихся фрагментов заменяемого и заменяющего термов - откат к оператору "ветвь 1". Здесь размещается оператор "повторение", к которому происходят откаты при дорисовке.

Переменной x9, если она равна 0 (экран пуст), присваивается вхождение первого информационного элемента полосы изображения текущей прорисовываемой задачи. Если x9 было не равно 0, то определяло вхождение последнего прорисованного элемента. Если этот элемент не последний в своем списке, то x9 переустанавливается на вхождение следующего элемента. Если же он последний, то проверяется, является ли вхождение x7 последним в наборе x5. Если не последнее, то x7 присваивается вхождение следующей пары набора x5, а x9 - вхождение первого информационного элемента полосы изображения задачи новой пары. Если вхождение x7 было последним, то рассматривается соответствующее ему вхождение x12 обобщенной задачи в цепь обобщенных задач. Если x12 - не последнее, то берется следующее за ним вхождение x13. Для обобщенной задачи по вхождению x13 операторы "элементызадачи", "инфзадачи" определяют набор x15 информационных элементов полос изображения; пара (x13 x15) заносится в конец набора x5, после чего x7 и x9 переустанавливаются на прорисовку данной обобщенной задачи. После перечисленных действий по коррекции ссылки x9 на текущий информационный элемент прорисовки - откат к оператору "ветвь 1", расположенному после "повторение".

Если информационный элемент по x9 не прорисован (его последний разряд равен 0), то оператор "высотаполосы" корректирует этот элемент, доопределяя все необходимые для прорисовки параметры, и присваивая переменной x10 высоту его полосы. Если места для прорисовки оказывается недостаточно, то последний разряд информационного элемента обратно устанавливается на 0. Иначе - оператор "Смполоса" выполняет прорисовку. Корректируется номер x8 первой свободной строки, и откат к переустановке x9 на следующий элемент. По завершении цикла дорисовки - переход через "ветвь 1" в начале рассматриваемого фрагмента.

Если прорисовка оборвалась не на последнем элементе набора x5, то этот набор укорачивается - из него исключаются пары, идущие после вхождения x7. Аналогично, от начала набора x5 отбрасываются пары, соответствующие задачам, у которых не прорисована ни одна полоса. Затем - переход к циклу обработки команд интерфейса просмотра цепи обобщенных задач.

Обработка команд интерфейса просмотра цепи обобщенных задач

Выйти на начальную точку обработчика команд можно через пункт "Цикл обращений к клавиатуре - Исходная точка" подраздела оператора "цепьзадач" в оглавлении программ. После оператора "повторение", к которому происходят откаты к запросу очередной команды, расположен оператор "автоменю(x10)", запрашивающий новую команду x10. Рассматриваются следующие типы команд:

1. x10 = "Плюс". Нажата клавиша "Esc" для обрыва решения задачи. Логический терминал "метка" корневого указателя второго информационного блока корректируется так, чтобы после внутреннего перезапуска был выполнен выход на просмотр того концевого пункта оглавления задачника, в котором находится решавшаяся задача. Затем - внутренний перезапуск. Отдельно обрабатывается случай обрыва вывода следствий в базе теорем, где перед внутренним перезапуском активируется 6-й информационный блок.

2. x10 = "частичныйответ" либо x10 = "нормпроизведениевсех". Нажата клавиша "о" либо "ф" для прорисовки текущего реализуемого фрагмента программы и перехода в отладчик ЛОСа. Клавиша "о" вызывает прорисовку фрагмента лишь до текущего оператора, клавиша "ф" - полную прорисовку. Сбрасывается массив текстов - в нем сохранялись текстовые фрагменты текстформульных структур информационных элементов полос изображения, и его нужно периодически расчищать во избежание переполнений. Фактически данная команда через комментарий (автоклавиатура ...) передается отладчику ЛОСа, где она обрабатывается так, как было описано выше. Перед выходом в отладчик ЛОСа удаляется, если он имелся, комментарий "цепьзадач" к посылкам исходной задачи. Этот комментарий был нужен, если велась техническая трассировка на уровне отладчика ЛОСа, и требовалось просмотреть текущую цепь задач.

3. x10 = "подстановка". Нажата клавиша "курсор вниз" для прокрутки изображения вниз. Эта и последующие операции прокрутки в операторе "цепьзадач" реализованы практически так же, как в операторе "списокзадач". Отличие состоит лишь в том, что вместо переменной x4 оператора "списокзадач", определявшей набор троек (номер задачи в списке - задача - список информационных элементов полос изображения), берется переменная x5 для набора пар (вхождение задачи в цепь обобщенных задач - список информационных элементов полос изображения).

4. x10 = "внешсумма". Нажата клавиша "курсор вверх" для прокрутки изображения вверх. Аналогично оператору "списокзадач".

5. x10 = "точкапривязки". Нажата клавиша "PageDn" для прокрутки изображения на одну страницу вниз. Аналогично оператору "списокзадач".

6. x10 = "контрольунификации". Нажата клавиша "PageUp" для прокрутки изображения на одну страницу вверх. Аналогично оператору "списокзадач".

7. x10 = "мышь". Нажата кнопка мыши для выделения задачи либо ее элемента. Аналогично оператору "списокзадач", однако при выделении вместо комментария (смполоса ...) используется комментарий (Смполоса ...).

8. x10 = "обл". Нажата клавиша пробела для отмены выделений. Аналогично оператору "списокзадач", с переходом от комментария (смполоса ...) к комментарию (Смполоса ...).

9. x10 = "отрезок". Нажата клавиша Ctr-PageDown для перехода к просмотру обобщенной задачи, описывающей текущее действие. Если прорисовка начата с задачи, отличающейся от расположенной по вхождению x2 в цепь обобщенных задач, либо прорисован лишь конец этой задачи, то набор x5 повторно инициализируется парой, задающей прорисовку задачи по x2, и откат к перерисовке.

10. x10 = "десчастное". Нажата клавиша Ctr-PageUp для перехода к просмотру первой задачи цепи обобщенных задач. Аналогично предыдущему, но x5 инициализируется для прорисовки начиная с первой задачи цепи.

11. x10 = "модули". Нажата клавиша Ctr-Del для удаления выделенной посылки либо выделенного условия задачи на описание. Это удаление затрагивает только текущий процесс трассировки и используется в отладочных целях. Находится комментарий (Смполоса ...) к посылкам исходной задачи, который сразу же исключается. Проверяется, что он определяет выделение единственного элемента - полосы изображения некоторой задачи, содержащей терм этой задачи. Переменной x18 присваивается вхождение в список x17 информационных элементов полос изображения указанной задачи выделенного элемента x19. Проверяется, что x19 определяет прорисовку терма. Переменной x20 присваивается задача, в которой выделен терм. Переменная x22 перечисляет вхождения данного терма в список посылок задачи, либо, если задача - на описание, в ее список условий. При этом переменной x23 передается значение 0 в случае списка посылок и 1 в случае условий. Проверяется, что выделенная полоса соответствует значению x23 (по наличию либо отсутствию предшествующих выделенному элементов типа "набор" либо "слово"). Затем предпринимается удаление в задаче x20 найденного вхождения x22 посылки либо условия. Корректируется изображение на экране: следующие после удаленной полосы смещаются вверх и корректируются их информационные элементы. Уменьшается значение x8 первой свободной строки; изменяется для дорисовки значение x9 (вхождение того информационного элемента полосы изображения, с которого начинается дорисовка), и откат к дорисовке.

12. x10 = "прогрпеременная". Нажата клавиша "ы" для включения либо выключения режима полного просмотра списков посылок и условий (без пропусков утверждений сопровождающего характера). Как и в случае оператора "списокзадач", это достигается вводом либо удалением комментария "полный" к посылкам исходной задачи. Каждый раз происходит повторная инициализация набора x5, в который заносится пара для задачи, отображающей текущее действие, и далее - откат к перерисовке.

13. x10 = "Неизвестная" либо x10 = "число". Соответственно, нажаты клавиша "Ф" либо "Т" для ввода нового условия либо новой посылки, соответственно, формульным либо текстовым редактором. Как и в случае удаления, команда

используется для ручной коррекции действий решателя при отладке. Сначала переменной x11 присваивается последняя из пар набора x5, а переменной x12 - вхождение в цепь обобщенных задач определяемой этой парой задачи. Переменная x13 инициализируется нулем. Далее находится комментарий (Смполоса ...) и просматриваются наборы x17 из этого комментария, определяющие выделения задач и их элементов.

Если встречается одноэлементный набор x17, т.е. выделена некоторая задача, то переменной x18 присваивается ее вхождение в цепь обобщенных задач, а переменной x19 - соответствующая задаче пара из набора x5. Проверяется, что выделенная обобщенная задача - обычная и что последняя из ее полос изображения прорисована на экране. После этого x11 заменяется на пару x19, x12 - на вхождение x18, а x13 - на номер строки, начиная с которой нужно вводить дополнительный терм в конце выделенной задачи. Для дальнейших действий по добавлению терма в данном случае - откат к оператору "ветвь 2".

Если встречается набор x17 длины 3, т.е. выделена полоса некоторой задачи, то проверяется, что отсутствует выделенная задача. Как и в предыдущем случае, переменной x18 присваивается вхождение рассматриваемой задачи в цепь обобщенных задач, а переменной x19 - ее пара из набора x5. Проверяется, что эта задача - обычная. Переменной x20 присваивается вхождение информационного элемента выделенной полосы, а переменной x21 - сам этот элемент. Проверяется, что полоса прорисована на экране и что заголовок элемента x21 - один из символов "терм", "набор", "слово". В первом случае добавление нового терма происходит внутри списка посылок либо условий, во втором - в конце этого списка. Находится число x24 предшествующих выделенной полосе полос изображения той же задачи, определяемых элементами типов "набор", "слово". Если x24 равно 0, то вставка происходит в список посылок, иначе - в список условий, и тогда проверяется, что тип задачи - "описать". При вставке в конце списка выделение сохраняется, но номер выделенной полосы увеличивается на 1, иначе - выделения сбрасываются. Экран расчищается начиная с выделенной полосы, и все информационные элементы последующих полос помечаются как непрорисованные. x7, x9 соответствующим образом корректируются для дорисовки.

Далее вводится (в зависимости от команды, формульным либо текстовым редактором) новый терм, и переменной x30 присваивается вхождение в задачу списка посылок при x24 = 0 и списка условий при x24 = 1. В этом списке локализуется точка вставки нового терма: если была выделена полоса, содержащая терм T, то находится вхождение того же терма T, имеющее тот же порядковый номер, что и в списке информационных элементов полос. Затем предпринимается собственно вставка нового терма в список по x30, причем два соседних справа списка - веса и комментарии - изменяются соответствующим образом. После изменения задачи предпринимается занесение в список информационных элементов полос изображения пары x19 нового элемента для добавленного терма, и откат к дорисовке.

Если, была выделена целая задача (см. выше), то после отката к оператору "ветвь 2" предпринимаются действия по вводу нового терма, аналогичные только что описанным.

14. x10 = "транслвыражения" либо x10 = "кортеж". Нажата клавиша "Ctr-ф" ли-

бо "Ctr-т" для изменения выделенного терма задачи формульным либо текстовым редактором. Операция используется при отладке. Прежде всего, находится комментарий (Смполоса ...), и проверяется, что он выделяет единственный элемент - полосу изображения, определяемую информационным элементом x15 типа "терм". Проверяется, что эта полоса прорисована на экране. Находится пара x17 набора x5, определяющая задачу, в которой выделена полоса. Переменной x18 присваивается набор информационных элементов полос изображения данной задачи; переменной x19 - вхождение в x18 элемента выделенной полосы. Переменной x20 присваивается номер строки, начиная с которой будет происходить ввод новой версии терма. Обычным образом определяется указатель x21 принадлежности изменяемого терма посылкам (0) либо условиям (1). Выделения сбрасываются; экран расчищается начиная со строки x20. Дальнейшие действия аналогичны случаю добавления нового терма, но вместо вставки происходит замена.

15. x10 = "подобл". Нажата клавиша "ч" для ввода либо изменения чертежа. Эта операция - тоже отладочная; в обычном режиме пошагового решения чертеж корректируется автоматически. Проверяется, что выделена единственная задача, отделяющая линия которой прорисована на экране, и переменной x11 присваивается пара x17 из набора x5, определяющая данную задачу. Переменной x12 присваивается выделенная задача.

    Если имеется информационный элемент (геомредактор ...) ее полос изображения, то он присваивается переменной x13. Затем переменной x14 присваивается тройка (описание чертежа в формате оператора "геомредактор" - верхняя строка рамки чертежа - нижняя строка рамки). Предпринимается обращение к оператору "геомредактор" для изменения чертежа; в x14 корректируются верхняя и нижняя строки рамки; корректируется комментарий (геомредактор ...) к посылкам задачи x12, и откат к перерисовке.

    При отсутствии элемента (геомредактор ...) - аналогичные действия для ввода чертежа.

16. x10 = "верхняяоценка". Нажата клавища "ц" для изменения целевой установки с помощью оглавления типов целевых установок. Снова отладочная операция; реализация ее аналогична случаю процедуры "списокзадач".

17. x10 = "помощь". Нажата клавиша "Ctr-ц" для изменения списка целей через текстовый редактор. Отладочная операция, реализованная так же, как в процедуре "списокзадач".

18. x10 = "внешдизъюнкция". Нажата клавиша "Insert" для вставки набора выделенных термов. Редко используемая отладочная операция. Термы выделяются как в цепи обобщенных задач, так и в задачнике (при просмотре цепи обобщенных задач можно перейти в задачник и снова вернуться к просмотру цепи), так что при определении их списка используются оба комментария (Смполоса ...) и (смполоса ...).

19. x10 = "внешконъюнкция". Нажата клавиша "курсор вправо" для входа в выделение подтерма последнего выделенного терма. Аналогично оператору "списокзадач".

20. x10 = "усмчисло". Нажата клавиша "c" для включения либо выключения режима просмотра термов в скобочной записи. Аналогично оператору "списокзадач".

21. x10 = "блокприемов". Нажата клавиша "Ctr-курсор вверх" для перехода к началу задачи, для которой на экране прорисована лишь нижняя часть, либо для перехода к предыдущей задаче. Аналогично оператору "списокзадач".

22. x10 = "символтеоремы". Нажата клавиша "Ctr-курсор вниз" для перехода к следующей задаче: в верхней части экрана оказывается верхняя отделяющая линия задачи, следующей за той, к которой относится первый прорисованный на экране элемент. Аналогично оператору "списокзадач".

23. x10 = "старшийчлен" либо x10 = "натурстепень". Нажата клавиша "э" либо "Э". В первом случае предпринимается просмотр и редактирование комментариев к выделенному терму либо выделенной задаче; во втором случае - просмотр и редактирование комментариев к списку посылок выделенной задачи. Находится комментарий (Смполоса ...) к посылкам исходной задачи и проверяется, что он выделяет единственный объект. Переменной x14 присваивается набор, определяющий это выделение.

    Сначала рассматривается случай, когда x14 определяет выделение задачи; эта задача присваивается переменной x15, и проверяется, что она является обычной задачей. Если тип задачи - "исследовать" либо была нажата клавиша "Э", то переменной x16 присваивается список комментариев к посылкам задачи x15, иначе - список комментариев к этой задаче. Текущий экран сохраняется в буфере, и предпринимается обращение к оператору "сквознойпросмотр" для просмотра и редактирования списка x16. По завершении редактирования экран восстанавливается. Одноэлементные наборы из измененного списка x16 заменяются на символы, из которых они составлены, после чего предпринимается регистрация измененного списка в задаче, и откат к запросу очередной команды.

    Если x14 определяет выделение полосы задачи, то действия аналогичны. При этом сначала происходит локализация в задаче выделенного терма - так же, как это делалось при изменении терма задачи.

24. x10 = "тринадцать". Нажата клавиша Enter для продолжения трассировки либо для повторного решения вспомогательной задачи. В случае, когда не было установлено трассировки по шагам решения, а просмотр цепи обобщенных задач был инициирован командой отладчика ЛОСа, выполнение данной команды блокируется. Иначе - переход через "ветвь 3", где анализируется наличие ситуации повторного решения вспомогательной задачи. Переменные x12 и x13 инициализируются нулями. Если имеется комментарий (Смполоса ...), выделяющий некоторую обобщенную задачу, то он присваивается переменной x12, а переменной x13 присваивается одноэлементный набор, состоящий из вхождения этой задачи в цепь обобщенных задач. Если такого комментария нет, то проверяется, что в верхней части экрана находится верхняя отделяющая линия некоторой обобщенной задачи, следующей после фиктивной задачи, описывающей текущий шаг. После этого переменной x13 присваивается одноэлементный набор, состоящий из вхождения указанной обобщенной задачи в цепь обобщенных задач. В обоих случаях далее - откат к оператору "ветвь 1".

Проверяется, что x13 не равно 0, т.е. представляет собой одноэлементный набор, образованный вхождением некоторой вспомогательной задачи в цепь обобщенных задач. Переменной x14 присваивается это вхождение.

Из набора x3 извлекается информационный элемент (смкадр ...), перечисляющий стэковые кадры обобщенных задач, предшествующих фиктивной задаче. Если x14 относится к одной из таких задач, то переменной x17 присваивается ссылка на ее стэковый кадр. В этой ситуации нажатие Enter означает продолжение трассировки с переустановкой ее на шаги решения выделенной задачи. Такая переустановка предпринимается, и выход из процедуры "цепьзадач" с выдачей указателя "трассировка".

Если же x14 относится к вспомогательной задаче, то откат к оператору "ветвь 1", где x14 присваивается рассматриваемая вспомогательная задача. Переменной x15 присваивается текущее значение счетчика шагов интерпретатора.

Далее сначала рассматривается случай, когда x14 - обычная задача. Удаляются комментарии к посылкам исходной задачи, сохраняющие результаты обращений к пакетнмы операторам - чтобы они не влияли на ход решения вспомогательной задачи при повторном запуске. Сбрасываются имевшие место выделения. Находится комментарий (буфер A) к посылкам задачи x14; у него A есть вхождение в комментарий (буфер ...) к посылкам исходной задачи набора $(B_1 \dots B_5)$, описывающего результат обращения к задаче x14 в процессе применения приема. Этот набор присваивается переменной x17. Здесь B_2 - задача x14 на момент обращения к ее решению. Для повторного решения создается копия x18 данной задачи, причем копируются и те поднаборы, изменение которых может изменить B_2. Из комментариев к посылкам x18 извлекается указатель x19 на максимальный уровень обращения, имевший место при первом запуске ее решения. Чтобы после повторного решения вспомогательной задачи можно было восстановить текущее состояние комментария (буфер C) к посылкам исходной задачи и текущее значение N счетчика шагов интерпретатора, пара (C, N) сохраняется в стэке комментария (наборзначений ...). На счетчике шагов устанавливается то значение, которое имелось на момент первого запуска решения задачи x14. Стэк комментария (буфер ...) к посылкам исходной задачи расчищается; для пошагового просмотра повторного решения вводится комментарий (прерывание решить) к посылкам задачи x18, и предпринимается запуск ее повторного решения с максимальным уровнем, определяемым элементом x19. По окончании решения восстанавливаются комментарий (буфер ...) и число на счетчике шагов, извлекаемые из стэка комментария (наборзначений ...). Затем - откат к повторной выдаче исходного экрана для просмотра цепи обобщенных задач.

Повторное обращение к пакетному оператору реализуется по аналогичной схеме.

При нажатии клавиши Enter в ситуациях, отличных от перечисленных выше, предпринимается продолжение трассировки - выход из оператора "цепьзадач" с выдачей указателя "трассировка".

25. x10 = "числитель". Нажата клавиша "р" для повторения исходного кадра. Происходит откат к первому оператору "повторение" процедуры "цепьзадач".

26. x10 = "внешнийквантор". Нажата клавиша "г" для перехода в оглавление базы

приемов ГЕНОЛОГа. Вводится комментарий (автоклавиатура ...), инициирующий повторное нажатие данной клавиши, и выход из процедуры "цепьзадач". Фактический переход в оглавление базы приемов произойдет после этого из программы отладчика.

27. x10 = "минимум". Нажата клавиша "л" для перехода в оглавление программ. Аналогично предыдущему, но в комментарии (автоклавиатура ...) сохраняется нажатие клавиши "л".

28. x10 = "префикснаярекурсия" либо x10 = "заменагруппы". Нажаты клавиша "Ctr-б" либо "б". В первом случае предпринимается переход к просмотру описания альтернативного приема, который должен был бы сработать на данном шаге согласно техническому протоколу; во втором случае - переход к просмотру описания текущего приема. Аналогично предыдущему - выход во внешнюю программу отладчика и повторное выполнение той же команды.

29. x10 = "группировки". Нажата клавиша "End" для обрыва повторной трассировки вспомогательной задачи. Находится комментарий (наборзначений ...), с помощью которого восстанавливаются комментарий (буфер ...) и число на счетчике шагов интерпреатора, имевшие место перед обращением к повторной трассировке. Затем находится ссылка x15 на кадр прерывания, предшествующий текущему кадру прерывания; предпринимаются откат к этому кадру и повторное обращение к отладчику.

30. x10 = "анализатор". Нажата клавиша "п" для включения либо выключения просмотра список посылок выделенного пакетного оператора. Определяется обобщенная задача x14 этого оператора; если она не имела комментария "смпосылка", то такой комментарий вводится, иначе - удаляется. Сбрасывается комментарий, определяющий выделения. Повторно (с учетом удаленного либо введенного комментария "смпосылка") определяется список информационных элементов полос изображения задачи x14; набор x5 переустанавливается на прорисовку начиная с x14, и откат к перерисовке.

31. x10 = "заменазнака". Нажата клавиша "С" для включения либо выключения режима просмотра всех термов в скобочной записи. Если имелся комментарий "терм" к посылкам исходной задачи, то он удаляется, иначе - вводится. Затем - откат к повторной обработке цепи обобщенных задач.

32. x10 = "извлечениеварианта". Нажата клавиша "т" для просмотра и изменения режима трассировки. Аналогично тому, как эта команда обрабатывается в операторе "списокзадач".

33. x10 = "фигуры". Нажата клавиша "Ctr-Enter" для входа в трассировку решения задачи либо выполнения пакетного оператора, обращение к которым происходит на текущем шаге. Переменной x13 присваивается ссылка на стэковый кадр данной задачи либо пакетного оператора, и предпринимается переустановка на трассировку внутри данного кадра. Затем - выход в отладчик с указателем "трассировка".

34. x10 = "оператор". Нажата клавиша "Ctr-р" (кир.) для просмотра альтернативного действия, которое должно было бы выполняться согласно техническому

протоколу. Переменной x13 присваивается комментарий (техпротокол ...) к посылкам исходной задачи. С помощью этого комментария определяется текст-формульная структура x20, кратко характеризующая очередной шаг согласно техническому протоколу. После ее выдачи - пауза до нажатия любой клавиши, после чего - откат к началу выполнения процедуры "цепьзадач" (т.е. повторная выдача исходного кадра).

35. x10 = "приведение". Нажата клавиша Home для перехода к просмотру задачника. Этот переход не является откатом к внешней процедуре "списокзадач", из которой была инициирована трассировка решения. Он представляет собой повторные обращения к оглавлению задачника и процедуре "списокзадач". Так как используется "память" оглавления, то прорисовка начинается с той задачи задачника, которая последней просматривалась до этого. По выходе из просмотра - откат к повторной выдаче исходной кадра текущего шага трассировки.

36. x10 = "префиксныйфильтр". Нажата клавища "0" для сброса всех установок на прерывания и немедленного продолжения решения задачи. Кром сброса установок на прерывания, удаляется комментарий (буфер ...).

37. x10 = "унисборка". Нажата клавиша "1" для перехода к пошаговой трассировке.

38. x10 = "прогрблок". Нажата клавиша F5 для полного повторения всей трассировки до момента входа в решение выделенной вспомогательной задачи после внутреннего перезапуска. Это - отладочная функция, которая используется в тех редких случаях, когда изолированное повторное решение вспомогательной задачи не дает всей необходимой информации. Переменной x14 присваивается вхождение выделенной обобщенной задачи x15 в цепь обобщенных задач. Затем переменной x16 присваивается пара (заголовок задачи либо оператора - номер шага, на котором предпринималось обращение к ним). Переменной x18 присваивается номер шага обращения, отсчитываемый с момента запуска решения задачи из задачника. Наконец, пара (десятичная запись числа x18 - заголовок задачи либо оператора) регистрируется в логическом терминале, достижимом из корня 2-го информационного блока по меткам "альтернатива", "альтернатива", и предпринимается внутренний перезапуск.

39. x10 = "конст". Нажата клавиша "Ctr-End" для обрыва решения и возвращения в главное меню. Предпринимается внутренний перезапуск.

40. x10 = "внешвывод". Нажата клавиша "щ" для сохранения номера текущего шага решения, на котором будет предпринят выход в отладчик при автоповторе. Корректируется содержимое логического терминала "повторение" узла задачи. Особо рассматривается случай автоповтора при выводе следствий в базе теорем.

41. x10 = "попытка". Нажата клавиша Ctr-F3 для отладочного выхода в просмотр отладчиком выполнения процедуры "цепьзадач".

Глава 18

Программа редактора приемов

Редактор приемов является не только средством для создания приемов вручную, но и интерфейсом для работы с генератором приемов. Поэтому, по мере развития логической системы, время от времени возникает необходимость ввести в него новые функции либо модифицировать старые. В этой главе мы приведем техническую информацию, необходимую для выполнения таких операций.

18.1 Структуры данных, используемые для хранения приемов

База приемов ГЕНОЛОГа размещается в 8-м информационном блоке. Файлы этого блока расположены в поддиректории GEN и имеют заголовки вида $Pi_1i_2i_3i_4$, где $i_1i_2i_3i_4$ - номер файла в блоке, образованный четырьмя восьмеричными цифрами. Файл $P0000$ содержит корневой указатель - каталог; прочие файлы - собственно база приемов.

Корневой указатель 8-го информационного блока представляет собой каталог. Общие принципы организации информационных блоков с корневым каталогом уже были изложены ранее, в разделах, посвященных ЛОСу. Ссылка из каталога по логическому символу f осуществляется на указатель - список, являющийся корнем статьи символа f. Из корня статьи могут иметься переходы к следующим логическим терминалам:

1. По метке "мощность" - переход к логическому терминалу, хранящему десятичную запись числа вхождений символа f в теоремы приемов;

2. По метке "длина" - переход к логическому терминалу, хранящему десятичную запись высоты дерева номеров узлов статьи символа f;

3. По метке "новыйузел" - переход к логическому терминалу, перечисляющему те цифры - метки входа в дерево номеров узлов, по которым расположены частично заполненные ветви. Эта информация нужна для поиска первого неиспользуемого номера в дереве при регистрации новой теоремы приема.

Из корня статьи символа f выходят цифровые метки, представляющие собой корневые ребра дерева номеров узлов статьи данного символа. Это дерево образовано указателями - списками, причем концевые его вершины суть указатели, называемые узлами статьи символа f. Последовательность цифр, проходимых от корня дерева

к узлу, образует номер узла. Нумерация начинается с 1 и не обязана быть сплошной. Пробелы в нумерации могут возникать при удалении ранее введенных узлов. Однако, при вводе новых узлов имеется тенденция заполнять пробелы - каждый раз выбирается первый из неиспользуемых номеров.

Каждый узел статьи логического символа в 8-м информационном блоке определяет некоторую теорему приема, а также группу приемов, основанных на этой теореме. Из него имеются переходы по следующим меткам:

1. По метке "терм" - переход к логическому терминалу, хранящему теорему приема;

2. По метке "команды" - переход к указателю-списку U, перечисляющему заголовки приемов, основанных на теореме узла. Обычно при вводе нового приема, основанного на той же теореме, что и предыдущий прием, происходит дублирование теоремы, и тогда из U будет выходить единственное ребро. Однако, можно вводить приемы более экономно, и тогда будут возникать неодноэлементные списки переходов из U. По меткам - заголовкам приемов из U ведут переходы к указателям - спискам, называемым узлами приемов.

3. По метке "геомредактор" - переход к логическому терминалу, хранящему описание чертежа, сопровождающего теорему приема.

Для работы с логическими терминалами, хранящими описания чертежа, существуют операторы ЛОСа "регистрациячертежа" и "чтениечертежа", записывающие структуру данных геометрического редактора в информационный блок либо считывающие ее оттуда. Форматы хранения чертежа в информационных блоках были приведены при описании операторов ЛОСа.

Узел приема является корнем ветви, хранящей всю основную связываемую с приемом информацию. Из него имеются следующие переходы:

1. По метке "команда" - переход к логическому терминалу, хранящему описание приема. Это описание имеет следующие компоненты:

   а) Терм "условие(и($A_1 \dots A_n$))" задает список фильтров приема $A_1, \dots, A_n$. Некоторые A_i могут представлять собой логический символ ","; они суть разделители между выводимыми на экран страницами, перечисляющими фильтры;

   б) Термы и логические символы с заголовками, отличными от символов "условие", "быстрпреобр" - указатели приема. Фиктивные однобуквенные термы "," используются как разделители между страницами, перечисляющими указатели;

   в) Термы вида "быстрпреобр(t $A_1 \dots A_n$)" - группы нормализаторов приема. Для краткости, иногда эти термы тоже называем нормализаторами приема. Здесь t - нормализуемый терм, заданный либо непосредственно, либо с помощью указателя вхождения в теорему приема "фикс(...)". Выбирается тот из вариантов задания t, который короче. $A_1, \dots A_n$ - последовательность нормализаторов терма t.

2. По метке "прием" - переход к логическому терминалу, хранящему название логического символа, за которым закреплена программа приема, и несколько

первых операторов этой программы. Этот терминал вводится для откомпилированных приемов; если его нет, то под описанием просматриваемого приема проводится красная черта.

При поиске программы приема в программе заданного логического символа, указанного в терминале "прием", используется ключевой оператор "контроль-приема(A_1 A_2 A_3)". При обращении к нему A_1 равно логическому символу, за которым закреплен прием, A_2 - номеру узла этого логического символа (числу в формате ЛОСа); A_3 - заголовку приема (логическому символу либо терму). В обычных режимах данный оператор не выполняет никаких действий, а нужен лишь для поиска той начальной точки, после которой все операторы ЛОСа относятся только к данному приему. Этот поиск может осуществляться либо компилятором ГЕНОЛОГа, либо системой трассировки, определяющей прием, к которому относится текущая точка выполняемой программы.

3. По метке "текстблок" - переход к указателю-списку, из которого по меткам 1,2,3,... - переходы к текстовым терминалам, хранящим комментарии к приему. Это - записная книжка, в которую можно заносить различные пометки при работе с приемами. В сложившейся практике обучения решателя она так и не была востребована. Вероятно, это произошло из-за того, что описание приема на ГЕНОЛОГе и без того оказалось достаточно доступным для понимания.

4. По метке "оглавление" - переход к логическому терминалу, хранящему набор логических символов - меток перехода, определяющих путь в оглавлении базы приемов к его концевому пункту, ссылающемуся на данный прием. Такие возвратные ссылки на оглавление бывают нужны, например, для переключения его текущего пути на текущий применяемый прием. При перестановке пунктов оглавления происходит автоматическая коррекция возвратных ссылок. Если, однако, по каким-либо причинам обратные ссылки оказались испорчены, их можно восстановить, войдя в корневое меню оглавления базы приемов и нажав клавишу "Ctr-т" (кир.).

5. По метке "начало" - переход к логическому терминалу, хранящему терм "прием(A_1 A_2 A_3)", ссылающийся на сохраненную в буфере базы приемов исходную версию приема. Здесь и далее используемый для ссылки на прием терм "прием(...)" имеет своим первым операндом A_1 логический символ, за которым закреплено описание приема, вторым операндом A_2 - номер узла этого символа, хранящего теорему приема; третьим операндом A_3 - заголовок приема. Напомним, что в буфер базы приемов заносятся: а) исходные версии измененных приемов (для таких исходных версий вводятся новые узлы); б) удаленные приемы (их узлы сохраняются); в) новые приемы (здесь нет дублирования приема, а есть только ссылки на него из двух разных точек).

6. По метке "конец" переход осуществляется только для приема, находящегося в буфере после изменения либо удаления. Он ведет к логическому терминалу, хранящему ссылку "прием(...)" на текущую версию этого приема буфера, а если прием попал в буфер не при изменении, а при удалении, то вместо ссылки "прием(...)" в терминале находится символ "пусто".

7. По метке "переменные" - переход к логическому терминалу, хранящему набор термов "знач(x A)", перечисляющий программные выражения A для термов,

идентифицированных с теоремными переменными x. Такой терминал вводится только для приемов, скомпилированных нажатием клавиши $F6$. При перекомпиляции их через $F3$ либо $F5$ он удаляется.

8. По метке "титр" - переход к текстовому терминалу, хранящему шаблоны для текстформульного сопровождения срабатывания приема.

9. По метке "примечание" - переход к логическому терминалу, хранящему установку на синтез приема. Такая установка представляет собой, по существу, прообраз описания приема на ГЕНОЛОГе, но гораздо более компактный. Из нее и из теоремы приема генератор приемов создает описание приема. При этом он использует ряд дополнительных источников информации, и в первую очередь - базу теорем. Фактически, генератор приемов здесь в значительной степени выполняет функции компилятора. Развитие процедур генератора приемов осуществляется на основе анализа массивов приемов, созданных вручную. Первым шагом в этом направлении явилось построение предварительной классификации типов приемов ГЕНОЛОГа. Ссылка на тип приема в данной классификации играет в установке на синтез приема роль, аналогичную роли заголовка приема в его описании. Автоматизация синтеза приемов потребовала "встречного движения" со стороны самой базы приемов - чтобы можно было объяснить программе принципы создания приема, его сначала необходимо освободить от различных предрассудков, накопившихся при быстрой проработке потока задач. Если же какие-то компоненты описания приема устранить не удается, то обычно бывают ясны и их источники. Таким образом, работа над генератором приемов оказалась одновременно организующим началом для работы над оптимизацией решателя. Так как она в настоящее время продолжается, оставим изложение подробностей до третьего тома монографии, посвященного базе теорем.

10. По метке "сравнениеприемов" - переход к логическому терминалу, содержащему шестерку чисел (A_1 A_2 A_3 A_4 A_5 A_6). A_1, A_2, A_3 - число различий междй автоматически сгенерированной версией приема и его "ручной" версией из базы приемов, соответственно, в фильтрах, нормализаторах и указателях. B_1, B_2, B_3 - те же значения на предыдущем цикле тестирования генератора приемов. Эти терминалы составляют часть системы обучения генратора приемов. Они нужны для контроля за его способностью создавать ранее проработанные приемы по мере внесения в него изменений.

11. По метке "теорема" - переход к логическому терминалу, хранящему ссылку "теорема(A_1 A_2)" на узел теоремы в базе теорем, являющийся источником данного приема. Здесь A_1 - логический символ, за которым закреплена теорема, A_2 - номер ее узла в базе теорем. Предполагается постепенно связать почти все теоремы приемов с реальными теоремами, размещаемыми в базе теорем. Исключение будут составлять только псевдотеоремы, имеющие заведомо технический характер. Для них будет предусмотрен другой способ указания источника их возникновения. Как уже отмечалось, теорема приема является компонентой алгоритмического языка и может несколько отличаться от той реальной теоремы, из которой она возникла. Интерфейс привязки приема к теореме несложен - из просмотра приема нажимается клавиша $Ctr - F9$, после чего на экране появляется либо описание теоремы, ранее связанной с приемом, либо (если таковой

еще нет) - появляется оглавление базы теорем. В этом оглавлении нужно войти в просмотр теоремы - источника теоремы приема, и нажать "И" для закрепления ее в качестве источника. Из узла теоремы при этом возникнет встречная ссылка на прием. Обе ссылки автоматически удаляются, если удаляется либо прием, либо теорема.

12. По метке "заголовокприема" - переход к логическому терминалу, хранящему пару логических символов A_1, A_2. A_1 характеризует последнюю попытку автоматического синтеза по теореме базы теорем установки на синтез данного приема; A_2 - предпоследнюю. Символ 1 означает, что попытка была удачной, символ 0 - что установка не была создана. Установки на синтез приемов создаются процедурами базы теорем на основе некоторой целевой характеризации теоремы, в частности, с учетом того, для чего она была выведена. Терминалы "заголовокприема" составляют часть системы обучения этих процедур.

Структуры данных оглавления базы приемов тоже располагаются в 8-м информационном блоке. Чтобы найти корень этого оглавления, нужно перейти от корневого каталога информационного блока по меткам "оглавление", "прием". Концевые логические терминалы оглавления базы приемов содержат термы "прием(A_1 A_2 A_3)", ссылающиеся на приемы в уже описанном выше формате. Чтобы выделить в этом списке ссылок на приемы ссылку на текущий прием (он будет прорисовываться при входе в просмотр концевого терминала оглавления), соответствующий терм "прием(...)" заменен на терм "текприем(...)". Все процедуры переустановки текущего пути оглавления базы приемов на найденный прием (например, на текущий прием трассировки) дополняются коррекцией выделения заголовком "текприем" непосредственной ссылки на прием.

Указатели и концевые пункты оглавления базы приемов могут снабжаться техническими пометками - комментариями. Комментарии к указателю-списку, представляющему меню оглавления, хранятся в логическом терминале "замечание" этого указателя. Комментарии $A_1, \ldots, A_n$ к концевому пункту оглавления, который сам является логическим терминалом, размещаются в нем же. При этом они группируются в терм "замечание($A_1 \ldots A_n$)", что позволяет отличить их от ссылок "прием(...)". Для выделения буфера базы приемов и его разделов используются следующие типы комментариев:

1. "буфер" - ветвь оглавления является буфером базы приемов;

2. "исключение" - ветвь буфера базы приемов, используемая для сохранения информации о последних удаленных приемах;

3. "изменение" - ветвь буфера базы приемов, используемая для сохранения информации об измененных приемах;

4. "новый" - ветвь буфера базы приемов, используемая для сохранения информации о новых приемах.

5. "копия" - логический терминал ветви "новый" буфера базы приемов.

По мере развития базы приемов происходит рост ее оглавления. К сожалению, в нынешней версии интерпретатора ЛОСа все оглавление (как и вообще любая корневая ветвь информационного либо программного блока, соответствующая заданному

логическому символу) должно целиком размещаться в одном файле информационного блока. Размер такого файла ограничен сверху 512 Кб. Это ограничение накладывается 19-битной кодировкой смещений в файлах, принятой в интерпретаторе (еще 12 бит приходятся на 4-значный восьмеричный номер файла в блоке). В принципе, можно модифицировать интерпретатор, выделив для нумерации файла в блоке лишь три восьмеричных цифры и увеличив размеры одного файла в 8 раз. Пока этого нет, приходится следить за возникновением переполнений в отдельных файлах информационного блока. Разумеется, при приближении этих размеров к 512 Кб происходит автоматическое разрезание файла на две примерно равные части, между которыми перераспределяются ветви логических символов, объединенные ранее в одном файле. Однако, в редких случаях оказывается, что почти весь файл был занят ветвью одного символа, и тогда разрезание практически ничего не дает. Эти редкие случаи до сих пор были связаны с двумя причинами - ростом размеров оглавления либо неравномерным распределением приемов по логическим символам. Для борьбы с ситуациями первого типа нужно время от времени разрезать оглавление, выделяя какую-либо большую его ветвь и закрепляя ее за отдельным логическим символом (см. описание структуры данных оглавлений и интерфейс их разрезания). Для борьбы с ситуациями второго типа можно "разгружать" чрезмерно часто используемый логический символ, перезакрепляя приемы за другим символом. Выбор такого символа вообще несущественен, и его не обязательно связывать с символами, встречающимися в теореме приема.

18.2 Обращение к редактору приемов

Программа редактора приемов реализуется оператором "блокприемов(x1)". Значением входной переменной x1 этого оператора является ссылка на концевой логический терминал оглавления базы приемов. Программа осуществляет просмотр списка приемов, адресуемых из этого терминала, редактирование либо удаление их, а также ввод новых приемов. Предусмотрен еще один вариант обращения к оператору "блокприемов" - для просмотра группы каким-либо образом отобранных приемов, безотносительно к ссылкам на них из оглавления. В этом случае значением x1 служит набор термов "прием(...)", ссылающихся на отобранные приемы. Здесь ввод новых приемов невозможен.

Обращение к оператору "блокприемов" из просмотра оглавления базы приемов выполняется программой общего интерфейса логической системы - программой символа "вход". Найти ту ее ветвь, где это происходит, можно, например, из раздела ("Общий интерфейс", "Обращение к базе приемов") оглавления программ. Эта ветвь, вплоть до обращения к оператору "блокприемов", уже была описана ранее.

18.3 Прорисовка текущего приема

Начиная с этого момента, будем шаг за шагом продвигаться вдоль текста программы оператора "блокприемов". Рекомендуется войти в просмотр корневого фрагмента этой программы и "перелистывать" ее по мере чтения данного описания.

Начинается работа оператора с прорисовки в верхней части экрана меню, соответствующего контексту просмотра описания приема. Если значением переменной x1 являлась ссылка на концевой логический терминал оглавления базы приемов, то находится содержимое x2 этого терминала; в противном случае переменной x2

передается значение x1. В обоих случаях значением x2 теперь является набор термов "прием(...)", быть может, пополненный термом "замечание(...)". Переменной x3 присваивается вхождение в набор x2 его первого разряда - далее эта переменная будет указывать ту позицию набора x2, на которой расположена ссылка на текущий просматриваемый прием. Если оказалось, что на позиции x3 размещается терм "замечание", то эта позиция сдвигается на единицу вправо.

Начиная с оператора "позиция(x4 x2)" реализуется цикл просмотра набора x2 для поиска терма "текприем(...)", ссылающегося на прием, который изначально выделен в качестве текущего. Если этот терм найден, то x3 переустанавливается на его вхождение, а заголовок "текприем" заменяется на "прием". После этого все ссылки на приемы в x2 равноправны; заголовок "текприем" будет восстановлен лишь по выходе из оператора - для того приема, который будет просматриваться на момент выхода.

По переходу "ветвь 2" попадаем в фрагмент, используемый в цикле тестирования генератора приемов на подразделе базы приемов. Указанием на наличие такого цикла служит комментарий "подраздел" к посылкам исходной задачи; в этом случае создается комментарий "автоклавиатура" для автоматического нажатия клавиши "П", запускающей генератор приемов.

Переходя через "ветвь 1", оказываемся в фрагменте, который прежде всего активирует 8-й информационный блок. По умолчанию, все дальнейшие действия оператора "блокприемов" предпринимаются в предположении, что активным является этот блок. Если возникает необходимость обратиться к другим информационным блокам, то переключение на них должно быть локальным - по завершении работы с ними сразу нужно активировать 8-й блок.

Затем инициализируется структура данных x4 для работы с текущим приемом. Она представляет собой набор информационных элементов, характеризующих прием, его визуализацию и текущее состояние выполняемых с ним действий. Прежде чем продолжить просмотр программы данного фрагмента, перечислим основные типы таких информационных элементов. Как и ранее, набор записываем в виде заключенной в скобки последовательности его разрядов.

1. (логсимвол A) - A есть логический символ, за которым закреплена теорема приема;

2. (число A) - A есть номер узла статьи указанного выше логического символа, содержащего теорему приема;

3. (адресузла A) - A есть непосредственная ссылка на узел теоремы приема в 8-м информационном блоке;

4. (заголовок A) - A есть заголовок приема. Если этот заголовок однобуквенный, то он представлен в формате логического символа, иначе - в формате терма;

5. (команда A) - A есть непосредственная ссылка на узел приема в 8-м информационном блоке;

6. (прием A) - A есть набор термов, образующих описание приема (включая терм "условие(...)", задающий фильтры, а также включая нормализаторы).

7. (терм A) - A есть теорема приема;

8. (видео A_1 A_2 A_3 A_4) - приводятся структуры данных A_1, A_2, A_3, A_4, используемые для выдачи на экран, соответственно, теоремы приема; заголовка приема; списка фильтров и списка указателей.

   A_1 - тройка (B_1 B_2 B_3), где B_1 - скорректированная для выдачи на экран теорема приема; B_2 - либо логический символ "терм", если теорема не поддается выдаче на экран с помощью формульного редактора, либо корень созданного формульным редактором дерева указателей для прорисовки теоремы. B_3 - либо 0, либо выделенное при работе с редактором приемов вхождение в теорему.

   A_2 - пара (B_1 B_2), где B_1 - номер строки, по которой проводится верхняя отделяющая линия второго окна; B_2 - заголовок приема.

   A_3, A_4 - наборы (B_1 B_2 B_3 B_4), где B_1 - номер строки, по которой проводится верхняя разделяющая линия; B_2 - набор термов (фильтров либо указателей); B_3 - набор той же длины, что B_2, сопоставляющий каждому терму координату исходной позиции для выдачи его на экран (пару символьных номеров столбца и строки); B_4 - набор той же длины, то B_2, сопоставляющий каждому терму либо 0, либо выделенное вхождение в этот терм.

9. (быстрпреобр A) - A есть набор всех нормализаторов приема;

10. (программа A) - элемент вводится в том случае, когда существует программа приема. Тогда A - логический символ, к которому относится эта программа;

11. (просмотртерма A) - A есть номер активного окна приема (от 1 до 4);

12. (конец A) - A есть номер строки для нижней граничной линии под изображением приема;

13. (комментарии A) - A есть ссылка на текстовый терминал, хранящий комментарии к приему;

14. (изменение A_1 A_2 A_3 A_4 A_5) - указатель измененных при редактировании окон приема. $A_i = 0$ означает, что i-е окно изменено не было; $i = 1, \ldots, 5$, причем под пятым окном понимается список нормализаторов приема;

15. (пассив A) - окно с номером A не выводится на экран (номер - от 1 до 4);

16. (геомредактор A_1 A_2) - A_1 есть описание чертежа в формате оператора "геомредактор"; A_2 - строка для выдачи теоремы под чертежом;

17. (оглавление A) - A есть путь в оглавлении базы приемов от его корня к текущему концевому пункту;

18. (примечание A) - A есть установка на синтез приема;

19. (слово A_1 A_2) - A_1 есть ссылка на текстовый терминал, содержащий текст, выводимый на экран вместо теоремы (используется в приемах операторов фильтра); A_2 - строка после выдачи данного текста;

20. (См A_1 A_2) - A_1 есть сивмольное число разделителей ",", после которых располагается выводимый на экран фрагмент списка фильтров; A_2 - символьное число разделителей ",", после которых располагается выводимый на экран фрагмент списка указателей приема.

21. (указатель A_1 A_2) - A_1 есть список пар (B_1 B_2), определяющих рассогласования между созданными генератором приемов и ранее имевшимися указателями приема. Либо B_1 - указатель, созданный генератором приемов и отсутствовавший ранее - тогда $B_2 = 0$, либо $B_1 = 0$ и B_2 - ранее введенный указатель, не созданный генератором приемов. Перед определением рассогласований групповые указатели расформируются в серии одиночных. A_2 - текущая позиция в списке A_1.

Вернемся к рассмотрению программы. После присвоения переменной x4 пустого слова расположен оператор "повторение", к которому будут происходить откаты при различных изменениях текущего приема. Сразу за ним предпринимается коррекция набора x4 - почти все его элементы удаляются. Остается, в частности, элемент (См ...), так как он фиксирует выдачу на экран конкретных страниц третьего и четвертого окон.

После оператора "исходнаязадача(x5)" сначала проверяется, не произошел ли вход в просмотр приемов из контейнера (переход через "иначе 3"). Эта ветвь сейчас для нас неинтересна. Затем проверяется наличие комментария (контрольпрограммы ...) к посылкам исходной задачи. Этот комментарий вводится при различных временных переходах из просмотра приема (например, при переходе к просмотру его программы). Он нужен для возвращения в исходную точку просмотра и позволяет восстановить значения переменных x2, x3, x4. Далее (переход через "иначе 4") - проверяется наличие комментария (регистрациятеоремы ...). Через него новому приему передается значение структуры данных x4 при копировании старого приема.

Возвращаясь в фрагмент, содержащий оператор "равно(x4 пустоеслово)", далее переходим через "ветвь 2" к фрагменту, который по комментарию (простыеделители ...) определяет текущий путь в оглавлении базы приемов и регистрирует его в x4. Напомним, что данный комментарий создается процедурой "оглавление" при выборе концевого пункта.

Далее - переход через "ветвь 1" к фрагменту, продолжающему заполнение структуры данных x4. Здесь определяются логический символ x5, за которым закреплен текущий прием, находятся номер x6 узла его теоремы и заголовок x7. Определяется ссылка x8 на узел теоремы приема. Находятся одноэлементные наборы x9 и x12, содержащие, соответственно, ссылку на логический терминал, хранящий теорему приема, и на логический терминал с описанием приема. При наличии терминала "примечание" с установкой на синтез приема эта установка регистрируется в x4.

После перехода через "ветвь 2" - фрагмент, в котором считывается содержимое x13 терминала описания приема. Если в описании имеется более одного терма "условие(...)", что свидетельствует об ошибке, то сохраняется только последний такой терм, и терминал корректируется. Далее - переход через "ветвь 1", где расположен еще один пункт "самокоррекции". Проверяется наличие идентичных указателей либо нормализаторов, и при наличии таковых лишние отбрасываются.

Переходя затем через "ветвь 1", оказываемся в фрагменте, где происходит считывание теоремы приема - ее содержит теперь одноэлементный набор x14. На основе накопленной к данному моменту информации в набор x4 заносятся элементы (логсимвол ...), (число ...), (заголовок ...), (адресузла ...), (терм ...), (команда ...), (прием ...). В случае приема оператора фильтра (заголовок "контекст") в x4 заносится также элемент (слово ...). Наконец, переход через "ветвь 1" к терминалу, где в x4 заносится элемент (программа ...).

На этом цикл предварительного комплектования набора x4 завершается, и проис-

ходит откат к фрагменту, инициализировавшему x4 пустым словом. Далее из него по "ветвь 1" - переход к фрагменту, начинающему прорисовку приема. Он имеет метку "прием(4 6)", и выйти на него можно также из оглавления программ - см. раздел ("Цикл прорисовки приема", "Исходная точка").

Фрагмент начинается с оператора "повторение", к которому будут выполняться откаты для перерисовки изображения приема. Предпринимается расчистка экрана, и далее проверяется наличие в наборе x4 информационного элемента (видео ...). Если такой элемент есть, то это означает, что прорисовка приема уже выполнялась, и можно воспользоваться готовыми данными для ее восстановления. Это делает ветвь программы, переход к которой - через оператор "ветвь 3". Предварительно анализируется наличие особых случаев, связанных с комментарием (теквхожд ...) к посылкам исходной задачи - здесь может произойти либо сброс элемента (видео ...), либо вообще отказ от прорисовки приема.

Прорисовка приема по элементу (видео ...), в общем, не содержит никаких особо интересных моментов и выполняется в соответствии с описанной выше структурой данных. Поэтому мы не будем ее рассматривать, а сразу перейдем через оператор "иначе 2" к главному циклу, выполняющему первичную прорисовку приема.

Прежде всего, из набора x4 извлекается элемент (терм ...), содержащий теорему приема; этот элемент присваивается переменной x6. Рамка прорисовки устанавливается на прямоугольник, расположенный правее левого 16-пиксельного столбца, так как в этом столбце размещаются номера окон. Теорема приема присваивается переменной x7. Вводится накопитель x8 структуры данных для выдачи теоремы на экран (он будет зарегистрирован в элементе (видео ...)). Проверяется отсутствие элемента (пассив 1), отменяющего прорисовку первого окна. Проверяется отсутствие элемента (слово ...), указывающего на прорисовку вместо теоремы некоторого текста. При наличии такого элемента прорисовка выполняется фрагментом, достижимым через "иначе 4". Далее прорисовывается цифра "1" - номер первого окна. Если имеется элемент (геомредактор ...), то переменной x9 присваивается номер строки под чертежом, и предпринимается прорисовка этого чертежа.

Если исходная задача имеет комментарий "терм" к своим посылкам, то теорема будет прорисовываться текстовым редактором - это сделает часть программы, достижимая через "ветвь 5". Иначе происходит обращение к формульному редактору, который дает дерево указателей x10, и оператор "блокредактора(3 ...)" осуществляет прорисовку теоремы. Переменной x8 присваивается тройка (теорема - корень дерева указателей - 0). Затем происходит "переназначение" переменной x7 - далее она будет указывать номер первой свободной строки. Под теоремой прорисовываются пары, указывающие для каждой переменной, встречающейся в теореме, ее обозначение в виде "xn". Далее - откат к переходу через "ветвь 2".

Перейдя через "ветвь 2", оказываемся в фрагменте программы, содержащем метку "прием (4 9)". К нему можно перейти также из пункта "Прорисовка заголовка приема" оглавления программ. Прежде всего, проверяется, равно ли x7 (первая свободная строка) нулю, и если не равно, то по x7 прорисовывается горизонтальная отделяющая линия. Затем из x4 извлекается элемент (заголовок ...), который присваивается переменной x9. Если заголовка приема еще нет, то в x4 заносятся элементы (видео ...) и (конец ...), соответствующие прорисовке одной только теоремы. Если же заголовок есть, то переменной x9 переприсваивается новое значение - структура данных для прорисовки заголовка, которая впоследствии будет передана в элемент (видео ...). Проверяется отсутствие элемента (пассив 2), отменяющего прорисовку заголовка; прорисовывается номер "2" окна заголовка; предпринимается

прорисовка заголовка приема в виде логического символа либо терма скобочной записи; корректируется номер x7 первой незанятой строки; по ординате x7 проводится горизонтальная отделяющая линия, и откат к переходу через "ветвь 2".

Здесь возникает фрагмент с меткой "прием(5 0)". Перейти к нему можно из пункта оглавления программ "Прорисовка фильтров приема". Прежде всего, находится элемент (прием ...) с описанием приема. Если его еще нет (на начальном этапе создания приема), то элементы (видео ...), (конец ...) заносятся в x4 по текущей прорисовке. Иначе вводится заготовка x11 структуры данных для прорисовки фильтров (см. элемент (видео ...)). Если в x4 нет блокирующего прорисовку фильтров элемента (пассив 3), то в описании приема находится элемент x12 вида "условие(...)". При отсутствии такого элемента переменная x7 увеличивается на 3, и откат к оператору "ветвь 2", где будет продолжаться прорисовка указателей приема. Увеличение x7 на 3 означает, что окно фильтров будет прорисовано символически - как очень узкая трехпиксельная полоса между двумя горизонтальными отделяющими линиями. Если же элемент "условие(...)" имеется, то прорисовывается номер "3" окна фильтров. Затем находится набор x13 операндов терма "условие(...)", т.е. фильтров приема. Вводится набор x14 той же длины, что и x13. Изначально он заполнен нулями, но в процессе прорисовки его разряды будут заменяться парами (столбец-строка), указывающими начальную позицию очередного фильтра. Для последовательной прорисовки фильтров устанавливается начальная позиция на экране. Чтобы выдать лишь текущую страницу списка фильтров, переменной x15 присваивается число разделителей ",", после которых начинается эта страница, и вводится переменная x16, которая при просмотре списка фильтров указывает на число пройденных запятых. Затем выполняется одновременный просмотр списков x13 и x14, с прорисовкой фильтров (на участке, где x15 и x16 совпадают) и регистрацией в x14 их координат. По окончании просмотра заполняются пустые позиции структуры данных x11 и корректируется номер x7 первой свободной строки. Далее - откат к переходу через "ветвь 2".

После указанного перехода возникает фрагмент, в середине которого размещена метка "прием(5 1)". Этой метке соответствует пункт "Прорисовка указателей приема" оглавления программ. Если в x4 нет элемента (пассив 3), указывающего на пропуск третьего окна, то по текущей ординате x7 проводится горизонтальная отделяющая линия. Затем переменной x12 присваивается список всех указателей нормализации "быстрпреобр(...)"; если в x4 еще не было элемента (быстрпреобр ...), перечисляющего эти указатели, то он вводится. Создается список x13 всех указателей приема, преобразованных перед прорисовкой к формату термов. Если для указателей места на экране не хватает, то они не выводятся; в набор x4 заносятся элементы (видео ...) и (конец ...) для прорисованной части приема, и откат к обработчику команд. Иначе реализуется цикл прорисовки указателей, аналогичный разобранному выше циклу прорисовки фильтров. По завершении его в набор x4 заносится полностью укомплектованный элемент (видео ...), а также элемент (конец ...). Проводится нижняя отделяющая линия, цвет которой информирует о наличии программы приема и наличии изменений в нем, и далее - откат к обработчику команд.

18.4 Обработчик команд редактора приемов

Обработчик команд редактора приемов располагается начиная с фрагмента, соответствующего пункту "обращение к клавиатуре" раздела "Редактор ГЕНОЛОГа" оглавления программ. После метки "прием(2)" в этом фрагменте размещен оператор

"повторение", к которому будут происходить откаты при запросе очередной команды. Далее идет оператор "автоменю(x5)", осуществляющий ввод новой команды x5. После этого следует весьма длинная цепочка фрагментов, обрабатывающих конкретные команды. Эти команды расположены в достаточно случайном порядке, поэтому ниже мы не будем его придерживаться, а описание действий команд дадим, сгруппировав их по функциональным признакам. Ряд второстепенных команд опустим.

18.4.1 Переходы

1. Выход из просмотра приема либо отмена выделения окна приема. Команда реализуется в том же фрагменте, где происходит ввод команды. Все команды, рассматриваемые в данном подразделе, собраны в разделе ("Редактор ГЕНОЛОГа", "Переходы") оглавления программ, и расположены в том же порядке. Выходить на начальные точки их обработки при чтении приводимых пояснений рекомендуется через пункты указанного раздела оглавления программ.

   x5 - либо один из символов "Плюс", "циклвариантов", либо символ "обл", в ситуации, когда x4 содержит элемент (просмотртерма ...). Это означает, что были нажаты, соответственно, клавиши "Esc", "курсор влево" либо "пробел", причем в последнем случае одно из окон приема выделено изменением цвета его номера.

   Обработка команды начинается с удаления комментария (сравнениеприемов ...) к посылкам исходной задачи, создаваемого генератором приемов для сравнения его версии приема с той версией, которая фактически имеется в решателе. Этот же комментарий удаляется рядом других команд, чтобы старая информация о тестировании генератора приемов не накладывалась на новую.

   Если в x4 имеется элемент (просмотртерма i), указывающий номер i выделенного окна, то восстанавливается черный цвет этого номера, элемент (просмотртерма i) удаляется, и откат к очередной команде. Если же такого элемента нет, то команда означает выход из редактора приемов. Перед выходом ссылка "прием(...)" на текущий прием в списке x2 заменяется на "текприем(...)", и это изменение регистрируется в текущем концевом пункте оглавления базы приемов. Если исходная задача имеет комментарий "текприем", означающий наличие цикла просмотра базы приемов с помощью процедуры "текприем", то при нажатии Esc выход из редактора приемов осуществляется по значению "ложь". Такой выход приведет к обрыву цикла. В остальных случаях выход из редактора приемов происходит по значению "истина". При наличии цикла просмотра это будет означать переход к следующему приему серии.

2. Переход к другому приему данного концевого пункта оглавления.

   x5 - символ "внешсумма" либо "подстановка". Соответственно, нажаты клавиши "курсор вверх" либо "курсор вниз". Фактически, здесь обрабатываются две возможных команды.

   Если в x4 имеется элемент (просмотртерма ...), означающий, что выделено некоторое окно приема, то указанные клавиши просто увеличивают либо уменьшают на единицу номер выделенного окна. Эта смена выделенного окна будет иметь место только в ситуации, когда еще не начат просмотр содержимого окна, так как иначе команды обрабатываются другими ветвями программы.

Если же в x4 нет такого элемента, то переход через "ветвь 2". Здесь переменной x6 присваивается вхождение в набор x2 ссылок на приемы левого либо правого соседа текущего разряда x3. Если это вхождение пришлось на терм "замечание(...)", то сдвиг вхождения в указанном направлении повторяется. Если позиция не была крайней в наборе x2, то переменной x3 присваивается значение x6, и происходит откат к циклу заполнения набора x4.

3. Переход к просмотру программы приема.

   x5 - символ "приведение". Нажата клавиша "Home". Выполнение команды блокируется, если прием просматривается из контейнера (т.е. из буфера, через который возможен обмен информацией между разными экземплярами логической системы), либо из серийного просмотра базы приемов с помощью процедуры "текприем", либо из трассировки решения задачи. Переменной x6 присваивается текущая ссылка "прием(...)" на просматриваемый прием, и по ней определяется вид x10 оператора "контрольприема(...)", используемого в программе для обозначения начала приема. Переменной x16 присваивается логический символ, за которым закреплена программа приема. Вводится комментарий (контрольпрограммы A_1 A_2 A_3 A_4 A_5) к посылкам исходной задачи, содержащий информацию для поиска исходного фрагмента программы приема и для возвращения в редактор приемов после просмотра программы. У этого комментария A_1 - логический символ, к которому относится программа; A_2 - указанный выше оператор "контрольприема(...)"; A_3 - ссылка на текущий концевой логический терминал оглавления базы приемов; A_4 - пара (x2 - x3); A_5 - набор x4. Вводится также комментарий (прогрблок x16) к посылкам исходной задачи, который указывает обработчику команд общего интерфейса системы на необходимость обращения к просмотру программы символа x16. Создается комментарий (автоклавиатура ...), обеспечивающий эмуляцию нажатия клавиши F8 для поиска редактором программ того фрагмента, в котором встречается оператор A_2. Так как при просмотре фрагментов программы название логического символа, к которому относится программа, не перерисовывается, то выполняется заблаговременная его прорисовка в верхней части экрана. На тот случай, если возвращение к редактору приемов произойдет не сразу по окончании просмотра программы, в концевом текущем логическом терминале оглавления базы приемов предпринимается выделение заголовком "текприем" текущего просматриваемого приема. Затем происходит выход из редактора приемов.

   Все дальнейшие переключения, приводящие к просмотру программы, обеспечиваются внешними процедурами. Эту траекторию можно проследить, если найти раздел ("Общий интерфейс", "Обращение к базе приемов") оглавления программ и просмотреть операторы, следующие за оператором "блокпримов(x11)". Произойдет откат к обработчику команд главного меню системы; оттуда (благодаря введенным комментариям к посылкам исходной задачи) - переход к просмотру программы символа x16, и далее - поиск фрагмента этой программы, содержащего A_2. На всех этих этапах промежуточная прорисовка блокируется. Возвращение в просмотр приема инициируется нажатием клавиши "End" из редактора программ. Здесь снова будет выполнен переход с откатом к обработчику команд главного меню, но уже с обращением к редактору приемов. Роль диспетчера, переключающего траекторию отката на редактор приемов, здесь играет вводимый перед откатом комментарий (цепьвхождений "x3").

4. Переходы к другим приемам того же самого узла теоремы.

   x5 - символ "контрольунификации" либо "точкапривязки". Соответственно, нажаты клавиша PageUp либо PageDn.

   Проверяется отсутствие в x4 элемента (просмотртерма ...), выделяющего окно приема. Если такой элемент есть, то переход через "иначе 2", где реализуется переход к предыдущей либо следующей странице выделенного окна фильтров либо окна указателей. Для смены страницы корректируется элемент (См ...) из x4. Если же выделения окон приема нет, то, как и указано выше, осуществляется переход к другому приему того же самого узла теоремы. Находится ссылка x8 на указатель-список, метки которого суть различные заголовки приемов, использующих этот узел. Переменной x9 присваивается набор всех таких заголовков, и в нем определяется вхождение x10 заголовка текущего просматриваемого приема. В зависимости от нажатой клавиши, находится вхождение x11 предыдущего либо следующего разряда в x9. Находится ссылка "начало(x12)" на узел приема, заголовок которого расположен по вхождению x11. Определяется путь x14 в оглавлении базы приемов к концевому логическому терминалу K, ссылающемуся на этот прием; текущий путь оглавления переустанавливается по x14. Находится содержимое x17 терминала K. В нем идентифицируется ссылка x19 на найденный прием, заголовок которой изменяется на "текприем". Наконец, значения x1, x2 и x3 модифицируются для просмотра нового текущего приема, и откат к переопределению набора x4.

5. Переходы к оглавлениям программ, задачника и базы теорем.

   x5 - один из символов "элементызадачи", "стрелкапирса", "эквивалентно". Соответственно, нажаты клавиши "Shift-1", "Shift-3" и "B" (кир.). В первых двух случаях будут выполнены переходы к оглавлениям программ и задачника; в третьем, весьма специфическом, текущий прием должен быть одним из так называемых приемов вывода теорем. Эти приемы имеют заголовок "теорема"; нажатие клавиши инициирует переход в базу теорем и одновременный запуск серийного тестирования применений данного приема. Подробнее об этом будет говориться в томе, посвященном базе теорем.

   Во всех трех случаях для организации перехода вводится комментарий (новпозиция A) к посылкам исходной задачи. Здесь A - код той клавиши, нажатие которой в главном меню системы вызывает переход к нужному оглавлению. После ввода комментария - выход из процедуры "блокприемов", за которым последует откат к обработчику команд главного меню.

6. Переход в базу теорем для просмотра источника приема.

   x5 - символ "путьвхождения". Нажата клавиша Ctr-F9. Источники приемов вводятся либо при автоматическом синтезе приема по некоторой теореме из базы теорем, либо при обработке приемов, введенных вручную и имеющих теорему "обычного", не технического характера. Последнее делается для создания обучающего материала, на котором развивается генератор приемов. Разумеется, при регистрации в базе теорем теорема приема преобразуется к виду логически корректного утверждения.

   Переменной x10 присваивается терм "прием(...)", ссылающийся на текущий прием. Переменная x12 инициализируется нулем; если узел приема имеет логический терминал "теорема" (в нем регистрируется ссылка на теорему - источ-

ник приема), то этой переменной переприсваивается содержимое данного терминала. Вводится комментарий (теорема x10 0 x12) к посылкам исходной задачи, который при ненулевом x12 направит траекторию отката на просмотр теоремы x12, и во всех случаях позволит выбрать либо изменить выбор источника для приема x10. Собственно для отката вводится комментарий (новпозиция заменагруппы), определяющий переход к оглавлению базы теорем.

7. Переход к просмотру основной версии приема из буфера базы приемов.

   x5 - символ "частичныйответ". Нажата клавиша "о" (кир.). Прежде всего, выясняется, имеется ли в наборе x2 - содержимом логического терминала текущего концевого пункта оглавления базы приемов - терм "замечание(копия)", указывающий, что текущий прием зарегистрирован в буфере как новый.

   Если такой терм есть, то в буфере имеется лишь дублирующая ссылка на прием. Чтобы найти место в оглавлении базы приемов, из которого на прием делается основная ссылка, считывается содержимое x8 логического терминала "оглавление" известного узла приема. Текущий путь в оглавлении переустанавливается по x8; рассматриваемый прием выделяется в найденном концевом логическом терминале оглавления заголовком "текприем"; корректируются переменные x1, x2, x3, и выполняется откат к повторному заполнению набора x4.

   Если же терма "замечание(копия)" нет, то находится содержимое x8 логического терминала "конец" хранящейся в буфере версии приема. Из него извлекается ссылка x9 на основную версию приема; по ней находится ссылка x11 на узел приема; определяется содержимое x13 терминала "оглавление" этого узла, и далее выполняются действия, аналогичные первому случаю.

8. Переход от основной версии приема к версии, сохраненной в буфере базы приемов.

   x5 - символ "заменагруппы". Нажата клавиша "б". Проверяется наличие логического терминала "начало" узла приема. Если этот терминал есть, то из него извлекается ссылка x9 на версию приема, хранящуюся в буфере. Затем происходит переустановка данных на просмотр этой версии, аналогичная предыдущей команде. Если же терминала "начало" нет, то это может означать, что мы находимся в ветви новых приемов буфера, но данный новый прием был изменен либо удален, так что ссылка на него одновременно имеется из ветви измененных или удаленных приемов буфера. Для поиска этой ссылки реализуется просмотр всего буфера, за вычетом ветви новых приемов. Сама ссылка известна - она состоит из логического символа, за которым закреплен прием, номера его узла и заголовка, найти же нужно тот концевой пункт буфера, где она находится. В конце - действия по переключению на найденную точку буфера, аналогичные первому случаю.

9. Вход в оглавление программ для выбора точки прерывания при отладке компилятора.

   x5 - символ "текуровень". Нажата клавиша Ctr-F5. Данная команда часто используется при анализе работы компилятора ГЕНОЛОГа. Она позволяет выбрать в ветви оглавления программ, относящейся к компилятору, нужную точку, и запустить компилятор с остановкой (выходом в отладчик ЛОСа) при ее достижении. Так как обучение решателя регулярно требует пополения этого

языка и, следовательно, пополнения компилятора, команда является существенным элементом интерфейса редактора приемов.

Прежде всего, активируется 9-й информационный блок, в котором находится оглавление программ. Реализуется обращение к оглавлению программ и выбор с его помощью некоторого концевого пункта; x9 становится равно паре ссылок на текстовый и логический терминалы этого пункта. Из логического терминала извлекается терм "программа(A_1 A_2)", ссылающий на контрольную точку "прием(A_2)" программы логического символа A_1. Символ A_1 присваивается переменной x12; число A_2 - переменной x13. Это число преобразуется к формату последовательности цифр x14 (включая случай одноэлементной последовательности). Затем сбрасываются старые установки на трассировку и вводится новая - на прерывание при выходе на указанный выше оператор "прием(A_2)". После этого предпринимается обращение к компилятору ГЕНОЛОГа - оператору "новыйприем(...)". После компиляции - откат к переформированию набора x4.

10. Переход к просмотру следующего приема серии отобранных для просмотра приемов.

    x5 - символ "схемаидентификации". Нажата клавиша "ш". Команда используется в различных процедурах просмотра серии отобранных по какому-либо признаку приемов. Отбор состоит в создании комментария (подборнеизвестных A_1 A_2 A_3) к посылкам исходной задачи. У него A_1 - список ссылок на просматриваемые концевые пункты текущего оглавления (базы приемов либо теорем). Каждая ссылка здесь - набор, определяющий путь в оглавлении. В случае базы приемов в конце набора размещается ссылка "прием(...)" на выбранный для просмотра прием; в случае базы теорем - ссылка "теорема(...)". A_2 - некоторый вспомогательный логический символ, играющий роль метки типа просмотра; A_3 - текущая позиция в наборе A_1.

    Фактическая смена текущей позиции A_3 и переход к рассмотрению очередного приема (или теоремы) происходит при нажатии клавиши "ш" в соответствующем оглавлении. Поэтому программа проверяет наличие комментария (подборнеизвестных ...), создает комментарий (автоклавиатура ...) к посылкам исходной задачи, эмулирующий повторное нажатие клавиши "ш", и выходит из редактора приемов для последующего обращения к оглавлению базы приемов.

    Если комментария (подборнеизвестных ...) нет, но имеется комментарий (теоремаприема ...), то это означает завершение цикла рассмотрения списка приемов, автоматически сгенерированных по теореме из базы теорем. Тогда реализуется возвращение в базу теорем.

11. Регистрация текущего приема в буфере перехода.

    x5 - символ "Неизвестные". Нажата клавиша "минус". Текущий прием регистрируется в буфере перехода, роль которого играет комментарий (Неизвестные A_1 A_2 A_3) к посылкам исходной задачи. Как и обычно, здесь A_1 - логический символ, за которым закреплен прием; A_2 - номер его узла; A_3 - заголовок приема.

12. Переход к приему, указанному в буфере перехода.

    x5 - символ "первыйключ". Нажата клавиша "равно". По комментарию (Неизвестные ...) находится ссылка на прием, к которому нужно перейти. Вместо

этого комментария создается другой - ссылающийся на текущий прием. Далее выполняется стандартная цепочка действий по переустановке данных на просмотр приема по известной ссылке.

13. Вход в справочное оглавление ГЕНОЛОГа.

    x5 - один из символов "новыйузел", "конъюнктоперанд", "записьзадачи", "смещениепеременных", "посылки". Соответственно, нажаты клавиши Ctr-з, Ctr - у, Ctr - х, Ctr - ы, Ctr - с (все - кир.). Они соответствуют справочным оглавлениям: заголовков приемов, фильтров, указателей, информационных элементов текстового анализатора (для удобства отладки текстового анализатора это оглавление достижимо как из редактора программ ЛОСа, так и из редактора приемов ГЕНОЛОГа), типов элементов установок на синтез приема.

    Прежде всего, проверяется наличие в x4 элемента (просмотртерма 2), указывающего, что выделено окно заголовка приема. Если он есть, то в оглавлении заголовков находится пункт, соответствующий заголовку текущего приема, и его текст прорисовывается под нижней линией приема. Иначе - происходит обращение к просмотру нужного оглавления. По выходе из оглавления реализуется несложный интерфейс изменения выбранного концевого пункта данного оглавления. Этот концевой пункт для справочного оглавления содержит ссылку, фиксирующую связанный с ним логический символ или терм. Например, для оглавления заголовков приемов с каждым его концевым пунктом связан терм, определяющий общий вид заголовков рассматриваемого типа. По нажатии клавиши Enter происходит редактирование данной ссылки - ввод нового или изменение старого символа либо терма.

14. Возвращение в главное меню.

    x5 - логический символ "группировки". Нажата клавиша End. Если просмотр приема происходит из трассировки (что определяется по комментарию "точкапривязки" к посылкам исходной задачи, то вместо возвращения в главное меню осуществляется возвращение к интерфейсу трассировки. Предварительно текущий прием выделяется в концевом терминале оглавления символом "текприем". Если же просмотр приема происходит не из трассировки, то вводится комментарий (автоклавиатура ...), эмулирующий повторное нажатие клавиши End при выходе в оглавление базы приемов - для отката к главному меню.

15. Просмотр справочной информации о логическом символе.

    x5 - логический символ "усмчисло". Нажата клавиша "с". Реализуется обращение к процедуре "помощь", обеспечивающей стандартный просмотр информации о логическом символе, вводимом в окне диалога. После просмотра - откат к перерисовке.

16. Обращение к справочнику по системе.

    x5 - логический символ "стандменьше". Нажата клавиша F1. Предпринимается обращение к процедуре "Помощь", обеспечивающей работу со справочником.

17. Просмотр теорем, полученных с помощью данного приема вывода.

    x5 - логический символ "минимум". Нажата клавиша "л". Команда относится только к приемам с заголовком "теорема". Она позволяет перейти к просмотру тех теорем заблаговременно установленного в базе теорем текущего раздела,

которые были выведены с помощью данного приема вывода. При просмотре теорем раздела отбираются те, комментарии к которым имеют ссылку "прием(...)" на текущий прием. Ссылки на отобранные теоремы регистрируются в комментарии (подборнеизвестных ...); для переключения на базу теорем вводится комментарий (новпозиция заменагруппы), и для входа в просмотр первой из отобранных теорем вводится комментарий (автоклавиатура ...), эмулирующий при входе в базу теорем нажатие клавиши "ш".

18.4.2 Просмотр окон приема

Начинать просмотр описываемых ниже ветвей следует с фрагмента, соответствующего пункту ("Редактор ГЕНОЛОГа", "Просмотр окон приема", "Вход в цикл выбора окна") оглавления программ. Так как многие операции запускаются из просмотра элементов описания приема, то эти ветви имеют сравнительно большую глубину и содержат ряд вспомогательных обработчиков команд.

1. Вход в цикл выбора окна.

   x5 - символ "внешконъюнкция". Нажата клавиша "курсор вправо". В действительности эта клавиша используется не только для входа в цикл выбора окна - через нее при уже выбранном окне осуществляется переход к просмотру содержимого окна, и в данной ветви программы содержатся также обработчики команд просмотра окон.

   Вход в цикл выбора окна осуществляется при отсутствии в наборе x4 элемента (просмотртерма ...). В этом случае находится номер x7 первого прорисованного окна, в x4 заносится элемент (просмотртерма x7), номер этого окна перекрашивается в малиновый цвет, и откат к вводу очередной команды.

   Если окно уже выбрано, т.е. элемент (просмотртерма ...) имеется, то переход через "иначе 2". Переменной x7 присваивается номер выбранного окна. Из элемента (видео ...) извлекается структура данных x9 для прорисовки этого окна; проверяется, что она отлична от 0. Дальнейшие действия, в зависимости от номера x7, описываются в следующих подпунктах данного раздела.

2. Вход в просмотр подтермов теоремы приемы.

   Действия продолжают предыдущий пункт для случая x7 = 1. Прежде всего, активная рамка прорисовки устанавливается на весь экран по высоте, но с отступлением шестнадцатипиксельного столбца слева.

   Если структура данных x9 для прорисовки теоремы указывает на текстовый режим прорисовки, то определяется строка x10, начиная с которой будет прорисовываться теорема (при наличии чертежа она сдвигается вниз). Затем теорема выдается текстовым редактором, причем цвет фона - голубой. Если в терме имеется вхождение, которое нужно выделить многоцветной указкой, то оператор "входвтерм(...)" создает комментарий (автоклавиатура ...), эмулирующий необходимые нажатия клавиатуры. Далее происходит обращение к оператору "просмотртерма(...)", который и реализует интерфейс просмотра подтермов.

   В рамках этого интерфейса предусмотрена обработка нажатия клавиши пробела - тогда по выходе из оператора "просмотртерма" режим выделения окон сбрасывается. Сначала в фрагменте, достижимом из текущего по "ветвь 3",

восстанавливается черно-белая текстовая прорисовка теоремы, затем происходит откат к запросу новой команды, где комментарий (автоклавиатура ...), эмулирует нажатие клавиши "курсор влево".

Если при работе с оператором "просмотртерма" применяется мышь, и левая кнопка ее нажимается вне зоны терма, то происходит переключение на просмотр выделенного мышью элемента другого окна приема. Для этого после выхода из данного оператора и перехода к фрагменту, достижимому по "ветвь 5", создается комментарий (автоклавиатура ...), инициирующий при откате сначала ввод команды "пробел", отменяющей выделение окна, а затем повторную обрабтку той же самой команды "мышь".

В случае прорисовки теоремы формульным редактором - переход через "иначе 2" после оператора "равно(левпозиция(x9 1)терм)". Прежде всего, переменной x10 присваивается набор пар (корень дерева указателей формульного редактора, соответствующего максимальному собственному подтерма теоремы приема, который может быть выделен формульным редактором - вхождение этого подтерма). Обычно такими максимальными подтермами являются антецеденты и консеквент теоремы. Инициализируется стэк x11 просмотра подтермов теоремы приема. В него заносятся пары (A_1 A_2), соответствующие уровням просмотра подтермов. На каждом уровне A_2 - набор пар (корень дерева указателей для одного из подтермов текущего уровня - вхождение этого подтерма в теорему); A_1 - вхождение в A_2, выделяющее текущий подтерм данного уровня. Начало набора x11 соответствует уровню текущего выделенного на экране подтерма, конец - корневому уровню теоремы. Далее идет цикл автоматического продолжения стэка x11 для выделения указанного в структуре данных x9 подтерма, если таковой имеется. Затем - переход через "ветвь 2", где располагается оператор "повторение", к которому будут предприниматься откаты после выбора очередного подтерма. Здесь, во-первых, текущий подтерм регистрируется в структуре данных x9 как выделенный для просмотра. Это позволит восстановить его, если понадобится временно переключиться с просмотра теоремы на какой-либо другой интерфейс. Затем предпринимается перекраска оператором "блокредактора" выделенного подтерма в малиновый цвет.

После перехода через "ветвь 1" попадаем в фрагмент программы, содержащий обработчик команд, получаемых оператором "автоменю(x18)". Здесь рассматриваются следующие команды:

а) x18 - символ "внешсумма". Нажата клавиша "курсор вверх" для выхода в надтерм из текущего подтерма. По достижении корня теоремы - выход из ее просмотра;

б) x18 - один из символов "извлечениеварианта", "обл". Нажаты клавиши "т" либо "пробел". Осуществляется выход из просмотра теоремы;

в) x18 - символ "нормпроизведениевсех". Нажата клавиша "ф". Находится указатель "фикс(...)" текущего вхождения в теорему, который регистрируется в комментарии (теквхожд ...) к посылкам исходной задачи. Эта команда используется при редактировании третьего либо четвертого окон приема: после нажатия в текстовом редакторе клавиши Shift-1 создается комментарий (теквхожд ...), происходит автоматическое переключение на просмотр теоремы приема, затем - выполняется данная команда, и далее - автоматическое возвращение в текстовый редактор с прорисовкой в нем найденного указателя "фикс(...)";

г) x18 - символ "подстановка". Нажата клавиша "курсор вниз" для входа в просмотр операндов выделенного подтерма;

д) x18 - символ "тринадцать". Нажата клавиша Enter для редактирования нормализаторов приема, связанных с текущим выделенным вхождением в теорему.

Заметим, что нажатия клавиш "ф" и Enter аналогичным образом обрабатываются и при просмотре теоремы в текстовом режиме - там это делает оператор "просмотртерма".

3. Вход в просмотр заголовка приема.

   В этот фрагмент попадаем из предыдущего через "иначе 1", расположенное после оператора "равно(x7 1)". Если заголовок односимвольный, то никаких действий не выполняется. Иначе - происходит перекраска его в голубой цвет, вход в оператор "просмотртерма", а по завершении просмотра - восстановление черного цвета заголовка.

4. Вход в просмотр третьего либо четвертого окон приема.

   Попадаем в этот фрагмент из предыдущего аналогичным образом - через "иначе 1" после оператора "равно(x7 2)". Переменным x10 и x11 присваиваются, соответственно, набор термов, которые прорисованы в окне, и набор координат (столбец - строка) начальных точек их прорисовки. Переменной x12 присваивается набор, указывающий для каждого терма из x10 выделенный подтерм, если он есть, и 0 в противном случае. Фактически такой выделенный подтерм может быть только один. Если у терма из x10 имеется выделенный подтерм, то он перерисовывается в черно-белом варианте - для устранения следов старых выделений подтермов. Переменной x13 присваивается позиция текущего выделяемого терма окна - изначально это первый разряд в x10. При наличии комментария (теквхожд ...), указывающего тот подтерм окна, к которому сразу нужно перейти, создается комментарий (автоклавиатура), эмулирующий нужное количество нажатий клавиши "курсор вправо" для выбора этого терма, завершающееся нажатием клавиши "курсор вниз". Затем - переход через "ветвь 1".

   В начале нового фрагмента располагается оператор "ветвь 1", ведущий к завершающему просмотр окна подфрагменту; далее расположен оператор "повторение", к которому будут происходить откаты при смене выделенного терма окна. Этот терм прорисовывается на голубом фоне черным цветом. Далее идет еще один оператор "повторение", и за ним - оператор "автоменю(x17)", вводящий команды для работы с текущим выделенным термом. Дадим краткое описание его команд:

   а) x17 - один из символов "циклвариантов", "внешконъюнкция", "внешсумма", "обл". Соответственно, нажаты клавиши "курсор влево", "курсор вправо", "курсор вверх", "пробел". Восстанавливается черно-белая прорисовка текущего терма. Если нажата клавиша "курсор вверх", то реализуется возвращение из просмотра окна в режим выбора окна; если нажата клавиша "пробел", то реализуется выход в общий просмотр приема; для остальных двух случаев реализуется переход к следующему либо предыдущему терму окна.

   б) x17 - один из символов "циклвариантов", "внешконъюнкция", "точкапривязки", "контрольунификации". В первых двух случаях снова нажаты клавиши

"курсор влево-вправо", но теперь они приводят к смене страницы окна, так как перед (соответственно, после) текущим термом расположена запятая. В последних двух случаях для смены страницы нажимаются клавиши PageDn, PageUp. Смена страницы выполняется за счет коррекции имеющегося в x4 элемента (См ...), указывающего, сколько нужно пропустить запятых для выхода на текущую страницу. После такой коррекции - откат к перерисовке всего изображения приема, с предварительным указанием в комментарии (автоклавиатура ...) нажатия клавиши номера окна для повторного входа в него.

в) x17 - символ "подстановка". Нажата клавиша "курсор вниз" для входа в просмотр подтермов текущего выделенного терма. Этот просмотр, как и для текстовой прорисовки теоремы приема, осуществляется оператором "просмотртерма". Соответственно, команды для различных операций с выделенными подтермами (например, для получения справочной информации) инициируются тоже из оператора "просмотртерма". По выходе из оператора "просмотртерма" анализируются введенные им комментарии (нормализация ...) и (теквхожд ...) к посылкам исходной задачи, хранящие дальнейшие инструкции о выполнении действий, начатых через его интерфейс.

г) x17 - один из символов "новыйузел", "конъюнктоперанд", "записьзадачи", "посылки". Этим символы вводятся нажатиями клавиш Ctr-з, Ctr-у, Ctr-х, Ctr-с (все - кир.). Команда осуществляет поиск в справочном оглавлении соответствующего типа - заголовков, фильтров, указателей и установок на синтез приема конструкций, имеющих своим заголовком тот же символ, что и выделенный в окне терм. Найденная информация прорисовыается снизу, под изображением приема. Если имеются несколько различных вариантов, то для смены их предусмотрен простейший обработчик команд.

д) x17 - символ "внутрвывод". Нажата клавиша "у" для регистрации текущего выделенного фильтра в буфере фильтров, роль которого играет комментарий (новоеусловие ...) к посылкам исходной задачи. Накапливаемые в этом буфере фильтры могут быть впоследствии одним нажатием клавиши "у" при общем просмотре какого-либо приема присоединены к его списку фильтров.

е) x17 - символ "мышь". Нажата одна из кнопок мыши. Если кнопка - левая и курсор мыши находится на каком-либо подтерме того же самого либо другого окна, то создается комментарий (автоклавиатура ...), эмулирующий последовательность нажатий клавиш курсора, необходимую для перехода к нужному окну и выделения в нем терма, содержащего выбранный подтерм. Затем предпринимается откат для реализации этой последовательности нажатий. Если подтерм был некорневым, то после входа в просмотр выбранного терма, осуществляемый оператором "просмотртерма", эмулируется повторный ввод команды "мышь". В результате формируется еще один комментарий (автоклавиатура ...) для оставшихся нажатий клавиш курсора, выделяющих подтерм цветовой указкой.

Если левой кнопкой была выделена запятая, то для смены страницы корректируется элемент (См ...) набора x4, и откат к переформированию этого набора.

Если левой кнопкой была выбрана пустая зона, то реализуется выход в общий режим просмотра.

Наконец, если была нажата правая кнопка мыши, то вводится комментарий, эмулирующий нажатие клавиши Ctr-у в случае третьего окна и клавиши Ctr-х

в случае четвертого - для извлечения из соответствующего справочного оглавления информации о выделенном в окне служебном символе. Это нажатие реализуется при откате к оператору, запрашивающему очередную команду x17.

ж) x17 - символ "числитель". Нажата клавиша "p" для разрезания текущей страницы окна на две новых страницы. В описании приема находится нужная точка для вставки запятой. Для фильтров рассматриваются операнды терма "условие(...)"; в список указателей запятая вставляется непосредственно. Изменения регистрируются в терминале "команда". Затем формируется комментарий, эмулирующий нажатие клавиши для повторного входа в окно после отката (т.е. клавиши номера окна), и реализуется откат к точке комплектования набора x4.

з) x17 - символ "фиктопер". Нажата клавиша F6 для склейки текущей страницы окна со следующей страницей. Действия аналогичны предыдущей команде, но запятая удаляется.

5. Непосредственный переход к просмотру заданного окна.

   x5 - один из символов "унисборка", "если", "то", "комплексныечисла". Нажата одна из цифровых клавиш 1,2,3,4, определяющая номер окна. Переход к просмотру содержимого окна происходит сразу, без использования номеров окон как пунктов меню. Для этого в набор x4 заносится элемент (просмотртерма ...), указывающий номер окна, и вводится комментарий, эмулирующий нажатие клавиши "курсор вправо". После отката будут выполняться описанные выше действия, начиная с пункта 1.

6. Присоединение содержимого буфера фильтров к списку фильтров текущего приема.

   x5 - символ "внутрвывод". Нажата клавиша "y". Находится комментарий (новоеусловие A) к посылкам исходной задачи. Если он имеется, то A представляет собой список термов, которые будут добавляться к фильтрам приема. Расширенный список фильтров регистрируется в информационном элементе (прием ...) набора x4, но в файлах изменения пока не осуществляются. Зато в информационном элементе (изменение ...) набора x4 делается пометка о том, что список фильтров был изменен. Такая пометка приведет к тому, что при перерисовке цвет нижней горизонтальной линии будет заменен на голубой. После этого для фактического изменения приема достаточно будет нажать F3. По завершении указанных коррекций - сброс элементов (видео ...), (конец ...), и откат к перерисовке.

7. Сброс буфера фильтров.

   x5 - символ "подборзначений". Нажата клавиша "Y" (кир.). Удаляется комментарий (новоеусловие ...).

8. Удаление либо восстановление прорисовки на экране окна с заданным номером.

   x5 - один из символов "метаперевод", "провменьшеилиравно", "попытка", "Текзадача". Соответственно, нажаты клавиши Ctr-F1, Ctr-F2, Ctr-F3, Ctr-F4. Находится номер x6 удаляемого либо восстанавливаемого окна. Если в x4 не было элемента (пассив x6), то он вводится (окно пропадает), иначе - удаляется (окно восстанавливается). Затем сбрасываются элементы (видео ...), (конец ...), и откат к перерисовке.

9. Смена режима просмотра "текст - формула".

   x5 - либо символ "извлечениеварианта", либо символ "нормпроизведениевсех". Соответственно, нажаты клавиша "т" либо "ф". При входе в текстовый режим просмотра создается комментарий "терм" к посылкам исходной задачи; при входе в формульный режим просмотра этот комментарий удаляется. Перед откатом к перерисовке сбрасываются элементы (видео ...), (конец ...), (просмотртерма ...) набора x4.

10. Вход в цикл просмотра нормализаторов.

    x5 - символ "усмотрениефильтра". Нажата клавиша "5". Переменной x7 присваивается список нормализаторов приема - термов "быстрпреобр($A\,B_1 \ldots B_n$)". У каждого такого терма A - либо нормализуемый подтерм, либо указатель вхождения его в теорему; $B_1, \ldots, B_n$ - список указателей нормализации для A. Переменной x8 присваивается текущее вхождение в x7 (изначально - вхождение первого элемента). После перехода через "ветвь 2" попадаем в фрагмент, начинающийся с оператора "повторение". Откаты к этому оператору будут выполняться при переходе к очередному нормализатору. Переменной x10 присваивается текущий нормализатор. Далее инициализируются переменные x11, x12, x13. Им будут присвоены координаты вхождения терма A в описание приема: x11 - номер окна (1, 3 либо 4), внутри которого выделяется терм с A; x12 - терм, содержащий A; x13 - вхождение в него подтерма A. После выполнения этих присвоений - переход через "ветвь 1".

    Здесь проверяется, что x13 отлично от 0, т.е. что вхождение A удалось найти. Если это не так, то предпринимаются предварительные действия по удалению текущего нормализатора из описания приема. Он удаляется из той версии описания приема, которая хранится в наборе x4; в элементе (изменение ...) этого набора регистрируется факт изменения описания приема, и откат к перерисовке с удаленным элементом (видео ...). После перерисовки нижняя линия приема становится голубой, что является сигналом для регистрации изменения в файле (например, путем нажатия F4).

    Если x13 было не равно 0, то прежде всего предпринимается просмотр всех выделенных согласно элементу (видео ...) подтермов и перерисовка их в черно-белом варианте - для удаления предыдущих цветовых указателей. После этого - переход через "иначе 2". Здесь находится строка, по которой проходит нижняя отделяющая линия приема, и под ней прорисовываются последовательно указатели нормализации $B_1, \ldots, B_n$. Затем - переход через "ветвь 1". По x11, x12, x13 находится то место на экране, где прорисован терм A, и выполняется цветовое выделение этого терма. После перекраски - переход через "ветвь 1". Далее идет небольшой фрагмент, анализирующий наличие комментария (теквхожд ...), и после него через "ветвь 1" - фрагмент, с которого начинается обработчик команд цикла просмотра нормализаторов.

    Предусмотрена обработка команд x16 следующих типов:

    а) x16 - один из символов "группировки", "контрольунификации", "точкапривязки". Соответственно, нажаты клавиша End для обрыва цикла просмотра, и клавиши PageUp - PageDn для перехода к предыдущему либо следующему нормализатору.

б) x16 - символ "группировка". Нажата клавиша "н" для редактирования текущего списка указателей нормализации. Вносимые здесь изменения регистрируются только во внутренних элементах набора x4; чтобы зарегистрировать их в файле, нужно после выхода из цикла просмотра нормализаторов нажать F3.

в) x16 - символ "модули". Нажата клавиша Ctr-Del для удаления текущего нормализатора. Фактическое удаление произойдет только при регистрации изменений в файле по выходе из цикла просмотра.

В двух последних случаях при изменении приема нижняя линия не меняет свой цвет на голубой - нажимать клавишу F3 или F4 здесь нужно и без этого напоминания.

18.4.3 Редактирование приема

Создание нового приема

1. Начало создания нового приема либо копирование старого.

   x5 - один из символов "замещение", "арктангенс", "переобозначение". Соответственно, нажаты клавиши "д", "Ctr-д", "Ctr-п". В первом случае выполняется копирование приема и его компиляция, во втором - копирование без регистрации в файлах и компиляции, в третьем - создается бланк нового приема.

   Прежде всего, блокируется выполнение команд, если текущий пункт оглавления базы приемов - пункт его буфера, содержащий новые приемы. Выполнение их блокируется также из трассировки либо из просмотра базы приемов с помощью процедуры "текприем".

   Далее проверяется наличие в начале набора x2, отображающего содержимое текущего концевого логического терминала оглавления базы приемов, терма "фикс(0)". Этот терм и играет роль бланка приема. Если он уже есть, то команда копирования отменяется, а в случае ввода нового приема указатель x3 текущей позиции набора x2 переустанавливается на начало набора, и откат к повторному формированию набора x4. Так как вместо ссылки "прием(...)" на позиции x3 расположен терм "фикс(0)", то экран останется пустым, и далее при пустом экране будет осуществлен переход к обработчику команд для ввода приема.

   Если набор x2 не имеет своим началом терм "фикс(0)", то этот терм заносится в начало x2, а x3 устанавливается на левый конец x2. В случае копирования приема вводится комментарий (регистрациятеоремы ...), сохраняющий все переносимые в новый прием элементы набора x4. Если копирование должно сопровождаться компиляцией, то вводится также комментарий (автоклавиатура ...), эмулирующий нажатие клавиши F3 после копирования. При этом создается комментарий (замечузел ...), в который переносятся списки комментариев к теореме текущего приема и к самому приему. После перечисленных действий - откат к повторной инициализации набора x4.

2. Регистрация ссылки на узел теоремы в буфере для создания нового приема.

   x5 - символ "префикснаярекурсия". Нажата клавиша Ctr-б. Вводится комментарий (утверждение A_1 A_2 A_3 A_4), где A_1 - логический символ, за которым закреплена теорема приема; A_2 - ссылка на узел теоремы приема; A_3 - номер узла;

A_4 - теорема приема. Этот комментарий играет роль буфера теоремы приема: при создании нового приема можно воспользоваться старым узлом теоремы, на который ссылается данный буфер.

3. Извлечение из буфера теоремы при создании нового приема.

   x5 - один из символов "Обобщподст", "выражение". Соответственно, нажаты клавиша "Ctr-к" либо "Ctr-и" (кир.). В первом случае по старой теореме создается копия ее узла, во втором - используется старый узел теоремы.

   Прежде всего, проверяется, что x3 указывает на терм "фикс(0)", т.е. имеет место работа с пустым бланком приема. Затем находится комментарий (утверждение ...), и из него извлекается набор (логический символ - узел теоремы приема - номер узла - теорема), присваиваемый переменной x8. Проверяется отсутствие в наборе x4 элемента (логсимвол ...), означающее, что ссылки из этого набора на какую-либо теорему приема еще нет. Затем в левом верхнем углу экрана прорисовывается номер окна теоремы - "1". В набор x4 заносятся элементы (логсимвол ...) и (терм ...), а в случае "Ctr-и" также элемент (адрес-узла ...). Все эти элементы формируются на основе информации, хранящейся в x8. Далее выполняется прорисовка теоремы (формульным либо, при наличии комментария "терм", текстовым редактором), проводится нижняя отделяющая линия, и откат к запросу очередной команды.

Редактирование окон приема

1. Вход в редактирование теоремы формульным редактором.

   x5 - символ "транслвыражения" либо "Неизвестная". Соответственно, нажаты клавиша "Ctr-ф" либо "Ф". В первом случае будет происходить либо набор новой теоремы, либо набор новой версии теоремы под старой версией (сохраняемой на экране, чтобы было удобнее вносить изменения). Во втором случае происходит восстановление режима редактирования старой версии на завершающем моменте ее набора - курсор формульного редактора располагается в конце теоремы, и можно либо продолжать набор, либо прибегнуть к Backspace для отката.

   После цепочки стандартных блокировок (нет трассировки, текущий раздел - не в буфере базы приемов, и т.п.) переменной x6 присваивается номер строки, начиная с которой будет выполняться редактирование теоремы.

   Переменной x7 сначала присваивается единица. Это значение будет передано формульному редактору как цвет символов (черный). Однако, если была нажата клавиша "Ф", то используется специальный режим работы формульного редактора. В этом режиме переменной x7 переприсваивается старая версия теоремы приема, причем вводится комментарий "формоперанды" к посылкам исходной задачи. Тогда формульный редактор будет сначала создавать дерево указателей для прорисовки старой версии x7, затем прорисует эту версию, удалит комментарий "формоперанды" и перейдет в режим ручного продолжения набора теоремы.

   Собственно обращение к формульному редактору - в следующем фрагменте; перед обращением предпринимается расчистка экрана вниз от строки x6. Если произошел отказ от редактирования, то откат к перерисовке приема.

К результату редактирования x9 сразу же применяется процедура "коррекция-обозначений". Она нужна для того, чтобы уточнить внутреннее представление ряда обозначений, совпадающих при формульной прорисовке, но различающихся как логические символы. Например, одним и тем же символом "плюс" на экране прорисовываются несколько различных операций - обычное сложение вещественных чисел ("плюс"), сложение комплексных чисел ("Плюс"), сложение функций ("плюсфунк"), сложение векторов ("плюсвект"), и т.п. Чтобы уточнить, какая именно операция имеется в виду, небходимо учитывать контекст. При наборе задачи роль этого контекста играют все утверждения списка посылок и условий. Поэтому коррекция обозначений в задаче происходит лишь после ее полного набора - перед началом решения либо при нажатии Home. В случае теоремы приема контекст беднее - он сводится лишь к ней самой. Соответственно, больше возможностей столкнуться с ситуацией, в которой он не позволяет однозначно восстановить смысл набранных обозначений - тогда уточнения в теорему приема придется вводить вручную с помощью текстового редактора.

Далее выполняется такое переобозначение связанных переменных теоремы, чтобы никакой квантор либо описатель не связывал переменную, уже связанную внешним квантором либо описателем. После этого - переход через "ветвь 2", где выполняется дополнительное переобозначение связанных переменных в подвыражениях с производными. Наконец, происходит регистрация теоремы в элементе (терм ...) набора x4, учет в элементе (изменение ...) факта изменения теоремы, отбрасывание элементов (видео ...), (конец ...), и откат к перерисовке.

2. Вход в редактирование теоремы текстовым редактором.

   x5 - символ "кортеж". Нажата клавиша Ctr-т (кир.). После цепочки блокировок определяется номер x6 строки, начиная с которой будет происходить редактирование. Затем расчищаются буфер текстов и экран. При наличии комментария (теквхожд ...) происходит выдача терма, указанного в нем. В противном случае, если набор x4 содержит элемент (терм ...), осуществляется прорисовка этого терма. Далее - переход через "ветвь 2" к фрагменту, начинающемуся с оператора "повторение". Откаты к этому оператору будут происходить при ошибочном наборе терма. При обращении к текстовому редактору сохраняется прорисованная на экране исходная версия терма. После редактирования применяется оператор "кодтекста", переводящий текст в терм x8. Если он обнаруживает ошибку, то информация о ней прорисовывается в нижней части рамки редактирования, и откат к повторному редактированию. Иначе - действия аналогичны случаю ввода теоремы формульным редактором, но без какой-либо коррекции обозначений.

3. Вход в редактирование 2,3 либо 4-го окна.

   x5 - один из символов "учетпоказателя", "полныепосылки", "теоремы". Соответственно, нажаты клавиши Ctr-2, Ctr-3 либо Ctr-4. После цепочки блокировок переменной x6 присваивается номер того окна, для которого выполняется редактирование. Если этот номер не равен 2, причем окно заголовка приема еще не создано, то команда блокируется. Переменной x8 присваивается структура данных для прорисовки окна, указанная в элементе (видео ...). Переменной x10 присваивается номер строки, с которой будет выполняться редактирование.

Если отсутствует комментарий (теквхожд ...), указывающий, что редактирование продолжается после временного выхода наружу - в справочное оглавление либо в первое окно для полуавтоматического ввода указателя вхождения "фикс(...)", - то экран расчищается начиная со строки x10, причем восстанавливается прорисовка под этой строкой номера текущего окна. Расчищается буфер текстов.

Если имеется комментарий (теквхожд ...), то осуществляется определяемая им дорисовка текста - добавление извлеченного из справочного оглавления терма с помощью оператора "возвращтекста" (он также прорисовывает в нижней части экрана сопровождающий текст к выбранному терму), либо добавление указателя вхождения "фикс(...)". Если этого комментария нет, то в окне перерисовываются все его термы - они извлекаются из структуры данных x8. В обоих случаях далее - переход через "ветвь 2".

Здесь размещается оператор "повторение", откаты к которому будут осуществляться при повторном входе в текстовый редактор после обнаружения ошибки редактирования. Вводятся комментарии "учетпоказателя" и "префиксныйфильтр x6" к посылкам исходной задачи, позволяющие текстовому редактору правильно реагировать на те нажатия клавиш, которые должны восприниматься в контексте данного окна приема. Например, с их помощью будет формироваться комментарий (теквхожд ...), обеспечивающий возвращение в окно после переключения на теорему либо на справочное оглавление. После обращения к текстовому редактору эти комментарии удаляются.

Если имел место отказ от редактирования, то переход через "иначе 1". Если редактировалось старое окно, то откат к перерисовке; если окно было новым ($x8 = 0$), причем номер его был более 2, то происходит регистрация в элементе (видео ...) пустой структуры данных текущего окна и проводится его нижняя линия, отстоящая от верхней на 3 пикселя. Для третьего окна вводится комментарий (автоклавиатура ...), эмулирующий нажатие клавиши входа в редактирование четвертого окна. Это делается для удобства ввода нового приема - чтобы не отвлекаться на промежуточные между редактированием окон нажатия клавиш.

Если редактирование успешно завершено, то проверяется наличие комментария (теквхожд ...) - оно означает необходимость выполнения внешнего переключения, и тогда происходит откат к запросу новой команды. При отсутствии такого комментария набранный текст переводится в логический формат x12 - терм в случае второго окна и список термов либо символов в случае третьего и четвертого окон. Если здесь обнаруживается ошибка, то информация о ней прорисовывается в нижней части экрана, и откат к повторному редактированию. При отсутствии ошибки выполняется ввод либо коррекция элементов (заголовок ...), (условие ...), (терм ...). Если окно было разбито на страницы, то изменение затрагивает только текущую страницу описания фильтров либо указателей, которая определяется по элементу (См ...). После такой коррекции - переход через "ветвь 2".

Здесь корректируется элемент (изменение ...), указывающий окна, в которых был изменен прием. Если редактировалось старое окно, отличное от последнего окна, то откат к перерисовке. Иначе предпринимается регистрация в элементе (видео ...) структуры данных для прорисовки текущего окна, новое содержи-

мое окна перерисовывается со стандартными пробелами между термами, проводится нижняя отделяющая линия, а в случае, если окно не было последним - создается комментарий (автоклавиатура ...), эмулирующий нажатие клавиши перехода к следующему окну. Затем - откат к запросу новой команды.

4. Удаление нормализаторов.

   x5 - символ "конст". Нажата клавиша Ctr-End. Из набора x4 удаляется элемент (быстрпреобр ...); из элемента (прием ...) исключаются все нормализаторы "быстрпреобр(...)"; корректируется элемент (изменение ...), цвет нижней линии приема изменяется на голубой, и откат к запросу очередной команды.

5. Редактирование чертежа.

   x5 - символ "дробнаявеличина". Нажата клавиша Ctr-ч. После стандартных блокировок создается бланк x6 структуры данных геометрического редактора. Если в x4 уже имелся элемент (геомредактор ...), то x6 переопределяется по этому элементу. После редактирования чертежа определяется указатель x7, показывающий, нужно ли набирать теорему формульным (x7=0) либо текстовым (x7=1) редактором. Для этого служит символ "терм", который заносится во второй набор пары x6 при завершении редактирования чертежа не по Enter, а по Ctr-Enter. Определяется нижняя из строк чертежа x8. Далее предпринимается коррекция в наборе x4 элемента (геомредактор ...). Если теорема приема уже имеется, то откат к перерисовке, иначе - вводится комментарий (автоклавиатура ...), эмулирующий нажатие клавиши, переводящей в редактирование теоремы формульным (x7=0) либо текстовым редактором.

6. Выбор нового логического символа для закрепления за ним теоремы приема.

   x5 - символ "подфрагмент". Нажата клавиша Ctr-й. Процедура перезакрепления приема за другим логическим символом бывает нужна достаточно редко. К ней можно обращаться, если новый прием возник в результате переделки копии старого приема, причем теорема его изменилась настолько, что нет смысла связывать прием с символами старой теоремы. Другой повод для использования данной команды - разгрузка слишком часто используемых логических символов, во избежание переполнения одного из файлов 8-го информационного блока. Несмотря на кажущуюся простоту операции, для ее выполнения приходится полностью переписывать все структуры данных приемов рассматриваемой теоремы и перекомпилировать эти приемы.

   Программа выполняет следующие действия. После цепочки стандартных блокировок находится строка x7 под первым окном, экран ниже этой строки расчищается, прорисовывается текст "логсимвол :", расчищается буфер текстов, и предпринимается обращение к текстовому редактору для ввода того символа x10, за которым нужно закрепить прием. Если узел теоремы приема еще не был введен, то выполнение команды сводится к коррекции на новый символ элемента (логсимвол ...). Иначе вводится новый узел статьи символа x10. Переменной x13 присваивается ссылка на этот узел; переменной x14 - номер узла. Просматриваются все заголовки x20 приемов, основанных на текущей теореме, и программы этих приемов удаляются. После этого - переход через "иначе 6".

Вся ветвь узла текущей теоремы копируется из нового корня x13. Это делается специальной процедурой "новаяветвь". Так как корень x13 может оказаться относящимся к другому файлу информационного блока, то переключить ссылку из него на старый экземпляр ветви, вообще говоря, нельзя. После копирования ветви происходит удаление ветви старого узла теоремы. Затем просматриваются узлы приемов в новой ветви, и предпринимаются обращения к процедуре "новыйприем", компилирующей эти приемы. При компиляции очередного приема осуществляется коррекция ссылки на него из текущего концевого пункта оглавления базы приемов. По завершении просмотра узлов приемов аналогичная коррекция ссылок выполняется в наборе x2.

7. Изменение теоремных переменных приема.

   x5 - символ "высотаполосы". Нажата клавиша "K" (кир.). Команда позволяет переобозначить переменные связывающей приставки теоремы, проводя необходимые коррекции во всех компонентах описания приема.

   Включается формульный редактор, с помощью которого под нижней строкой приема набираются через запятую те переменные, на которые должны быть переобозначаены переменные связывающей приставки. Переменной x12 присваивается набор этих переменных. Переменной x14 присваивается теорема приема; переменной x15 - ее связывающая приставка. Находится список x20 тех переменных, встречающихся в теореме приема, ее фильтрах, указателях либо нормализаторах, которые входят также в список x12. Для них выбираются новые обозначения x23. Далее (переход через "ветвь 2") происходит переобозначение старых переменных на новые во всех компонентах описания приема, представленных в наборе x4 - в его теореме, фильтрах и указателях. Факт изменения этих компонент регистрируется в элементе (изменение ...). Вводится комментарий (автоклавиатура ...), эмулирующий нажатие клавиши F4, и откат к запросу очередной команды.

8. Просмотр и редактирование текстов, сопровождающих срабатывание приема.

   x5 - символ "транслоперандов". Нажата клавиша "6".

   Если узел приема не имеет текстового терминала "титр", то сразу же создается комментарий (автоклавиатура ...), эмулирующий нажатие клавиши Enter - оно будет переводить в редактирование текста. Находится строка x8 под изображением приема. Затем - переход через "ветвь 2".

   После оператора "повторение" (к нему будут происходить откаты после выхода из текстового редактора) происходят расчистка экрана начиная со строки x8 и расчистка буфера текстов. Затем - снова оператор "повторение", и после него - оператор ввода команды редактирования x9. Здесь возможны следующие случаи:

   а) x9 - символ "тринадцать". Нажата клавиша Enter для входа в текстовый редактор. Сразу же после редактирования предпринимается сохранение новой версии текста в терминал "титр" узла приема.

   б) x9 - символ "модули". Нажата клавиша Ctr-Del для удаления ранее введенного терминала "титр".

   в) При нажатии любой другой клавиши экран ниже строки x8 расчищается, и откат к запросу очередной команды редактора приемов.

9. Просмотр и редактирование текстовых примечаний к приему.

   x5 - символ "старшийчлен". Нажата клавиша "э". Для редактирования используется процедура "текстблок". Она позволяет просматривать и изменять текстовые терминалы, достижимые по меткам 1,2, ... из указателя, к которому от узла приема имеется переход по метке "текстблок".

Удаление приема

x5 - символ "модули", что соответствует нажатой клавише Ctr-Del. После цепочки стандартных блокировок проверяется наличие в x4 элемента (просмотртерма ...), указывающего на выделенное окно приема.

Если такой элемент есть, то переход через "иначе 3". В случае выделенного третьего либо четвертого окна приема происходит, соответственно, удаление из описания приема всех фильтров либо всех указателей. Удаление затрагивает лишь набор x4, и после него выполняется откат к перерисовке.

Если окна приема не выделены, то начинается удаление приема. Прежде всего, оператор "длялюбого(x12 x13 ...)" удаляет его программу.

Если узел приема имеет логический терминал "конец", то это означает, что удаляется прием, находящийся в буфере базы приемов и либо являющийся старой версией измененного приема, либо находящийся в "корзине" буфера после удаления его из основной части базы приемов. В этом случае будет происходить не перенесение приема в буфер, а фактическое его удаление. Соответствующие действия начинаются после контрольной точки "прием(5 8)". Сначала здесь проверяется наличие у приема источника в базе теорем; если такой источник есть, то обратная ссылка из него на прием удаляется. Далее - переход через "ветвь 6". Здесь анализируется содержимое терминала "конец". Если в нем имеется ссылка на прием, то мы имеем дело со старой версией измененного приема. Тогда удаляется встречная ссылка из основной версии приема, хранящаяся в его терминале "начало". Затем удаляется ветвь узла приема; если после этого теорема приема оказывается неиспользуемой, то удаляется и ее узел. Далее - откат к переходу через "ветвь 4" в исходном фрагменте команды, где выполняется удаление ссылки на прием из концевого терминала оглавления; предпринимается выбор в этом терминале другой текущей ссылки, и откат к формированию набора x4. Новая текущая ссылка берется идущей сразу после удаленной; если удалена последняя ссылка, то новая текущая ссылка будет последней из оставшихся. Такой выбор удобен при удалении нескольких приемов, расположенных подряд. В заключение заметим, что удаление конкретного приема из буфера практически не применяется. Обычно весь буфер сбрасывается целиком с помощью нажатия клавиши "о" из оглавления базы приемов. Эта команда интерфейса оглавления реализована аналогично вышеописанному.

Если узел приема не имеет терминала "конец", то переход через "иначе 5". Здесь сначала проверяется, имеет ли узел приема терминал "начало". Если такой терминал есть, то это означает, что удаляется новая версия ранее измененного приема. Так как старая версия сохранена в буфере, то можно сразу же удалить узел приема для новой версии, что и делается. Если при этом узел теоремы приема оказывается неиспользуемым, то он тоже удаляется. Затем - переход через "ветвь 2", где по содержимому x13 терминала "начало" определяются ссылки x15, x16 на узлы теоремы и приема для старой версии. Находится терминал "конец" старой версии, содержимое которого заменяется на символ "пусто". Это сразу переводит старую версию из категории "измененные приемы"в категорию "удаленные приемы", однако с точки зрения ссылок

из оглавления она по-прежнему остается в категории "измененных приемов". Чтобы перевести ее в раздел буфера "удаленные приемы", предпринимается просмотр буфера для поиска либо создания его ветви "удаленные приемы". Ссылка на корень этой ветви присваивается переменной x25. После этого - переход через "ветвь 2", где продолжается работа с оглавлением. Прежде всего, проверяется, заполнена ли ветвь удаленных приемов целиком. Она содержит не более 30 приемов, и в случае наличия 30-го приема (переход к нему через символьный номер "элементарно", соответствующий числу 30) предпринимается удаление остатков этого приема. Затем - переход через "ветвь 1", где происходит увеличение на 1 меток оставшихся пунктов ветви буфера. Одновременно корректируются встречные ссылки на оглавление, хранящиеся в терминалах "оглавление" узлов приемов. После этого- переход через "ветвь 1" и определение содержимого терминала "оглавление" старой версии приема, согласно ее новому размещению - она будет иметь номер 1 в ветви удаленных приемов. Наконец, предпринимается регистрация ссылки на старую версию по метке 1 в ветви буфера и переносится ее сопровождающий текст из ветви "измененные приемы". Далее - откат к переходу через "ветвь 1" после оператора "пунктыоглавления(...)". Здесь завершается реконструкция оглавления: из ветви "измененные приемы" исключается переход к терминалу старой версии; номера меток последующих пунктов уменьшаются на 1, и корректируются встречные ссылки из логических терминалов "оглавление" узлов приемов. Далее - откат к уже упоминавшемуся выше завершающему фрагменту, достижимому по "ветвь 4" из исходного фрагмента обработки команды.

Наконец, остается случай, когда узел удаляемого приема не имеет ни терминала "начало", ни терминала "конец", т.е. прием не был за последнее время изменен и не находится в буфере. Обработка этого случая начинается с фрагмента, переход к которому - через оператор "иначе 1" после проверки наличия терминала "начало". Первым оператором фрагмента служит метка "прием(5 7)", которой соответствует пункт "Регистрация удаляемого приема в буфере базы приемов". После проверки того, что текущий концевой терминал оглавления не имеет комментария "копия", указывающего на расположение его в ветви буфера "Новые приемы", начинается поиск самого буфера. Если буфера не было, то он создается; в результате ссылка на корневой указатель ветви буфера оказывается присвоена переменной x15. Затем - переход через "ветвь 1". Здесь предпринимается поиск ветви удаленных приемов; при осутствии такой ветви она создается. Переменной x17 присваивается ссылка на корень ветви, и переход через "ветвь 1".

Далее практически полностью воспроизводятся действия, имевшие место при перенесении старой версии приема из ветви "Измененные приемы" в ветвь "Удаленные приемы": если из указателя x17 имеется переход по метке 30, то соответствующий прием удаляется; номера всех меток в x17 увеличиваются на 1; удаленный из основной части решателя прием регистрируется по метке 1; корректируется его возвратная ссылка на оглавление, и откат к переходу через "ветвь 4" для завершения команды.

Компиляция приема

1. Регистрация приема в базе приемов.

   x5 - либо символ "узелраздела", либо символ "копияблокаанализа". Соответственно, нажаты клавиша F3 либо F4. В первом случае кроме регистрации приема в базе приемов происходит также его компиляция. После предварительного анализа ситуации (блокировка команды для обращений не из редактора

приемов и пр.) - переход через "ветвь 2". Здесь проверяется наличие у приема чертежа; если чертеж есть, а теорема приема уже была ранее зарегистрирована и имеет свой узел, то предпринимается обновление терминала "геомредактор", хранящего описание чертежа. Далее - переход через "ветвь 1".

Переменной x8 присваивается описание приема, извлеченное из элемента (прием ...) набора x4; это описание пополняется нормализаторами из элемента (быстрпреобр ...). Если была нажата клавиша F3, то заблаговременно вводится комментарий (автоклавиатура ...), эмулирующий нажатие клавиши F5 - обращение к компиляции после того, как прием будет зарегистрирован. Затем - переход через "ветвь 1".

Здесь проверяется наличие в наборе x4 элемента (адресузла ...), ссылающегося на узел теоремы.

Если такого элемента нет, то переход через "иначе 1" к фрагменту, создающему новые структуры данных приема (это делает оператор "добавлениеприема(x4 x9)") и корректирующему текущий терминал оглавления для ссылки на новый прием. Оператор "добавлениеприема" создает новый узел теоремы; регистрирует ссылку на прием в ветви "Новые приемы" буфера базы приемов; регистрирует чертеж приема; создает узел приема и регистрирует относящиеся к нему данные; регистрирует ссылку на прием в текущем логическом терминале оглавления базы приемов и создает возвратную ссылку из узла приема на оглавление.

Если узел теоремы имеется, то проверяется наличие в x4 элемента (команда ...), ссылающегося на узел приема.

Если узла приема нет, то переход через "иначе 2". Здесь проверяется, была ли изменена теорема приема. Если она была изменена, то создается новый узел теоремы (для того же самого логического символа), и измененная версия теоремы регистрируется в этом узле. Затем создается узел приема, для которого вводятся терминалы "команда" (описание приема) и "оглавление" (возвратная ссылка на оглавление); ссылка на прием регистрируется в текущем терминале оглавления, и откат к перерисовке. Если теорема не изменена, то узел приема вводится для уже имеющегося узла теоремы.

Наконец, если узел приема имелся, то происходит регистрация измененного приема. Сначала проверяется, имеется ли в x4 элемент (изменение ...), Если его нет, то выполнение команды отменяется. Иначе - начинается реализация ветви, начинающейся с метки "прием(5 9)", соответствующей пункту оглавления программ "Регистрация изменяемого приема в буфере базы приемов".

Прежде всего, проверяется наличие ссылки на прием из ветви буфера "Новые приемы". Если такая ссылка имеется, то изменяемый прием сохраняется в данной ветви и не переносится в ветвь "Измененные приемы". Наличие ссылки обнаруживается операторами в конце текущего фрагмента программы. В этом случае оператор "сброс(десять)" обеспечивает откат к точке "ветвь 4", переводящей в ветвь программы, изменяющей структуры данных приема. Если же ссылка не обнаружена, то переход через "ветвь 5", где выполняются действия по регистрации приема в ветви "Измененные приемы". Они начинаются со ввода узла приема, копирующего старую версию (см. оператор "новыйузел(x14 x15 x16)"). Регистрация этой копии в ветви буфера происходит так же, как рассмотренная выше регистрация приема в ветви удаленных приемов.

После перехода через "ветвь 4" проверяется, изменена ли теорема. Если это так, то корректируется терминал "терм" ее узла. Коррекция выполняется оператором "регистрациятеоремы", который, кроме изменения терминала, учитывает также изменение чисел вхождений в теоремы приемов логических символов. Эти числа используются при выборе точек привязки программ приемов и хранятся в терминалах "мощность" корней статей логических символов 8-го информационного блока.

Если других изменений в описании приема нет, то элемент (изменение ...) набора x4 удаляется, и выполнение команды завершается. Иначе - проверяется, изменен ли заголовок приема. Если он изменен, то удаляется программа старой версии приема; удаляется старый узел приема, и вводится новый, соответствующий новому заголовку. В нем регистрируется новое описание приема, после чего - откат к перерисовке. Если же заголовок не изменен, то происходит коррекция терминала "команда" узла приема, хранящего его описание, и тоже откат к перерисовке.

2. Компиляция приема.

   x5 - один из символов "фиктопер", "прогрблок". Соответственно, нажаты клавиши F6 либо F5. В первом случае сохраняется информация о компиляции, позволяющая при отладке идентифицировать программные переменные с теоремными переменными; во втором (обычном) случае такая информация не сохраняется.

   После цепочки блокировок и извлечения из набора x4 необходимых для дальнейшего элементов происходит удаление терминала "переменные" узла приема, сохраняющего информацию о соответствии программных и теоремных переменных. Эта информация будет восстановлена лишь в случае нажатия F6. Так как она нужна лишь при отладке, то ее разумно "по умолчанию" удалять. Затем предпринимается обращение к процедуре "новыйприем", которая и реализует компилятор ГЕНОЛОГа. Это - очень большая процедура, и изложению общей схемы ее организации будет посвящена вся следующая глава. Сразу заметим, что размеры компилятора не позволят нам пройти его "от начала до конца" столь же подробно, как это делалось для других реализованных на ЛОСе процедур - во многих случаях уточнение подробностей будет предоставляться читателю. Основная задача, которую должно будет решить описание компилятора - позволить быстро находить те его участки, в которые нужно вносить изменения при развитии ГЕНОЛОГа.

3. Удаление программы приема.

   x5 - логический символ "простоенеравенство". Нажата клавиша F7. Сначала проверяется наличие логических терминалов "контроль" узла приема, в которых содержатся ссылки на программы справочника "контроль", созданные для приема. Эти программы не имеют "своей" теоремы и являются дополнениями к программе текущего приема, вынесенными в отдельные фрагменты. Они нужны для проверки условий отката к ситуации, имевшей место до применения приема, если по прошествии заданного числа шагов после применения не выполняется заданное условие. Если такие программы есть, то все они удаляются оператором "удалениеприема" (несмотря на название, этот оператор удаляет не прием, а только программу). Затем удаляется программа самого приема.

Заметим, что во всех случаях удаления либо изменения программы происходит обращение к оператору "прогфайл(компонента A)", где A - тот логический символ, к которому относилась программа. Такое обращение приводит к удалению из зоны программ фрагментов старой версии программы символа A и восстановлению в каталоге программ ссылки на корень программы, хранящийся в файле и уже измененный. Без этой коррекции система могла бы еще некоторое время пользоваться старыми фрагментами измененной программы, что приводило бы к ошибкам. Чтобы данную операцию можно было реализовать, обращение к изменяющей программу процедуре не должно размещаться внутри процесса выполнения изменяемой программы.

4. Восстановление исходной версии приема по буферу базы приемов.

   x5 - символ "редакторответа". Нажата клавиша "Б". Происходит полное восстановление того вида приема, который имел место до зарегистрированного в буфере изменения, и его перекомпиляция.

   Из терминала "начало" узла приема извлекается ссылка "прием(...)" на исходную версию приема. Переменным x10 , x11 и x12 присваиваются, соответственно, логический символ, номер узла этого символа и заголовок приема, определяемые по найденной ссылке. Переменной x13 присваивается ссылка на узел теоремы для исходной версии приема. Далее происходит перезапись теоремы из терминала "терм" узла исходной версии x13 в соответствующий терминал узла теоремы текущей версии. Сразу после этого такая же перезапись выполняется для терминалов "команда", хранящих описания приемов. Если прием имеет чертеж, то из исходной версии в текущую перезаписывается также описание чертежа. Далее - переход через "ветвь 2".

   Здесь предпринимается перекомпиляция текущей версии приема и удаляется терминал "начало" ее узла приема. Набор информационных элементов x4 сбрасывается. Если зарегистрированная в буфере исходная версия была откомпилирована, то ее программа удаляется. Удаляется узел ее приема; если после этого узел теоремы оказывается неиспользуемым, то он тоже удаляется. Из оглавления ветви буфера исключается концевой пункт, соответствующий восстановленному приему. Чтобы восстановить сплошной характер нумерации, используемый в метках оглавлений, предпринимается вычитание единицы из меток, следующих за удаленной. При этом корректируются возвратные ссылки на оглавление из соответствующих узлов приемов. Далее - откат к переформированию набора x4.

5. Откат либо восстановление исходной версии программы приема, для которого буфер приема хранит старую версию.

   x5 - либо символ "верхняягрань", либо символ "число". Соответственно, нажаты клавиши "И", "Т" (кир.). В первом случае происходит компиляция программы исходной версии приема и удаление программы текущей версии, во втором - компиляция программы текущей версии и удаление программы исходной. Структуры данных самих версий приема при этом не изменяются.

   Переменной x11 присваивается набор (логический символ - номер узла - заголовок приема - ссылка на узел теоремы - ссылка на узел приема) для просматриваемой версии приема. Заметим, что эта версия может быть как текущей, так и старой, если просмотр ведется из буфера базы приемов. Переменной x12

присваивается 0 - эта переменная играет роль индикатора изменений. Если выполнение команды оказывается невозможным, то она сохраняет нулевое значение, иначе становится равна 1. Далее инициируется пара x13, первый элемент которой полагается равным 0, а второй - набору x11. После цикла коррекций, на первой позиции этой пары оказывается набор, соответствующий исходной версии приема (она в буфере), а на втором - набор, соответствующий текущей версии. Если просматривается удаленный прием, сохраненный в буфере, то второй элемент пары x13 будет равен 0.

Затем выполняется переход через "ветвь 3", где сначала удаляется программа одной из версий (в зависимости от нажатой клавиши), а затем (переход через "ветвь 1") компилируется другая версия. В обоих случаях индикатору x12 присваивается единица. Далее - сначала откат к переходу через "ветвь 2", и если x12 оказывается равно 1, то набор x4 сбрасывается и выполняется откат к его переформированию.

Автоматический синтез приема или его компонент

Архитектура средств, развиваемых для автоматического синтеза приемов по базе теорем, будет рассматриваться в третьем томе монографии. Хотя эти средства пока недостаточны для саморазвития системы, все же они частично используются при ручном обучении решателя. Собственно говоря, одной из главных целей развития решателя и является накопление необходимого фактического материала для исследования принципов автоматического развития базы приемов. Поэтому пока основная часть интерфейса, связанного с автоматическим синтезом приемов, обслуживает не поток информации "от теорем к приемам", а встречный поток - "от приемов к теоремам". Последний дает возможность проследить шаг за шагом на конкретных примерах весь процесс возникновения приема из теоремы и отобрать для базы теорем те теоремы, которые фактически оказываются востребованы при синтезе приемов.

Здесь мы ограничимся кратким описанием интерфейса запуска основных процедур генератора приемов, сообщая о них минимально необходимые общие сведения.

1. Просмотр и редактирование установок на синтез приемов.

   Заголовок приема ГЕНОЛОГа определяет тип выполняемых приемом преобразований, но ничего не говорит о цели этих преобразований. Такая цель лишь косвенным образом извлекается из всей совокупности фильтров и указателей приема. Чтобы сопроводить приемы более явными указаниями тех целей, для которых они были созданы, а заодно предпринять классификацию фактически накопленных в решателе приемов согласно этим целям, был создан специальный интерфейс. Он позволяет развивать оглавление типов целей приемов и связывать с каждым приемом соответствующий пункт такого оглавления. Во многих случаях более точная формулировка целей применения приема требует сопровождения типа цели некоторым списком параметров. Создается впечатление, что в действительности указание теоремы приема, типа его цели и сопровождающих параметров вполне достаточно для автоматического доопределения полного описания приема на ГЕНОЛОГе. Такое доопределение предполагает как анализ уже имеющихся теорем и приемов для оптимизации взаимодействия их с новым приемом, так и возможные циклы коррекций приема на сериях задач. Фактически, здесь возникает язык еще более высокого уровня,

чем ГЕНОЛОГ. Описание приема на нем требует, помимо теоремы, лишь заданной в некотором стандартном формате целевой характеризации. Эта характеризация (тип цели и ее параметры) была названа установкой на синтез приема, а процедуры компиляции, обеспечивающие ее преобразование в описание приема на ГЕНОЛОГе, названы генератором приемов. На самом деле, генератор приемов лишь завершает работу по синтезу новых приемов, так как появлению его входной информации предшествуют процессы еще более высоких уровней, происходящие уже не в базе приемов, а в базе теорем. Мы не будем касаться сейчас трудностей, возникающих при создании генератора приемов - речь о них пойдет в третьем томе. Опишем вкратце лишь упомянутый выше интерфейс, позволяющий вручную вводить установки на синтез уже имеющихся приемов. Эта, безусловно, бессмысленная при автоматическом синтезе приемов работа пока что совершенно необходима для предварительной классификации приемов и получения "задачника" для процедур предыдущего уровня - тех, которые по теореме определяют установку на синтез приема.

Переходим к рассмотрению программы интерфейса, реализующей данную команду. Начальный ее фрагмент соответствует пункту (Редактирование приема - Автоматический синтез приема или его компонент - Просмотр и редактирование установок на синтез приема).

x5 - логический символ "новыесвязки". Нажата клавиша "и". Установка на синтез приема хранится в логическом терминале "примечание" узла приема. Содержимое этого терминала присваивается переменной x8. Из него извлекается элемент "тип(A)", где A - логический символ, кодирующий тип установки. Будем называть его далее типом приема.

Типы приемов представляют собой логические символы, к которым обращается справочник "заголовокприема", инициирующий создание приема. Все приемы этого справочника зарегистрированы в оглавлении программ, в разделе (Генератор приемов - Справочник ЗАГОЛОВОКПРИЕМА). Название концевого пункта оглавления программ, соответствующего заданному типу приема, используется в интерфейсе редактора приемов вместо обозначающего тип логического символа.

После того, как тип приема присвоен переменной x10, оператор "названиетипа" присваивает переменной x11 ссылку на текстовый терминал оглавления программ, хранящий название типа. Происходит расчистка экрана ниже текста теоремы приема, и на этом месте осуществляется прорисовка указанного текстового терминала. Затем проводится горизонтальная линия, и под ней прорисовываются остальные элементы установки на синтез. После прорисовки сразу выполняется вход в текстовый редактор. Если нужно изменить параметры установки (тип ее изменяется другими средствами), то по окончании редактирования нажимается Enter, иначе - нажимается Esc. В первом случае корректируется содержимое терминала "примечание".

2. Автоматический синтез приема либо характеризация приема с помощью оглавления типов приемов.

   x5 - символ "деление" либо "нижняяоценка". Соответственно, нажаты клавиша "a" либо "x". В первом случае предпринимаетя попытка синтеза приема после ручного ввода теоремы либо создание альтернативной версии уже имеющегося

приема, со своей установкой на синтез. Для этого обеспечивается вход в оглавление типов приемов и ручной выбор нужного типа. Эта команда пока используется только с одной целью - для автоматического формирования приемов справочников, причем нужна она бывает достаточно часто. Во втором случае (клавиша "x") предпринимается вход в оглавление типов приемов, чтобы связать с приемом установку на синтез приема либо изменить ранее связанную с ним установку. После того, как с приемом связана установка на его синтез, он превращается в обучающий материал для развития генератора приемов: можно запускать на нем генератор и анализировать расхождения между ручным и автоматическим синтезом.

Обработка команды начинается с проверки того, что при нажатии клавиши "x" прием уже создан. Если ранее выполнялась тестовая генерация приема и был создан комментарий (сравнениеприемов ...), хранящий результаты сравнения синтезированного приема с его "ручной" версией, то этот комментарий удаляется. Удаляется также другой относящийся к автоматическому синтезу приемов комментарий - (смнабор ...). Переход через оператор "иначе 2" относится к случаю инициализации оператора фильтра и сейчас для нас неинтересен. Далее происходит ввод комментария "заголовокприема" к исходной задаче, указывающий, что имеет место режим входа в оглавление программ для выбора типа приема. Оператор "путьвоглавлении(программа набор(7 1)x7)" устанавливает текущий путь в оглавлении программ на корень ветви справочника "заголовокприема", которая одновременно служит оглавлением типов приемов. Далее реализуется вход в оглавление программ. Если выход из него осуществляется по нажатии клавиши Esc либо End, то откат к перерисовке. Иначе в выбранном концевом терминале оглавления находится терм "программа(A_1 A_2)", где A_1 - логический символ, программа которого, относящаяся к справочнику "заголовокприема", инициализирует конвейер синтеза приема. A_2 - номер выбранной для ссылок контрольной точки "прием(...)" в этой программе. Символ A_1 присваивается переменной x13.

Дальнейшая обработка двух команд выполняется раздельно. Рассмотрим сначала случай нажатия клавиши "a"(x5 = "деление"). На экране восстанавливается исходное изображение - то, которое имелось перед обращением к оглавлению. Вводится заготовка x15 установки на синтез приема - в нее заносится терм "тип(x13)".

Если создаваемый прием не является приемом справочника, то переменной x16 присваивается ссылка на текстовый терминал оглавления для выбранного концевого пункта. Этот текст прорисовывается под текстом теоремы приема, и осуществляется обращение к текстовому редактору для ввода оставшихся элементов установки на синтез, которые присоединяются к набору x15. Далее - откат к переходу через "ветвь 6". Если же создается прием справочника, то пополнение установки на синтез не требуется, и сразу выполняется переход через "ветвь 6".

Здесь переменной x16 присваивается пара (приемы пустоеслово), второй элемент которой представляет собой накопитель четверок (теорема приема - заголовок приема - установка на синтез приема - описание приема) для синтезируемых приемов. Инициализируется структура данных x17 генератора приемов, в которую заносится пока единственный элемент x16. Эта структура данных называется блоком приема. Если происходит создание новой версии уже

имеющегося приема, то предпринимается попытка найти в базе теорем ту теорему, которая является источником приема. В x17 при этом дополнительно заносится элемент (теорема ...) - ссылка на найденную теорему.

Далее - переход через "ветвь 1", где происходит обращение к справочнику "заголовокприема", выполняющему синтез приемов. В действительности его программы обычно весьма просты - они являются лишь началом конвейера синтеза приема. Далее, по цепочке взаимных обращений, следуют операторы "схемапосылок", "схемаидентификации", "схеманормализации", "фильтрыприема" и "учетприема". Описание их будет приведено в третьем томе монографии. После обращения к генератору приемов анализируется список x19 четверок указанного выше формата, определяющих созданные приемы. Переменной x20 присваивается первая четверка этого списка.

Если выполняется синтез альтернативной версии имеющегося приема, то переход через "иначе 2", где к началу списка x2 ссылок на приемы текущего просматриваемого пункта оглавления добавляется элемент "фикс(0)", x3 заменяется на вхождение этого элемента, а x4 переустанавливается на просмотр приема, задаваемого набором x20. Затем - откат к перерисовке. Так как этот прием не зарегистрирован в файлах, то цвет нижней его линии будет голубым. С другой стороны, наличие элемента "фикс(0)" означает, что все готово для регистрации, и при необходимости ее можно осуществить обычными средствами. Если набор x19 был более чем одноэлементным, то его остаток сохраняется в комментарии (смнабор ...) к посылкам исходной задачи.

Если же происходит синтез приема после ручного ввода теоремы, то осуществляется коррекция указанной в наборе x4 теоремы приема - она берется такой, как в четверке x20. Это нужно из-за того, что генератор приемов может ввести в теорему технические модификации. Затем к набору x4 присоединяются компоненты описания приема, извлекаемые из четверки x20, и откат к перерисовке. Как и в предыдущем случае, остаток набора x19 сохраняется в комментарии (смнабор ...).

Вернемся теперь к точке раздельного рассмотрения команд и перейдем к рассмотрению случая клавиши "x"(см. оператор "иначе 5"). Здесь начинается формирование набора x14 элементов установки на синтез приема. Прежде всего, в него заносится терм "тип(x13)". В зависимости от описания текущего приема, в x14 заносятся также: для приемов замены - терм "направл(A)", указывающий направление замены A; для приемов проверочного оператора либо синтезатора - терм "оператор(A)", указывающий название A этого оператора. Далее набор x14 регистрируется в элементе (примечание ...) набора x4 и в логическом терминале "примечание" узла приема. Наконец, создается комментарий (автоклавиатура ...), эмулирующий нажатие клавиши "и" для входа в ручное доопределение установки на синтез, и происходит откат к точке ввода команд.

3. Тестовая генерация приема.

   x5 - символ "область" либо "прообраз". Соответственно, нажаты клавиша "Л" либо "П". Первая команда используется при отладке генератора приемов: она позволяет установить нужную точку прерывания в его работе через оглавление программ; вторая - запускает генератор приемов без прерываний. В обоих случаях создается некоторая версия описания приема, которая выдается на экран для сравнения со старой версией. Для удобства сравнения предусмотрен ряд

несложных вспомогательных команд. Команда тестовой генерации приема может запускаться в цикле серийной обработки всех приемов выбранного раздела. В базе приемов сохраняется информация о последнем и предпоследнем циклах тестовой генерации приемов; предусмотрен интерфейс для просмотра этой информации по заданному разделу оглавления. При просмотре указываются количества рассогласований различных типов между автоматически созданными приемами и приемами, хранящимися в решателе; можно переходить к просмотру групп приемов, в которых эти рассогласования имеются. Таким образом, имеется интерфейс для работы над сближением "ручной" и автоматической версий приемов. В отдельных случаях для такого сближения приходится доучивать генератор приемов, в отдельных - изменять старые приемы. Фактически, работа над генератором приемов для решателя оказалась одним из наиболее эффективных средств оптимизации самого решателя. Но вернемся к рассмотрению программы, реализующей команды.

Переменной x8 присваивается установка на синтез приема; если была нажата клавиша "Л", то осуществляются обращение к оглавлению программ, выбор в этом оглавлении концевого пункта для прерывания работы генератора приемов и создание установки на прерывание при выходе на соответствующую контрольную точку "прием(...)". Для обоих команд далее - переход через "ветвь 3". Переменной x12 присваивается накопитель блока приема, в который заносится накопитель результатов (приемы пустоеслово). Как и в случае команды "а", к блоку приема добавляется элемент (теорема ...), ссылающийся на источник приема в базе теорем. Далее - обращение к справочнику "заголовокприема", выполняющему синтез приемов.

По окончании работы генератора приемов просматриваются четверки x18 из накопителя новых приемов x11. Для отображения на экране выбирается та из них, заголовок приема которой совпадает с заголовком рассматриваемого приема решателя. Формируются тройки x22 и x23 вида (теорема приема - заголовок - описание приема), соответственно, для старой и новой версий приема. К ним применяется оператор "сравнениеприемов", анализирующий различия и выдающий список x24 указателей рассогласования между приемами. Типы этих указателей можно посмотреть в справочной информации к логическому символу "сравнениеприемов". В логическом терминале "сравнениеприемов" узла приема решателя сохраняются данные о числе различий в фильтрах, указателях и нормализаторах. Ранее имевшиеся в терминале данные о последнем тестировании переносятся в разряд предпоследнего тестирования, а те - удаляются. Затем информация о новом приеме и найденных различиях сохраняются в комментарии (сравнениеприемов ...), вводится комментарий (автоклавиатура ...), эмулирующий нажатие клавиши "р"(кир.), и откат к запросу очередной команды. При выполнении команды "р" произойдет переход к просмотру синтезированной версии приема (см. ниже).

Если же различий между новой и старой версиями не было, то оператор "сравнениеразличий" оказывается ложным. В этом случае нижняя линия приема перекрашивается в зеленый цвет, информация о тестировании сохраняется в терминале "сравнениеприемов", и переход к запросу очередной команды.

При наличии серийного тестирования, распознаваемого по комментарию "подраздел", происходит внутренний перезапуск системы для перехода к очередному приему.

4. Переходы между основной и автоматической версиями приема.

   x5 - символ "числитель". Нажата клавиша "р" (кир.). Команда позволяет попеременно переходить от просмотра основной версии приема к его версии, возникшей при тестировании генератора приемов, и обратно. При изображении на экране автоматической версии нижняя отделяющая линия имеет голубой цвет.

   Находится комментарий (сравнениеприемов ...), хранящий информацию об автоматической версии приема. Указателем на основную версию приема является наличие в наборе x4 элемента (команда ...), ссылающегося на узел приема. В этом случае набор x4 переформируется по комментарию "сравнениеприемов", вводится комментарий "точкапривязки", блокирующий на период просмотра автоматической версии операции по изменению базы приемов, и откат к перерисовке. Если же просматривалась автоматическая версия приема, то комментарий "точкапривязки" удаляется, и откат к переформированию набора x4.

5. Просмотр списка рассогласований между основной и автоматической версиями приема.

   x5 - символ ",". Нажата клавиша "Р" (кир.). Клавиша должна нажиматься при просмотре основной версии приема, иначе команда игнорируется. Под теоремой приема прорисовывается текущий указатель рассогласования (поясняющий текст и термы); смена этих указателей осуществляется нажатием клавиш "курсор вверх" - "курсор вниз". Выход из просмотра - по нажатии любой другой клавиши.

   Информация о рассогласованиях извлекается из комментария (сравнениеприемов ...) и присваивается переменной x8. Она представляет собой набор элементов, указывающих на появление нового либо исчезновение старого фильтра; указателя; нормализатора. Программная реализация описанного выше интерфейса просмотра элементов несложна, и ее можно при необходимости разобрать самостоятельно.

6. Автоматическое формирование указателей приема.

   x5 - символ "группировка". Нажата клавиша "н" (кир.). Эта команда (и еще команда "а" для автоматического формирования приемов справочников) - единственная на сегодня функция генератора приемов, реально используемая в обучении решателя. Она позволяет доопределить указатели и нормализаторы приема в полуавтоматическом режиме. Генератор приемов предлагает последовательно новые (созданные им, но ранее отсутствовавшие) либо старые (которых он не создал) элементы данных типов, и на каждом шаге достаточно лишь нажать клавишу Insert либо Delete для добавления либо удаления элемента.

   После цепочки стандартных блокировок располагается оператор "повторение". Затем проверяется наличие в наборе x4 элемента (указатель ...), содержащего информацию о предложениях генератора приемов по вводу либо удалению указателей и нормализаторов. Если такого элемента нет, то переход через "иначе 2". После его создания - откат к указанному выше оператору "повторение", и далее - продолжение действий по отображению серии предложений на экране.

   Элемент (указатель A_1 A_2) набора x4 содержит список A_1 пар (B_1 B_2), где либо B_1 - элемент, созданный генератором приемов и отсутствовавший ранее, а B_2 - 0, либо B_1 - 0, а B_2 - ранее имевшийся элемент, не созданный генератором. A_2 - текущая отображаемая на экране позиция списка A_1.

Начнем с рассмотрения ветви "иначе 2", где происходит создание комментария (указатель ...). Здесь начинается формирование блока приема x9. В него заносится накопитель результатов (приемы пустоеслово). Так как прием доопределяется, то набор x4 должен иметь элемент (прием ...), содержащий введенную часть описания приема. Это позволяет добавить к блоку приема элемент (условие ...), перечисляющий фильтры приема, и элемент (прием ...), перечисляющий прочие компоненты описания приема. В x9 заносится также элемент (текприем ...), содержащий ссылку на доопределяемый прием в базе приемов, и элемент (примечание ...) - установку на синтез приема, если она есть. Наконец, проверяется, что доопределяемый прием имеет заголовок, после чего выполняется обращение к середине конвейера генератора приемов - процедуре "схемапосылок". Начиная с этой точки и происходит доопределение блока приема x9.

После обращения к генератору приемов проверяется, что прием был доопределен однозначно (иначе выполнение команды отменяется). Переменной x12 присваивается список указателей и нормализаторов доопределенного приема, причем если указатель объединял в себе несколько различных подуказателей, то в данном списке он разбивается на части. Аналогичный список x13 создается для исходной версии приема. Указатели и нормализаторы, имеющиеся в обоих версиях, отбрасываются, а из оставшихся комплектуется список A_1 комментария (указатель ...). После ввода комментария - откат к оператору "повторение" для отображения на экране результатов обращения к генератору приемов.

При обнаружении комментария (указатель A_1 A_2), содержащего информацию о попытке доопределения приема, переменной x8 присваивается текущая пара (B_1 B_2) набора A_1. В зависимости от того, определяет ли эта пара элемент, созданный генератором либо не созданный им, в верхней части экрана активизируется одна из двух версий меню. В левом верхнему углу этого меню указывается, какой случай рассматривается. Переменной x9 присваивается указатель либо нормализатор, извлеченный из текущей пары.

Если x9 - нормализатор, то определяются значения x11,x12,x13 - соответственно, номер окна приема (1, 3 либо 4), в котором нужно выделить цветовой указкой нормализуемый подтерм; терм, к которому относится подтерм, и вхождение подтерма. Чтобы убрать цветовые элементы предыдущего состояния интерфейса, все термы первого, третьего и четвертого окон перекрашиваются в черно-белый цвет. Затем под нижней линией приема прорисовываются указатели нормализации, извлеченные из терма x9. Наконец, происходит перекраска того подтерма, к которому они относятся.

Если x9 - указатель, то, как и в предыдущем случае, сначала убираются цветовые элементы предыдущего состояния интерфейса, а затем указатель прорисовывается под нижней линией приема.

После прорисовки - откат к переходу через "ветвь 4" в исходном фрагменте команды "н". Здесь размещен обработчик команд, реагирующий на три команды: Esc - сброс режима просмотра предложений генератора приемов; Insert - введение нового либо сохранение старого элемента; Delete - удаление старого либо отказ от нового элемента. Все изменения затрагивают только набор x4; для регистрации их в файле необходимо после выхода из серии воспользоваться стандартными средствами - клавишами F3 либо F4. При удалении либо добав-

лении указателей сохраняется принятая в ГЕНОЛОГе группировка однотипных указателей в один общий указатель.

7. Извлечение установок на синтез, созданных генератором приемов.

   x5 - символ "помощь". Нажата клавиша "Ctr-ц". Команда используется в сочетании с интерфейсом базы теорем. После того, как генератор установок на синтез приемов, запускаемый из базы теорем, создает список возможных установок, и некоторая из них выбирается через специальный интерфейс, она регистрируется в комментарии (комментарий ...). При этом осуществляется автоматический переход из базы теорем в базу приемов, где нажатие Ctr-ц приводит к замене старой установки для текущего приема на новую, извлекаемую из указанного комментария.

8. Переход к рассмотрению очередного приема из серии автоматически созданных приемов.

   x5 - символ "схемаидентификации". Нажата клавиша "ш". Как и в предыдущем случае, команда завершает действия, инициируемые в базе теорем. Там выполняется просмотр серии установок на синтез приемов, созданных для текущей теоремы; одна из них выбирается, и по ней синтезируются приемы. Результаты синтеза - набор четверок (теорема приема - заголовок приема - установка на синтез приема - описание приема) - передаются в комментарий (смнабор A). После автоматического перехода в базу приемов, ручного выбора нужного концевого пункта и входа в просмотр его приемов нажимается клавиша "ш". После этого находится комментарий (смнабор ...) и из него извлекается первая четверка x9 указанного вида. К началу набора x3 добавляется, если его там не было, терм "фикс(0)" - бланк ссылки на новый прием; выбирается символ x10, за которым будет закреплен прием; первый элемент набора A отбрасывается; переменной x4 присваивается набор для отображения синтезированного приема x9, и откат к перерисовке. Таким образом обеспечивается серийный просмотр синтезированных приемов и ручной отбор их в базу приемов. Другие, более автономные процессы синтеза приемов через базу теорем будут рассмотрены в третьем томе монографии.

Перенесение приема на новое место

Перенесение приема из одного концевого пункта оглавления базы приемов в другой происходит в два этапа. Сначала ссылка на него заносится нажатием клавиши Insert (x5 - символ "внешдизъюнкция") в специальный буфер, роль которого играет комментарий (ссылканаприем A). Здесь A - набор пар (ссылка на концевой терминал оглавления - терм "прием(B_1 B_2 B_3)"). Затем, уже в оглавлении базы приемов, после выделения нужного концевого пункта, снова нажимается Insert, что и завершает процесс перенесения в данный пункт всех отобранных приемов. Реализацию этой, а также ряда других осуществляемых через интерфейс оглавлений операций редактора приемов мы рассмотрим ниже, в отдельном подразделе.

Буфер переносимых приемов можно расчищать нажатием клавиши Delete (x5 - символ "соответвхожд").

Обработчик команд мыши

При нажатии кнопки мыши из общего просмотра приема начинается обработка команды x5 = "мышь".

Если нажата правая кнопка мыши, то осуществляется выход из просмотра приема. Текущая ссылка "прием(...)" в наборе x3 заменяется на "текприем(...)", чтобы при следующем попадании в данный концевой пункт оглавления базы приемов снова оказаться на том же приеме. Это изменение регистрируется в файле. Далее - выход из редактора приемов. Если имел место цикл просмотра приемов процедурой "текприем", то при выходе устанавливается значение "ложь" - это приводит к возвращению в главное меню.

Если нажата левая кнопка мыши, то прежде всего определяется номер x11 окна приема, в котором находится курсор мыши. Если выделения окна приема на момент нажатия не было, то вводится комментарий (автоклавиатура ...), эмулирующий последовательное нажатие клавиши - номера окна (для входа в окно) и получение сигнала от мыши (параметры этого сигнала - номер кнопки и координаты курсора - берутся из регистров интерпретатора прежние). Если окно было выделено, то тоже вводится комментарий (автоклавиатура ...), но в этом случае он эмулирует последовательное нажатие клавиши "пробел" (для отмены выделения окна) и получение сигнала от мыши - для повторения данной команды. В обоих случаях далее - откат к запросу следующей команды.

Управление трассировкой через просмотр приема

Из просмотра приема можно вводить различные параметры процесса решения задачи либо получать дополнительную информацию при трассировке:

1. Ввод установки на прерывание при применении приема либо при начале попытки его применения.

   x5 - один из символов "список", "фигуры", "тринадцать". Соответственно, нажаты клавиша F9, либо Ctr-Enter, либо Enter. В первом случае должна быть введена установка на прерывание при начале попытки применения приема в кадре текущей задачи; во втором - на прерывание при начале попытки применения приема в произвольном кадре (наиболее употребительный вариант); в последнем случае - при фактическом применении приема.

   Проверяется наличие комментария "блоктрассировки", указывающего на режим просмотра приема из запуска решения задачи либо из трассировки. Если обращение к просмотру приема произошло из логического вывода в базе теорем (что распознается по наличию внешней процедуры "блоктеорем"), то установка на прерывание при попытке применения приема вводится сразу, и далее - выход из редактора приемов. Иначе - переход через "ветвь 2". Здесь, в зависимости от типа команды, вводится комментарий с одним из заголовков: "начало", "левыйкрай", "трассперечисл", содержащий координаты приема. Этот комментарий будет использован внешней процедурой для фактической установки прерывания. В случае команды Enter на уровне интерпретатора ЛОСа вообще не вводится каких-либо специальных установок - просто операторы, завершающие преобразования приема в случае его применения, будут учитывать наличие комментария (трассперечисл ...). Перед выходом из редактора приемов текущая ссылка "прием(...)" заменяется на "текприем(...)".

2. Ввод установки в буфере задачника на слежение за срабатываниями приема.

   x5 - либо символ "заменазнака", либо символ "достижимо". Соответственно, нажаты клавиша "C" либо "A" (кир.). В первом случае буфер задачника пополняется новым подразделом, в котором при решении задач будут сохраняться ссылки на те задачи, где сработал данный прием. Изначально этот раздел пустой; его заголовок воспроизводит заголовок концевого пункта оглавления приемов, к которому относится прием. Во втором случае тоже вводится такой подраздел, но он сразу заполняется всеми задачами текущего раздела задачника, в которых ранее прием срабатывал. Установка нужного раздела задачника должна быть выполнена до обращения к данной команде.

   Переменной x9 присваивается ссылка "прием(...)" на текущий прием. Затем в оглавлении задачника находится ветвь буфера; если ее нет, то она создается. Ссылка на корень этой ветви присваивается переменной x12. Просматриваются разделы буфера; если уже есть раздел, связанный с данным приемом (его терминал "замечание" содержит ссылку на прием), то откат к переходу через "ветвь 2". Иначе такой раздел вводится, и далее - откат к тому же пункту.

   После отката проверяется, что была введена команда "A". Если это не так, то действия завершается. Иначе - создается комментарий (список ...) к посылкам исходной задачи, содержащий ссылку на текущий прием. Он будет использоваться вспомогательной процедурой "буферзадачника", применяемой в цикле просмотра задач текущего раздела. Для ссылки x14 на очередную задачу выполняется обращение к процедуре "архивзадачи", определяющей в 10-м информационном блоке ветвь с информацией о приемах, применявшихся при последнем запуске ее решения. Если в этой ветви указан текущий прием, то создается фиктивная задача x20, содержащая ссылку x14, и осуществляется обращение к оператору "буферзадачника(x20 1)", заносящему ссылку на нее в тот раздел буфера, который соответствует текущему приему. Для определения текущего приема и нужен введенный выше комментарий (список ...).

3. Просмотр термов, идентифицированных с теоремными переменными.

   x5 - символ "анализатор". Нажата клавиша "п". Команда позволяет просмотреть термы, идентифицированные на текущий момент с теоремными переменными приема. Она применима, если прием был откомпилирован с помощью клавиши F6. Определяется ссылка x6 на внешний кадр глобального стэка; находится набор x7, описывающий этот кадр, и проверяется, что он является кадром оператора "прерывание" (т.е. отладчика ЛОСа). Находится содержимое x10 логического терминала "переменные" узла приема, возникающего при компиляции через F6. Этот терминал содержит пары (теоремная переменная - программное выражение для определения идентифицированного с ней терма). Переменной x12 присваивается ссылка на тот стэковый кадр, в котором происходит применение приема. Предпринимается заполнение накопителя x13 парами (теоремная переменная - идентифицированный с ней терм). Для определения терма по программному выражению x16 сначала находится список x18 значений всех его переменных, а затем применяется процедура "знач", находящая значение данного выражения по списку его переменных x17 и списку их значений x18. После получения списка x13 реализуется несложный интерфейс поочередного просмотра его элементов с помощью PageUp - PageDown.

4. Блокировка применения приема в задаче.

x5 - символ "базисвывода". Нажата клавиша Ctr-0. Предпринимается блокировка срабатываний текущего приема в задаче из задачника, при трассировке решения которой выполняется эта команда. Блокировка состоит в том, что ссылка "прием(A_1 A_2)" на теорему приема (A_1 - логический символ, A_2 - номер узла теоремы приема) заносится в логический терминал "блок" узла задачи. Тогда при следующем запуске решения этой задачи будет отключено с помощью оператора "трассировка(фильтр ...)" срабатывание всех приемов данной теоремы. Убрать блокировку можно из просмотра задачи до начала ее решения - для этого нужно нажать клавишу "Б", выделить в возникающем списке заблокированных теорем все теоремы, блокировка приемов которых отменяется, и нажать Esc.

18.5 Программы редактора приемов, запускаемые из интерфейса оглавлений

Ряд операций редактора приемов инициируется из оглавления базы приемов. В основном, пояснения о том, как они выполняются, уже были даны в разделе, посвященном процедуре "оглавление". Дадим несколько более подробные описания программ, реализующих две из таких операций - регистрацию отобранных приемов в текущем концевом пункте оглавления и запуск цикла просмотра раздела с обращениями к процедуре "текприем". Выход на начальные точки этих программ осуществляется через раздел ("Интерфейс оглавлений" - "Обработчик команд") оглавления программ. Текущая команда интерфейса оглавлений присвоена переменной x10.

1. Регистрация приемов из буфера в текущем концевом пункте оглавления базы приемов.

   x10 - символ "внешдизъюнкция". Нажата клавиша Insert, чтобы все отобранные в буфер приемы (они отбирались нажатием той же клавиши из просмотра описаний приемов) были извлечены со своих прежних позиций и перенесены в текущий концевой пункт.

   Прежде всего, проверяется наличие комментария (вид прием), который указывает, что текущее оглавление является оглавлением базы приемов. Та же самая команда может выполняться также в базе теорем, и переход к реализующей ее программе - через "иначе 2". Переменной x14 присваивается комментарий (ссылканаприем A), играющий роль буфера приемов. Из набора x5, перечисляющего данные о пунктах текущего меню оглавления, извлекается пятерка x15. Проверяется, что она соответствует концевому пункту (в этом случае x15 заканчивается символом 1). Буфер приемов сбрасывается. Просматриваются все пары x16 из набора A. Первый элемент каждой такой пары является ссылкой на концевой логический терминал, в котором находится терм "прием(...)" либо "текприем(...)", ссылающийся на прием, причем этот же терм является вторым элементом пары. При просмотре происходит исключение ссылок на приемы из старых терминалов оглавления. Затем - переход через "иначе 3".

   Переменной x16 присваивается ссылка на логический терминал, соответствующий текущему концевому пункту оглавления. В этом терминале регистрируются ссылки на все приемы набора A. Если какой-либо из перенесенных на новое место приемов представлял собой старую версию измененного приема,

хранящуюся в буфере базы приемов, то встречные ссылки из старой и новой версий отбрасываются - теперь это два никак не связанных между собой приема. Наконец, корректируются встречные ссылки на оглавление из терминалов "оглавление" узлов приемов.

2. Запуск цикла просмотра приемов процедурой "текприем".

   x10 - один из символов "элементы", "простоенеравенство", "нормвариант". Соответственно, нажаты клавиша F8, F7 либо Ctr-F8. В первом случае происходит полный просмотр всех приемов базы приемов ГЕНОЛОГа, с обращением для каждого из них к процедуре "текприем". Иногда такой просмотр бывает нужно прервать в некоторой точке, запомнивши ее нажатием клавиши "ю" на текущем приеме, а повторный просмотр начать с приемов, идущих после выделенного. Тогда используется клавиша F7. Наконец, чтобы найти все неоткомпилированные приемы, нажимается Ctr-F8. В последнем случае обращения к процедуре "текприем" не происходит - просто неоткомпилированные приемы один за другим выдаются на экран. Как и при просмотре через "текприем", переход к очередному приему вызывается здесь нажатием клавиши "курсор влево".

   Так как F8 применяется и в других оглавлениях (для теорем и задач), то прежде всего проверяется, что текущее оглавление является оглавлением базы приемов. Если была нажата клавиша F7, то находится логический терминал, достижимый из корневого указателя по меткам "прием", "начало". Он хранит ссылку "прием(...)" на прием, отобранный ранее при нажатии клавиши "ю". По этой ссылке формируется тройка x11 (логический символ - номер узла приема - заголовок приема), и переход через "ветвь 3".

   Переменной x12 присваивается одноэлементный набор, состоящий из пустого набора. Этот набор является накопителем замечаний, которые можно вводить при обращениях к оператору "текприем". Далее переменной x13 - указателю текущего просматриваемого логического символа - присваивается символ "метка", имеющий номер 1. Переменной x14 присваивается 0 - эта переменная будет при просмотре приемов определять текущий номер узла логического символа x13. Затем располагается оператор "повторение" - к нему будут происходить откаты при переходе к очередному узлу. Оператор "ветвь 1" завершает цикл просмотра узлов одного логического символа. После перехода через него проверяется, не достиг ли номер символа максимальной величины (пока взята константа 9000). Если этот номер не достигнут, то x13 увеличивается на 1, x14 - заменяется на 0, и откат для продолжения просмотра. Иначе - просмотр завершается с выходом в отладчик ЛОСа через оператор "трассировка(стоп 0)".

   Смена узлов одного и того же символа выполняется оператором "следузел". Для текущего узла x16 просматриваются все метки указателя, достижимого через "команды". Эти метки суть заголовки всевозможных приемов, созданных для данного узла теоремы. Текущий такой заголовок - x23.

   Если была нажата клавиша F7 и переменная x11 не равна 0, то проверяется совпадение текущего приема с приемом по ссылке x11. При различии их - сразу откат к продолжению просмотра, иначе - перед откатом x11 заменяется на 0, и для следующего приема уже начнутся обращения к процедуре "текприем".

   После перехода через "ветвь 3" доопределяется информация о приеме, которую следует передать процедуре "текприем". Затем - переменной x27 присваивается

либо результат обращения к этой процедуре, либо, для случая Ctr-F8, 0 при наличии программы приема и однобуквенный терм "См" при ее отсутствии. Наконец, при наличии в наборе x27 элемента "См" предпринимается обращение к программе редактора приемов, которой передается содержимое концевого логического терминала оглавления базы приемов, ссылающегося на текущий прием. В этом терминале ссылка на текущий прием выделена заголовком "текприем". Если выход из редактора приемов произошел по Esc, то выполняется внутренний перезапуск, иначе - просмотр продолжается.

Глава 19

Компилятор ГЕНОЛОГа

Компилятор ГЕНОЛОГа представляет собой очень большую программу, и мы не будем приводить слишком подробного ее описания. Однако, при создании приемов в новых предметных областях приходится время от времени расширять возможности ГЕНОЛОГа, и тогда оказывается необходимым пополнять и компилятор. По мере развития логической системы, такая работа возникает все реже и реже, но все же для эффективного обучения решателя нужно хорошо ориентироваться в архитектуре компилятора и иметь навыки его модернизации. Бесспорно, это является существенным недостатком ГЕНОЛОГа. Он объясняется в первую очередь тем, что ГЕНОЛОГ представляет собой копилку способов "алгоритмизации" теорем, пополняемую лишь в процессе обучения системы. Одной из основных целей самого процесса обучения является накопление и анализ информации о такого рода алгоритмизации теоретических знаний. ГЕНОЛОГ не относится к числу тех алгоритмических языков, которые можно изначально задать некоторым исчерпывающим перечнем определений. Поэтому данный недостаток представляется неизбежностью. С другой стороны, затраты усилий на пополнение компилятора в начале освоения предметной области до сих пор с лихвой окупались эффективностью последующей проработки всей этой области - интегральная трудоемкость, по сравнению хотя бы с ЛОСом, уменьшалась здесь во много раз. В будущем, чтобы упростить процесс развития компилятора ГЕНОЛОГа, представляется естественным реализовать его самого на ГЕНОЛОГе. Это хорошо согласуется с тем, что основной блок компилятора ГЕНОЛОГа - процедура "идентификатор" - организован по принципу сканирования задания на синтез программы с помощью некоторой базы процедур - своего рода приемов компиляции.

Обращение к компилятору имеет вид "новыйприем(x1 x2 x3 x4 x5)", где x1 - логический символ, за которым закреплен компилируемый прием; x2 - ссылка на узел теоремы приема; x3 - номер узла теоремы в формате десятичного числа; x4 - заголовок приема; x5 - набор дополнительных данных, обычно пустой. Если нужно скомпилировать прием так, чтобы была сохранена информация о соответствии теоремных и программных переменных, в набор x5 заносится логический символ "знач". Компилятор синтезирует программу приема, записывает ее в блок программ и создает ссылку на программу из терминала "прием" узла приема. В начале компиляции старая версия программы приема удаляется, и если компиляция обрывается либо не удается, то эта версия оказывается утеряна.

19.1 Общая архитектура компилятора

При ознакомлении с архитектурой компилятора рекомендуется войти в подраздел "Компилятор ГЕНОЛОГа"оглавления программ и идентифицировать его пункты с упоминаемыми ниже процедурами.

19.1.1 Входной блок компилятора

Собственно программа оператора "новыйприем(...)" крайне невелика и выполняет лишь предварительные действия. Прежде всего, она находит старую версию программы компилируемого приема (если таковая была) и удаляет ее. Заметим, что в "аварийных" случаях - если в узле приема имеется ссылка на то, что его программа была создана, но найти ее не удается (например, если она была удалена вручную), - инициируется прерывание со входом в отладчик ЛОСа. Это прерывание можно игнорировать, нажав для продолжения компиляции клавишу "0" и затем - Enter.

Далее, программа "новыйприем" выполняет следующие действия:

а) Преобразует теорему приема, вставляя в нее указатели "фикс(...)" для блокировки нормализации тех ее подтермов, которые в описании приема выделены указателями "копия(...)";

б) Преобразует указатели нормализации "быстрпреобр(фикс(...))", заменяя в них указатели вхождений подтермов в теорему "фикс(...)" на сами эти подтермы.

Затем происходит обращение к справочнику "новыйприем" на логическом символе, с которого начинается заголовок приема; этот справочник и выполняет все оставшиеся действия: создание программы приема, запись ее в блок программ, и регистрацию ссылки на программу из узла приема. Таким образом, фактически компилятор оказывается разбит на множество независимых процедур, соответствующих различным типам заголовков приемов. При появлении нового типа заголовков расширение компилятора осуществляется путем программирования соответствующей процедуры справочника "новыйприем". Разумеется, различные процедуры этого справочника обращаются к многочисленным блокам "общего пользования", которые будут подробнее описаны далее.

19.1.2 Компиляция приемов, применяемых при сканировании задачи

Для всех приемов, применяемых при сканировании задач, справочник "новыйприем" немедленно осуществляет обращение к процедуре "продукция(...)", которая и выполняет оставшиеся действия. Эта процедура ниже будет рассмотрена специально; сейчас укажем лишь общую последовательность ее действий:

1. Предпринимается необходимая для компиляции модификация теоремы приема. Здесь выделяются следующие преобразования:

   а) Для приемов геометрического цикла введен специальный оператор "развязка", осуществляющий обобщение теоремы, связанное с возможностями альтернативного обозначения элементов чертежа (прямых, углов и т.п.) через точки чертежа. Так, если в теореме упоминается прямая AB, причем точки A и B встречаются также вне обозначения данной прямой, то для обозначения прямой вводятся новые точки C и D, вместе с указаниями на принадлежность точек A,B прямой CD, и т.п.

б) Для тождеств вида $f(A_1 \dots A_n) = f(B_1 \dots B_m)$ с коммутативно - ассоциативной операцией f вводится дополнительная переменная x (она называется в описаниях компилятора "модификатором"): $f(A_1 \dots A_n x) = f(B_1 \dots B_m x)$, что позволяет идентифицировать все не участвующие в преобразовании операнды операции f с x.

в) Для приемов усмотрения истинности либо ложности предпринимается преобразование теоремы "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)" к виду "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то эквивалентно(A_0 истина))". Если A_0 имело вид "не(B)", то консеквент последней импликации будет иметь вид "эквивалентно(B ложь)".

2. В теореме выбирается точка привязки приема - вхождение некоторого логического символа, с обнаружения которого при сканировании задачи будет инициализироваться попытка применения приема. Для этого служит вспомогательная процедура "точкапривязки". Она стремится выбрать для привязки такой логический символ, который является наиболее "редким" - для уменьшения числа неудачных попыток применения приема. Разумеется, выбор точки привязки должен учитывать также возможность фактического отсутствия в задаче явного вхождения того или иного символа (что определяется указателями идентификации из описания приема) - такие символы не выбираются в качестве символов привязки.

3. Формируется исходная заготовка "программного блока" - накопителя создаваемой программы и информационных элементов, связываемых в процессе компиляции с этой программой. Эти элементы играют роль списка утверждений, описывающего текущее состояние процесса компиляции. Так как они имеют дело с техническими структурами данных, то для их представления выбран не логический, а технический формат - набор, начинающийся с логического символа, указывающего тип элемента. Программный блок представляет собой четверку (A_1 A_2 A_3 A_4). Здесь A_1 - первая еще не использованная в создаваемой программе программная переменная; A_2 - набор информационных элементов; A_3 - символьное число фрагментов создаваемой программы, введенных на текущий момент; A_4 - набор этих фрагментов. Ссылки между фрагментами имеют вид "ветвь n", "иначе n", где n - номер фрагмента в наборе A_4. Нумерация здесь начинается с единицы.

   Ввиду большого количества типов информационных элементов программного блока, справочная информация о них вынесена в специальное оглавление. Вход в это оглавление осуществляется нажатием клавиши "Ctr-п" из редактора программ ЛОСа.

4. Собственно процесс создания программы приема складывается из трех основных этапов - формирования программы, идентифицирующей в задаче определяемую теоремой ситуацию; формирования вставок в эту программу операторов, проверяющих истинность фильтров приема, и добавления в конце программы операторов, выполняющих преобразования задачи.

   Формирование идентифицирующей программы осуществляется специальным блоком компилятора - оператором "идентификатор". Этот блок работает со списком текущих установок на идентификацию - своего рода текущей задачей на построение идентифицирующей части программы. Он представляет собой

базу приемов, компоненты которой соответствуют различным типам установок на идентификацию. Имеется шкала уровней срабатывания этих приемов; предпринимаются циклы последовательного просмотра (сканирования) установок с уровнями срабатывания 1,2,... - до тех пор, пока не сработает какой-либо прием, дописывающий к программе дополнительные операторы и модифицирующий информационные элементы. После его срабатывания сканирование установок на идентификацию (их список может быть изменен приемом) возобновляется начиная с уровня 1. Процедура "продукция" перед обращением к оператору "идентификатор" создает начальный набор установок на идентификацию и пополняет исходную заготовку программного блока информационными элементами, необходимыми для компиляции.

Компиляция фильтров приема и вставка их в заготовку программы приема выполняется процедурой "блокпроверок". Она последовательно обрабатывает фильтры из описания приема, причем каждый фильтр компилируется рекурсивным образом, с помощью справочника "блокпроверок". Сначала рассматриваются простейшие фильтры, которые размещаются в начале программы приема - например, фильтр "уровень(...)", определяющий уровни срабатывания приема. Затем компилируются остальные фильтры. Операторы программы, реализующие проверку фильтра, размещаются возможно ближе к началу программы, если в описании приема не оговорены специальные принципы размещения данного фильтра. Фильтр размещается в первой же точке, где оказываются идентифицированы все его переменные. Поэтому порядок размещения фильтров в описании приема, как правило, не влияет на размещение операторов этих фильтров в программе.

Формирование операторов, выполняющих преобразования приема - замену подтерма, занесение либо удаление условий и посылок, - осуществляется процедурой "преобразователь". Она создает также операторы для разнообразных дополнительных действий приема, сопровождающих основное действие - ввода и удаления комментариев, изменения весов посылок и условий, и т.п. После создания цепочки операторов, формирующей новые термы, используемые при преобразовании, она размещает в конце программы обращение к одному из стандартных операторов, используемых для выполнения основного действия приема.

5. После обращения к процедурам "идентификатор", "блокпроверок" и "преобразователь" возникает предварительная версия программы приема. Эта версия пополняется операторами, которые нецелесообразно было вводить на предшествующих этапах компиляции (оператор "вставкафрагментов") и проходит оптимизацию (оператор "редакцияпрограммы", обращение к которому происходит из процедуры "вставкафрагментов"). Затем она регистрируется в блоке программ (оператор "записьприема").

19.1.3 Компиляция приемов пакетных операторов

Приемы пакетных операторов создаются процедурами справочника "новыйприем", относящимися к следующим логическим символам:

а) "замена" - приемы нормализаторов;
б) "спуск" - приемы проверочных операторов;
в) "значение" - приемы синтезатора;

г) "внутрвывод" - приемы анализаторов;
д) "контекст" - приемы оператора структурного уровня;
е) "продукция" - приемы пакета продукций;

Так как пакетные операторы имеют входные параметры, то выбор точки привязки для них не требуется - идентификация начинается с корневых вхождений входных термов. Компиляция пакетов типов а) - г) начинается с обращения к оператору "развязка", который выполняет необходимое для компиляции обобщение теоремы приема - так же, как в случае приема сканирования задачи. До формирования программного блока и обращения к процедуре "идентификатор" предпринимается вставка операторов, проверяющих наличие требуемых заголовков входных термов. После формирования идентифицирующей части программы выполняются компиляция и вставка фильтров, а также присоединение преобразующей либо выдающей ответ части. В отличие от приемов сканирования задачи, где эти действия были вынесены в специальные процедуры "блокпроверок" и "преобразователь", для пакетного оператора они выполняются непосредственно программой справочника "новыйприем".

19.1.4 Компиляция приемов вывода теорем

Программы приемов логического вывода теорем размещены в процедурах справочника "теорема". Они создаются справочником "новыйприем" на логическом символе "теорема".

19.1.5 Компиляция приемов справочников

Все программы справочников синтезируются процедурами справочника "новыйприем" на логических символах - заголовках (либо заголовках заголовков) приемов этих справочников. Как правило, такие процедуры крайне невелики; в редких случаях здесь вводится программный блок и предпринимается обращение к процедуре "идентификатор".

19.2 Схема компиляции приема сканирования задачи

Для компиляции приемов сканирования задачи служит процедура "продукция(x1 x2 x3 x4 x5 x6 x7 x8)", входные данные которой копируют входные данные справочника "новыйприем" : x1 - логический символ; x2 - ссылка на узел теоремы приема; x3 - номер этого узла; x4 - заголовок приема; x5 - дополнительная информация; x6 - описание приема; x7 - теорема приема; x8 - ссылка на узел приема. Под ссылками на узлы здесь понимаются стандартные ссылки на указатели - списки 8-го информационного блока. Будем прослеживать действия компилятора по программе "продукция", выйти на отправную точку которой можно, например, через пункт ("Компилятор ГЕНОЛОГа", "Приемы сканирования задачи", "Начало компиляции") оглавления программ.

19.2.1 Предварительное преобразование теоремы приема

Первые этапы работы процедуры "продукция" связаны с предварительным преобразованием теоремы. Некоторые из этих действий используются достаточно редко,

и при первом ознакомлении с компилятором их описания можно опустить.

Если описание приема не имеет элемента "развязка", блокирующего необходимое для компиляции геометрического приема преобразование теоремы, то проверяется, встречаются ли геометрические термины ("прямая", "угол", "окружность" и т.п.) в теореме или в термах описания приема. При обнаружении их происходит обращение к процедуре "развязка".

Процедура "Развязка"

Процедура "развязка" выполняет обобщение теоремы геометрического приема, устраняющее искусственную привязку к случайно выбранным обозначениям объектов при наличии многих равноправных их обозначений. Такая ситуация возникает, например, при обозначении углов, прямых, плоскостей и т.п. через некоторые опорные точки, выбираемые неоднозначно. Так, в условии задачи может содержаться требование о параллельности прямых AB и CD, а также рассматриваться условие на расстояние точки A до некоторой другой точки E. Вместе с тем, в задаче прямая AB может оказаться обозначенной без использования точки A, хотя некоторые посылки будут указывать на принадлежность точки A этой прямой. Чтобы идентифицирующая процедура могла усмотреть условия теоремы, целесообразно переформулировать ее так, чтобы прямая AB в теореме обозначалась через какие-нибудь совсем другие точки P, Q (для них выбираются новые переменные), добавив посылки, указывающие, что A и B лежат на прямой PQ. Такого рода преобразования и выполняются оператором "развязка".

Чтобы выйти в начальную точку программы "развязка", используем пункт ("Компилятор ГЕНОЛОГа", "Процедура РАЗВЯЗКА") оглавления программ. Прежде всего, осуществляется просмотр всех не корневых и не расположенных непосредственно под заголовками "контекст" либо "число" указателей идентификации вида "усм(...)" в описание приема. Каждый такой указатель A при компиляции должен рассматриваться как идентифицирующий терм; чтобы это было возможно, он заменяется на "контекст(A)". Затем - переход через "ветвь 2".

Здесь определяется объединенный список x7 переменных, встречающихся в теореме приема и в термах его описания. Для удобства, далее переменной x1 вместо теоремы $\forall_{x_1 \ldots x_n}(A_1 \,\&\, \ldots \,\&\, A_m \to A_0)$ присваивается терм вида "набор($A_0 A_1 \ldots A_m$)". Это позволяет, в частности, исключить из рассмотрения вхождения переменных связывающей приставки. Будем далее понимать под теоремой приема указанное значение переменной x1. Переменной x9 присваивается список фильтров приема; переменной x10 - набор указателей и нормализаторов приема. Оператор "повторение" организует цикл просмотра вхождений в теорему и в элементы описания приема для проведения развязки. В этом цикле значением переменной x12 служит текущее вхождение либо в теорему (тогда переменной x11 присваивается 0), либо в элемент (фильтр или указатель) x11 описания приема. Рассматриваются только те вхождения в фильтры и указатели, которые расположены внутри терма "усм(...)". Анализируются лишь вхождения символов "прямая", "плоскость", "угол", "окружность", "круг", так как именно они допускают существенную неоднозначность при выборе обозначения геометрического объекта. Из рассмотрения исключаются подтермы, выделенные нормализаторами (кроме особых случаев), а также подтермы, содержащие символы новых переменных, вводимых при срабабывании приема. Фактически это означает отбрасывание тех подтермов теоремы и описания приема, которые не идентифицируются с термами задачи, а создаются для выполнения преобразований.

Начиная с контрольной точки "прием(2)", происходит рассмотрение различных случаев развязки. Предварительно вводятся накопители x13 (заменяющий терм) и x14 (набор сопровождающих утверждений, добавляемых при развязке) Выделяются следующие случаи:

1. x12 - вхождение терма T вида "прямая(AB)" либо "плоскость(ABC)". Если существует вхождение в теорему приема либо в терм описания приема (в последнем случае - под заголовком "усм") одной из переменных A, B, C, которое не расположено внутри экземпляра терма T, либо внутри терма "прямая(...)" или "плоскость(...)", обрабатываемого нормализатором общей стандартизации, то предпринимается решение о развязке. Выбираются новые переменные D, E, F и формируется заменяющий терм R вида "прямая(DE)" либо, соответственно, "плоскость(DEF)". Создается также список утверждений для регистрации в контексте замены: "принадлежит(A R)"; "принадлежит(B R)"; "принадлежит(C R)", а также (если различие точек A, B, C не усматривается из контекста) вспомогательные утверждения "несовп(A B)", "несовп(A C)" и "несовп(B C)". Последние используются при компиляции для создания ускоряющих фильтров, отсекающих случаи совпадения обозначений соответствующих точек.

2. x12 - вхождение терма T вида "угол(ABC)". Если существует вхождение в теорему приема либо в терм описания приема (в последнем случае - расположенное под "усм(...)") одной из переменных A, C, которое не расположено внутри экземпляра терма T либо внутри обрабатываемого нормализатором терма "прямая(...)", то принимается решение о развязке. Выбираются новые переменные D, E, F, G, P, Q, и формируется заменяющий терм "угол(DBE)". Для регистрации в контексте замены создается следующий список утверждений: "принадлежит(B прямая(FG))"; "принадлежит(A прямая(FG))"; "принадлежит(D прямая(FG))"; "принадлежит(B прямая(PQ))"; "принадлежит(C прямая(PQ))"; "принадлежит(E прямая(PQ))"; "точкалуча(B D A)"; "точкалуча(B E C)". Если в описание приема был включен указатель "нормугол(угол(ABC))", означающий, что при идентификации несущественна ориентация лучей угла, то последние два утверждения в контекст замены не включаются.

3. x12 - вхождение терма T вида "окружность(AB)" либо "круг(AB)". Если существует вхождение в теорему приема либо расположенное под "усм(...)" вхождение в терм описания приема переменной B, которое не расположено внутри экземпляра терма T либо внутри обрабатываемого нормализатором терма, то принимается решение о развязке. Выбирается новая переменная C, и формируется заменяющий терм R вида "окружность(AC)" либо, соответственно, "круг(AC)". Для регистрации в контексте замены вводится утверждение "принадлежит(B R)".

После определения заменяющего терма x13 и контекста замены x14 - откат к переходу через "ветвь 3". Здесь проверяется, расположено ли текущее вхождение x12 внутри подтермов "контекст(...)" или "число(...)" терма описания приема. Если расположено, то переход через "иначе 1". В противном случае - осуществляется замена на x13 всех вхождений в теорему, в фильтры и указатели подтермов, равных подтерму по вхождению x12.

Если по вхождению x12 располагался терм "угол(ABC)", то одновременно с переобозначением самого угла переобозначаются прямые, соответствующие его сторонам. Это переобозначение выполняется в теореме приема, фильтрах и указателях. После переобозначения проверяется, входят ли переменные A, C в теорему и описание приема; если не входят, то из x14 исключаются все утверждения с соответствующей переменной.

Наконец, список фильтров x9 пополняется утверждениями "усм(A)" для всех утверждений A из контекста замены, и откат к повторному просмотру вхождений x12.

В случае, когда x12 было расположено внутри терма x15 вида "контекст(...)" либо "число(...)", выполнялся переход через "иначе 1" (см. выше). Здесь определяется набор x16 операндов терма x15, и предпринимается замена в них вхождений подтермов T, равных x12, на x13. Это не относится к фрагментам "терм(T)", выделяющим создаваемые новые копии терма T. В случае углов, как и выше, предпринимается сопутствующее переобозначение прямых - сторон угла. После преобразования термов набора x16 к ним присоединяются утверждения "усм(A)" для всевозможных утверждений A из набора x14. Наконец, выполняется замена старого варианта содержащего x15 фильтра либо указателя на новый, и откат к повторному просмотру вхождений x12.

Если просмотр вхождений x12 не привел к новым переобозначениям, то переход через "иначе 2". Здесь переменной x12 присваивается консеквент преобразованной теоремы; x13 - список ее антецедентов. Находятся все фильтры вида "усм(A)", и утверждения A присоединяются к концу списка антецедентов. Одновременно к списку указателей добавляется элемент "усм(...)", ссылающийся на новые антецеденты. Переменной x16 присваивается список оставшихся фильтров. Далее - переход через "ветвь 1".

Здесь предпринимается удаление избыточных элементов, введенных при развязке. Сначала осуществляется просмотр всех антецедентов x19, имеющих вид "актив(A)" и выделенных указателем "усм(...)", у которых выражение A встречается в другом антецеденте, также выделенном указателем "усм(...)". Такие антецеденты x19 исключаются; соответственно корректируется нумерация антецедентов в указателях приема. Затем - переход через "иначе 2", где выполняется просмотр фильтров приема, содержащих идентифицирующие термы "усм(... актив(A) ...)", расположенные внутри подтермов "контекст(...)" либо "число(...)". Если терм A встречается внутри антецедента, выделенного указателем "усм(...)", либо рассматриваемое вхождение идентифицирующего терма "усм(...)" расположено внутри подтерма "контекст(...)" или "число(...)", имеющего другой содержащий A идентифицирующий терм "усм(...)", то происходит исключение элемента "актив(A)" из первого идентифицирующего терма. Далее - переход через "иначе 1". Здесь устраняются избыточные обозначения для одной и той же прямой. Если имеются два антецедента x19, x21 вида "принадлежит(A прямая(EF))" и "принадлежит(B прямая(EF))", выделенные указателями "усм(...)"; имеется антецедент "несовп(...)", указывающий на различие точек A, B; имеются два других антецедента "принадлежит(A прямая(GH))", "принадлежит(A прямая(GH))", причем выражение "прямая(GH)" лексикографически предшествует выражению "прямая(EF)", то выполняется замена в приеме всех вхождений второго обозначения на первое. Антецеденты x19, x21 удаляются, и корректируется нумерация антецедентов в оставшихся указателях.

По завершении перечисленных действий - откат к оператору "ветвь 1" перед меткой "прием(1 1)". Здесь переменной x17 присваивается результат сборки кванторной

импликации для теоремы приема (она создается по списку антецедентов x13 и ранее выделенному консеквенту x12); создается скорректированное описание приема x18, и выдается результат.

Прочие случаи модификации теоремы приема

Кроме сравнительно сложной процедуры модификации теоремы приема, выполняемой оператором "развязка", имеется ряд более простых случаев модификации теоремы. Начала их программ легко найти через подраздел "Предварительная модификация теоремы и описания приема" раздела "Приемы сканирования задачи" оглавления программ:

1. В некоторых случаях бывает нужно явно указатеь в теореме приема выражение "величина(...)", обозначающее десятичную константу перечислением ее цифр. Однако, при прорисовке таких выражений формульный и тектовый редакторы выдают лишь цифры, игнорируя символ "величина". Чтобы создать в теореме приема запись "величина(...)" с переменной вместо конкретного набора цифр, введен вспомогательный одноместный символ "Величина", который прорисовывается явным образом. При компиляции терм "Величина(a)" заменяется на "величина(a)". Использовать в теореме, фильтрах и указателях приема переменную a вне контекста "Величина(a)" не рекомендуется, так как самостоятельным образом она не идентифицируется.

2. При описании указателей, уточняющих тип основного преобразования, был рассмотрен указатель "однозначно". Он относится к теоремам вида $A \to P(t_1 \ldots t_n)$, где отношение P однозначно определяет значение t_i по значениям $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$, и задает преобразование теоремы к виду

$$A \to (P(t_1 \ldots t_{i-1}\ x\ t_{i+1} \ldots t_n) \leftrightarrow x = t_i).$$

   После определения нового варианта теоремы x18 предпринимается коррекция указателей вхождений "фикс(...)" в старую версию теоремы, расположенных в термах описания приема.

3. Если заголовок приема - "связка" либо "второйтерм", а консеквент теоремы не является равенством либо эквивалентностью, то предпринимается его преобразование к виду "равно(A истина)" либо "равно(A ложь)". Если прием имеет указатель "эквивалентно", то такое же преобразование выполняется и для равенства. Как и в предыдущем случае, после изменения теоремы корректируются указатели вхождений "фикс(...)".

4. Если теорема представляет собой тождество с коммутативно-ассоциативным заголовком в заменяемой части $f(t_1 \ldots t_n)$, то обычно вводится дополнительная переменная x этой части, с которой будут идентифицироваться все остаточные операнды операции f. Такая же переменная вводится и в заменяющей части. Добавленная переменная x называется модификатором; ее ввод выполняется процедурой "модификатор". Чтобы модификатор не вводился, необходимо либо наличие указателя "модификатор", либо наличие фильтра "полный" (в последнем случае - при отсутствии указателя "набор(первыйтерм)"), либо наличие указателя "набор(второйтерм)".

19.2.2 Выбор точки привязки приема

Чтобы определить ту точку теоремы приема либо его указателей, с которой будет начинаться идентификация ситуации, требующей его срабатывания, используется процедура "точкапривязки(x1 x2 x3 x4 x5)". Ей передаются в качестве входных данных: x1 - теорема приема; x2 - заголовок приема; x3 - описание приема. Процедура присваивает переменной x4 пару (вхождение корня идентифицируемого подтерма, т.е. заменяемой части тождества либо эквивалентности, антецедента, консеквента, указателя и т.п. - вхождение точки привязки в идентифицируемый подтерм). Переменной x5 присваивается логический символ, выбранный для привязки. Попытка применения приема будет начинаться с его усмотрения. Основная задача процедуры выбора точки привязки в теореме приема - найти вхождение в непосредственно идентифицируемую часть теоремы такого логического символа, который возможно реже встречается в задачах. Это нужно для уменьшения трудоемкости попыток применения данного приема. Если указатели идентификации предоставляют возможность размещения по заданному вхождению альтернативных символов, либо вообще отсутствия символа ввиду вырождения операции, то такое вхождение при выборе точки привязки не рассматривается.

В оглавлении программ момент обращения к процедуре "точкапривязки" из процедуры "продукция" отдельным пунктом не выделен, так как он непосредственно предшествует пункту "x11 - прогр. блок для ...". Вместо этого введен подраздел "Выбор в теореме точки привязки", пункты которого соответствуют этапам работы оператора "точкапривязки". К рассмотрению программы этого оператора мы и переходим.

Прежде всего, переменной x6 присваивается вхождение подлежащего идентификации терма, если такой терм усматривается из описания приема, либо вхождение первого символа теоремы приема. Если имеются явные указания на то, что привязка выполняется по корневому вхождению, то сразу выдается результат - вхождение x6. В противном случае переменной x7 присваивается заполненный нулями список, длина которого равна числу вхождений логических символов в теорему. В зависимости от описания приема, в теореме выделяется группа вхождений корней идентифицируемых подтермов (для приема замены - уже определенное выше x6; для приема вывода - вхождения непосредственно идентифицируемых антецедентов, и т.п.). Эти вхождения регистрируются в начале списка x7. При последующем анализе вхождений в список x7 будут заноситься (на свободные позиции, заполненные нулями) расположенные внутри идентифицируемых подтермов другие кандидаты на выбор точки привязки. Одновременно формируется список x8 - длина его равна длине списка x7, причем на позициях, соответствующих используемым элементам списка x7, т.е. вхождениям в теорему, будут размещаться пары (логический символ по данному вхождению - глубина вхождения этого символа в идентифицируемый подтерм). Фактически эти позиции инициализируются, по мере заполнения списка x7, парами, у которых первый элемент равен 0, и лишь впоследствии этот элемент заменяется на логический символ. Значением переменной x9 служит текущее анализируемое вхождение в список x7; значением переменной x10 - первое неиспользуемое вхождение в x7.

После оператора, заполняющего начальную часть списка x8, размещен оператор "повторение". К нему будут выполняться откаты при переходе к следующей позиции списка x7. Переменной x11 присваивается логический символ, расположенный на текущем вхождении в теорему (т.е. вхождению по x9). Если x11 - символ "значение",

являющийся вхождением функциональной переменной, то ее заголовок не определен - такая переменная может идентифицироваться с произвольным термом. Поэтому рассмотрение текущего вхождения в теорему отменяется путем занесения нуля на соответствующую позицию списка x8. Затем - откат к "ветвь 1". Иначе, после перехода через "ветвь 2", находится список x12 вхождений всех операндов текущего вхождения в теорему. Первый элемент текущей пары списка x8 заменяется на x11. Если указатель "альтернатива" предоставляет возможность какому-либо из подтермов, вхождения которых перечислены в списке x12, иметь неоднозначно определенный заголовок, то его позиция в списке x12 заменяется на 0. Далее все ненулевые элементы списка x12 регистрируются в конце списка x9 и инициализируются соответствующие им позиции списка x8. При этом корректируется указатель первой свободной позиции x10.

После рассмотрения операндов текущего вхождения в теорему - откат к переходу через "ветвь 1". Здесь прежде всего проверяется, равны ли x9 и x10. Если не равны, то x9 сдвигается на единицу вправо, и откат к "повторению" для анализа очередного текущего вхождения. Если x9 равно x10, то цикл заполнения списков x7 и x8 завершен. Теперь в списке x7 представлены все предварительные кандидаты на точку привязки, а в списке x8 - соответствующие им пары. Далее, начиная с контрольной точки "прием(7)", начинается цикл просмотра списков x7, x8 и отбрасывания тех вхождений, которые заведомо непригодны в качестве точки привязки. Для исключения их на соответствующую позицию списка x8 заносится 0.

Текущие вхождения в списки x7, x8 суть значения переменных x14, x15. Просматриваются только вхождения, на которых расположены ненулевые значения. Текущее вхождение в теорему при просмотре присваивается переменной x16. Исключение вхождения происходит в следующих случаях:

1. x16 - вхождение операции $f(A_1 \dots A_n)$, у которой некоторый операнд A_i - переменная, выделенная указателем "единица", либо некоторый операнд - коммутативная операция $g(x1 \dots x_k)$, все операнды $x_1, \dots, x_k$ которой могут оказаться идентифицированы с ее единицей, причем эта единица одновременно является единицей для внешней операции f. Если $n = 2$, то символ f в указанных ситуациях может отсутствовать, и выбирать его вхождение в качестве точки привязки нельзя;

2. x16 - вхождение операции $f(x)$, у которой переменная x выделена указателем "отрицание(...)" - такая операция может отсутствовать, если само x идентифицируется как $f(t)$;

3. x16 - вхождение операции, выделенное указателями "нормзнака(...)" либо "общаястепень(...)";

4. x16 - вхождение операции с единственной переменной, для которого справочник "вид" предусматривает специальную идентификацию. Фактически этот справочник используется только для идентификации степеней с константными натуральными показателями, которые усматриваются специальной процедурой выделения степени. Логический символ "степень" здесь может вообще отсутствовать - например, при усмотрении полного квадрата в числовой константе;

5. x16 - вхождение константного терма, для заголовка которого справочник "пересечениесписков" предусматривает специальную идентификацию. Например,

для произведений, которые могут усматриваться в выражениях, не имеющих заголовка "умножение" - в константных степенях и числовых константах;

6. x16 - вхождение операции, для которого возможна специальная идентификация с изменением заголовка либо вырождением идентифицируемого терма - для указателей "частнпроизв", "новаргумент", "извлечениеварианта", "подстановка", "знаксуммы", "заменатермов" и др.

По окончании просмотра списков x7, x8 - откат к переходу через "ветвь 2". Здесь составляется список x14 всех логических символов, оставшихся для выбора точки привязки после проведенной расчистки. Предпринимается попытка исключить из него все общелогические символы - "и", "или", "не", "длялюбого", "существует", "отображение", "класс", и т.п. Если после этого список становится пустым, то данная попытка отменяется.

Наконец, начиная с контрольной точки "прием(9)" происходит окончательный выбор точки привязки. Последовательно просматриваются символы набора x14, и среди них отбирается тот, который реже всего встречается в теоремах приемов. Переменная x15 играет роль накопителя, в котором окажется отобранный в результате просмотра символ, а переменная x16 - роль оценки частоты его появления. Такая оценка берется из логического терминала "мощность" узла статьи логического символа в базе приемов. Как уже говорилось выше, в этих терминалах автоматически создается и корректируется число вхождений символа в теоремы приемов. Если в описании приема имелся указатель "точкапривязки(A)", явно определяющий логический символ привязки A, то оценка этого символа полагается равной 1. Заметим, что при порче содержимого терминалов "мощность", когда в них попадают отрицательные величины, компилятор будет игнорировать указатель "точкапривязки" и отбирать символ с наибольшим по модулю отрицательным содержимым терминала "мощность". В этих случаях можно либо корректировать испорченные терминалы вручную, либо запускать цикл автоматического переформирования их нажатием клавиши Ctr-ь из оглавления базы приемов.

Если среди кандидатов на точку привязки имеется несколько вхождений выбранного символа A, то выбирается вхождение с наибольшей глубиной. Такой выбор дает наибольшую длину цепочки от точки привязки к корню идентифицируемого подтерма. Идентификация символов этой цепочки происходит без какого-либо ветвления и может дать хорошее отсечение заведомо неподходящих ситуаций уже на первых шагах.

19.2.3 Создание программного блока и системы установок на идентификацию

При создании фрагментов программы приема используется специальная структура данных, называемая программным блоком. Она сохраняет возникшие на текущий момент компиляции наборы операторов этих фрагментов, а также содержит различную информацию, характеризующую текущий момент компиляции. В основном, это информация о связи программных и теоремных переменных, а также специально подготовленная для ускоренного использования информация, извлеченная из указателей приема. Например, вместо косвенных ссылок на вхождения в теорему через указатели вхождений "фикс(...)" здесь могут приводиться сами вхождения в теорему.

Программный блок представляет собой четверку (x A n P), у которой x - первая еще не использованная в текущем создаваемом фрагменте программы программная переменная; A - набор информационных элементов, характеризующих текущий момент компиляции; n - символьное число уже введенных фрагментов программы (включая текущий фрагмент); P - список этих фрагментов ($P_1 \ldots P_n$). Текущим заполняемым фрагментом является фрагмент P_n. Так как в программах ЛОСа предпочтительно определять значения программных переменных последовательно, в порядке их нумерации, то для присвоения очередных значений компилятор выбирает одну или несколько идущих подряд переменных, начиная с x, сразу же увеличивая на соответствующую величину номер первой не использованной переменной. Для ссылок между фрагментами набора P используются номера этих фрагментов (нумерация, как показано выше, начинается с 1). Таким образом, могут возникать операторы "ветвь(k)" либо "иначе(k)", причем номер k, как и число n, представлен в символьном формате.

Кроме программного блока, в процессе создания идентифицирующей части программы приема используется набор так называемых установок на идентификацию. Они представляют собой задания на создание групп операторов, выполняющих какую-то конкретную часть общей идентификации ситуации, определяемой описанием приема. Обработка этих установок на идентификацию происходит рекурсивным образом, и после рассмотрения какой-либо одной из них вместо нее могут быть введены несколько новых. По исчерпании списка установок на идентификацию должна возникать такая программа, что прохождение через нее вплоть до последнего оператора гарантирует наличие ситуации, заданной в описании приема, кроме истинности фильтров приема. Вставка операторов, проверяющих истинность фильтров, осуществляется на следующем этапе компиляции.

Число различных типов информационных элементов, используемых в программном блоке, весьма велико. Ниже мы посвятим описанию этих типов специальный раздел, а здесь приведем лишь совсем краткий список наиболее часто встречающихся элементов. Как уже говорилось выше, справочное оглавление типов информационных элементов программного блока достижимо из редактора программ ЛОСа по нажатию клавиши Ctr-п.

Информационные элементы программного блока

Сначала перечислим элементы, несущие общую информацию о компилируемом приеме. Они бывают нужны, если вспомогательная процедура, которой передается программный блок, не имеет среди прочих своих входных переменных полной информации о приеме.

1. (прием s N A) - ссылка на компилируемый прием: s - логический символ; N - номер узла теоремы приема в статье символа s; A - заголовок приема;

2. (логсимвол A) - A есть логический символ, в программу которого будет включен результат компиляции;

3. (точкапривязки A) - A есть вхождение в теорему точки привязки;

4. (корень x) - x есть программная переменная, значением которой служит вхождение корня идентифицируемого терма;

5. (модификатор x) - x есть теоремная переменная, введенная при модификации теоремы приема как дополнительный операнд ассоциативно-коммутативной операции для идентификации с остатком операндов (см.выше);

6. "и" - компилируемая программа должна состоять из единственного фрагмента. Например, такой элемент вводится при создании группы операторов, которую нужно будет разместить под квантором или описателем;

7. (комментарии A) - A есть совокупность всех таких фильтров приема, которые автоматически оказываются истинными после выполнения уже составленной части программы. Элемент используется на этапе завершающей расстановки операторов, реализующих фильтры приема;

8. (уровень A) - A есть список всех значений уровней срабатывания приема;

9. (меткаперехода A) - указание на логический символ A, уже использованный в качестве метки перехода. Метки перехода - явление, отсутствующее в обычных программах ЛОСа и введенное только на этапе их компиляции. Фиктивный оператор "меткаперехода(B)" вставляется в ту точку программы, к которой потом нужно будет делать откат. Для организации отката используется оператор "переходпометке(B)". Метка перехода B выбирается произвольным образом; чтобы каждый раз выбирать новую метку, создаются элементы (меткаперехода ...), обозначающие уже использованные метки. На завершающем этапе компиляции происходит исключение операторов "меткаперехода" и "переходпометке" - для откатов используются операторы "продолжение", "обрыв" и "сброс";

10. (команда A) - A есть описание приема;

11. "контекст" - программный блок введен для реализации фильтра "контекст(...)" либо одного из аналогичных ему выражений - "число(...)", "нод(...)", "максимум(...)" и т.д.;

12. "перечисление" - при компиляции не вводятся смешанные проверочно- перечисляющие операторы. Они преобразуются в формат перечисляющего оператора, чтобы можно было точно определить параметр оператора "сброс(...)";

13. (левыйкрай A) - A есть вхождение корня идентифицируемой части теоремы приема;

14. (теорема A) - A есть теорема приема;

15. (адресузла A) - A есть ссылка на узел приема;

Перечислим еще ряд специальных типов информационных элементов программного блока, которые встретятся нам на начальном этапе компиляции.

1. (операнд A B) - A есть некоторое вхождение в теорему; B - программное выражение для того вхождения в терм задачи, которое идентифицировано с A. Заметим, что не каждое вхождение в теорему идентифицируется с вхождением в терм задачи. Например, это невозможно, если операнд какой-либо ассоциативно-коммутативной операции теоремы идентифицируется с группой операндов такой же операции в задаче.

2. (набороперандов A B) - A есть вхождение в теорему терма $f(\ldots)$ с ассоциативно-коммутативным заголовком f. B - программное выражение, значением которого служит набор еще не идентифицированных операндов той операции F, которая идентифицируется с A. Обычно этот элемент создается в тех случаях, когда какой-либо операнд терма A идентифицируется с группой операндов терма F. После того, как элемент создан, по ходу компиляции выражение B может несколько раз корректироваться - до полного исчерпания операндов терма F.

3. (подтерм A) - A есть вхождение в теорему какого-либо подлежащего идентификации подтерма, расположенного вне того идентифицируемого подтерма, к которому относится точка привязки. Например, в приеме замены может иметься антецедент, непосредственно идентифицируемый с посылкой или условием задачи, и тогда он сопровождается данным информационным элементом.

4. (антецедент A B) - A есть вхождение в теорему приема корня антецедента, идентифицируемого с посылкой либо условием задачи. B инициализируется нулем, а в процессе компиляции становится равно программному выражению, определяющему найденную для A посылку либо условие.

5. (блокпроверок A) - A есть набор вхождений антецедентов теоремы приема, для обработки которых предусмотрено обращение к проверочному оператору;

Установки на идентификацию

Установки на идентификацию используются только процедурой "идентификатор", создающей идентифицирующую часть программы приема. Однако, вводятся они различными блоками компилятора перед обращением к процедуре "идентификатор", и поэтому полный список таких установок мы приведем уже сейчас. Этот список имеется также в справочной информации символа "идентификатор". Более подробная информация о том, как обрабатываются перечисляемые установки, составит значительную часть раздела, посвященного процедуре "идентификатор".

1. (операнд v x) - v есть вхождение в теорему приема корня подлежащего идентификации подтерма; x - программная переменная, значением которой служит вхождение, идентифицированное с v. Это - один из наиболее часто встречающихся типов установок на идентификацию. Предполагается, что на момент ее рассмотрения логический символ по вхождению v уже идентифицирован. Компилятор анализирует корневые операнды вхождения v, сопоставляя им программные выражения для соответствующих термов задачи и идентифицируя заголовки этих термов. Для этих операндов, если они не являются переменными или логическими символами, вводятся рекурсивным образом новые установки (операнд ...). Если точка привязки располагалась внутри идентифицируемого терма, то при обработке данной установки на идентификацию возможно также "движение наружу" - рассмотрение внешней операции для v.

2. (корень v) - v есть вхождение антецедента теоремы, подлежащего идентификации с утверждениями, содержащимися в контексте срабатывания приема.

3. (извлекается v) - v есть вхождение антецедента, для проверки которого используется проверочный оператор;

4. (значения A) - A есть список вхождений антецедентов, обрабатываемых одним пакетным синтезатором;

5. (программа v) - v есть вхождение программно реализуемого антецедента теоремы приема;

6. (идентификатор v) - v есть вхождение первого символа антецедента вида "равно(A_1 A_2)", одна из частей которого определяется на основе значений входящих в нее ранее идентифицированных теоремных переменных, а затем происходит идентификация с ней другой части равенства. Возможен также случай, когда обе части к моменту обработки установки составлены из идентифицированных переменных - тогда происходит сравнение построенных для них термов. В обоих случаях, при построении термов могут применяться обращения к нормализаторам;

7. (усм v) - v есть вхождение антецедента, обрабатываемого идентифицирующим оператором;

8. (очевидно v) - v есть вхождение антецедента, истинность которого устанавливается при помощи оператора "очевидно";

9. (легковидеть v) - v есть вхождение корня антецедента, истинность которого устанавливается при помощи задачи на доказательство, решаемой до максимального уровня 4 и имеющей комментарий "извлекается";

10. (усматривается v) - v есть вхождение антецедента теоремы, проверка истинности которого осуществляется с помощью вспомогательной задачи на доказательство, решаемой до максимального уровня 5;

11. (доказать v) - v есть вхождение корня антецедента, истинность которого устанавливается при помощи задачи на доказательство, к которой сводится текущая задача на доказательство;

12. (следствие v) - v есть вхождение антецедента, истинность которого устанавливается при помощи задачи на доказательство, уровень обращения к которой берется равным максимальному уровню текущей задачи;

13. (транзитпереход v) - v есть вхождение антецедента, выделенного указателем "транзитпереход";

14. (типзначения v) - v есть вхождение антецедента, выделенного указателем "типзначения(...)";

15. (чертеж v) - v есть вхождение антецедента, выделенного указателем "чертеж(...)";

16. (числоценка v) - v есть вхождение антецедента, выделенного указателем "числоценка(...)";

17. (выч v) - v есть вхождение антецедента, выделенного указателем "выч(...)";

18. (рекурсия A) - A есть набор вхождений антецедентов, выделенных указателем "рекурсия(...)";

19. (контекст A) - A есть терм "идентификатор($A_1 \ldots A_n$)", перечисляющий набор идентифицирующих и проверочных термов $A_1, \ldots, A_n$, извлеченных из указателя "контекст($A_1 \ldots A_n$)", определяющего дополнительную идентификацию;

20. (новаргумент A B) - A есть программное выражение для терма, идентифицированного с теоремным термом "значение(x C)", выделенным указателем "новаргумент($x \ldots$)" либо "функаргумент($x \ldots$)". B есть вхождение данного терма в теорему приема;

21. (развертка v A_1 A_2) - v есть вхождение в теорему терма t, идентифицируемого в соответствии с указателем "развертка(...)". A_1 - программное выражение, значением которого служит либо вхождение (при $A_2 = 0$) терма T, идентифицируемого с t, либо (при $A_2 = 1$) сам терм T. Возможен также случай, когда A_1 - информационный элемент программного блока (набороперандов B_1 B_2), где B_2 - набор операндов коммутативно-ассоциативной операции, из которых комплектуется T. Тогда $A_2 = 2$.

22. (вычисл A_1 A_2) - A_1 есть набор вхождений антецедентов, представляющих собой равенства, выделенные указателем "вычисл". Эти равенства реализуются с помощью вспомогательной задачи на описание, решаемой для выражения ее неизвестных через заданные параметры.

23. (группировка v x) - v есть вхождение корня заменяемого терма в приеме, имеющем указатель "набор(второйтерм)"; x - программная переменная для вхождения, идентифицируемого с v. Установка аналогична установке (операнд ...), однако требует специальной обработки для режима группировки всех операндов рассматриваемой ассоциативно-коммутативной операции;

24. "свертка" - установка на идентификацию остаточных посылок задачи в приеме с заголовком "свертка";

25. (теквхожд v) - v есть вхождение корня заменяемой части тождества либо эквивалентности. Эта установка вводится при наличии указателя "теквхожд(...)" и нужна для компиляции первых шагов идентификации согласно данному указателю;

26. (унификация A) - A есть набор вхождений антецедентов с заголовком "равно", обрабатываемых с помощью процедуры унификации. Эта установка нужна для компиляции приемов вывода в базе теорем;

27. (тип x A) - переменная x идентифицируется с логическим символом, являющимся типом значения выражения A;

28. (вход A_1 A_2) - A_1 есть теоремный терм; A_2 - программное выражение для объекта, идентифицируемого со значением терма A_1. Установка используется при компиляции приемов пакетов продукций.

Инициализация программного блока и системы установок на идентификацию

Возвращаемся к программе "продукция" для продолжения рассмотрения действий компилятора после выбора точки привязки. Для этого используем пункт "x11 -

прогр.блок ..." оглавления программ, который переводит к контрольной точке "прием(9)".

Сразу после этой контрольной точки располагается оператор "равно(x11 ...)", присваивающий переменной x11 заготовку программного блока. На первой позиции набора x11 находится переменная "x6", которая является первой не определенной переменной при сканировании задачи. В число информационных элементов программного блока включены наборы (точкапривязки ...), (операнд ...), (логсимвол ...), (прием ...), (адресузла ...), (теорема ...), (команда ...). Здесь элемент (операнд ...) констатирует, что значением программной переменной x2 при сканировании задачи является вхождение точки привязки приема. Если точка привязки совпала с корнем идентифицируемого подтерма, то добавляется элемент (корень ...). В заготовке программного блока имеется единственный фрагмент программы, состоящий из единственного оператора "решить".

Непосредственно после контрольной точки "прием(10)"располагается группа операторов, обеспечивающая пополнение программного блока элементом (модификатор x15). Этот элемент вводится для приема тождественной либо эквивалентной замены, если заменяемый и заменяющий термы имеют одинаковый ассоциативно-коммутативный заголовок x13 и среди их операндов усматривается одна и та же переменная x15, не связанная с прочими операндами. После анализа необходимости создания данного элемента - переход через "ветвь 1".

Здесь инициализируется набор x12 установок на идентификацию. В зависимости от типа приема, в него заносится либо элемент (операнд v x2), где v - точка привязки, либо элемент (группировка v x2). В особых случаях (при наличии указателя "теквхожд(...)") набор x12 остается пустым.

Дальнейшие действия по доопределению программного блока и набора установок на идентификацию разбиваются на несколько ветвей. Основная линия компиляции будет продолжена после отката к переходу через "ветвь 1", а сначала размещаются несколько подветвей, соответствующих различным заголовкам приема.

Прежде всего, располагается ветвь, к которой относятся приемы с заголовками "выводусловия", "вывод", "заменатермов", "замечание" (только для теорем, имеющих консеквент "пусто"), "выразимо", "подборзначений", "заменаусловия", "параметризация", "существует", "связка". Здесь переменной x13 присваивается список вхождений в теорему всех ее антецедентов. Если имеются указатели "конец(...)", перечисляющие номера тех антецедентов, обработка которых отодвигается на конец компиляции, то создаются соответствующие информационные элементы (конец ...) программного блока, ссылающиеся на эти антецеденты непосредственно через их вхождения. Далее - переход через "ветвь 3".

Здесь располагается оператор "длялюбого(x14 x15 ...)", просматривающий указатели приема, определяющие способы обработки антецедентов. Для текущего указателя в набор x12 заносится одноименная установка на идентификацию, причем соответствующая антецеденту позиция набора x13 заменяется на 0. После него - аналогичным образом просматриваются сначала указатели "рекурсия(...)", "значения(...)", а затем - указатели "вычисл(...)". Создаются одноименные установки на идентификацию, и в наборе x13 обнуливаются уже не единичные позиции, а сразу группы таких позиций, определяемых указателем.

После перехода через "ветвь 1" попадаем в фрагмент, где список x12 пополняется установками на идентификацию (корень v) для всех оставшихся в наборе x13 вхождений антецедентов v. В приемах некоторых типов происходит идентификация с утверждениями из контекста срабатывания приема не только антецедентов, но и

специальных подтермов консеквента. Для этих подтермов создаются дополнительные установки на идентификацию (корень ...).

На основе занесенных в x12 установок (корень ...) происходит регистрация в программном блоке элементов (антецедент ...). В случае приемов вывода такой же элемент регистрируется для того антецедента, в котором находится точка привязки приема. Далее - переход через "ветвь 1", где создается информационный элемент программного блока (блокпроверок ...), ссылающийся на все антецеденты, обрабатываемые проверочными операторами. После следующего перехода через "ветвь 1" - попытка усмотрения того, что точка привязки приема должна являться корневым вхождением соответствующего терма задачи. Если это так, то к накапливаемой в программном блоке программе добавляется оператор "корень(x2)".

Наконец, в случае приемов с заголовками "существует", "связка" создаются информационные элементы (неизвестная ...), перечисляющие переменные, которые должны быть идентифицированы с неизвестными текущей задачи на описание. После этого - откат к оператору "ветвь 1" для продолжения основной линии компиляции.

Следующая ветвь соответствует приемам с заголовками "первыйтерм", "второйтерм", "замечание" (для теорем, консеквент которых отличен от символа "пусто"). Как и в предыдущей ветви, переменной x13 присваивается набор вхождений в теорему приема ее антецедентов. Если прием замены имеет указатель "теквхожд(...)", то точка привязки его располагается в антецеденте, выделенном таким указателем. Тогда вхождение заменяемого терма находится независимо от точки привязки, и для идентификации его в накопитель x12 заносится установка (теквхожд v). Затем - откат к переходу через "ветвь 3".

Здесь выполняются действия, аналогичные рассмотрению антецедентов в предыдущей ветви: создаются информационные элементы (конец ...) для антецедентов, рассмотрение которых отодвинуто на конец идентификации; создаются установки на идентификацию, соответствующие различным указателям обработки антецедентов. При вводе каждой новой установки соответствующие ее антецедентам разряды набора x13 обнуливаются. Для оставшихся в x13 антецедентов вводятся установки (корень ...). Далее создаются информационные элементы (антецедент ...), (блокпроверок ...), а также аналогичные последним элементы (мнимаячасть A), выделяющие список A вхождений тех антецедентов, которые обрабатываются усиленными проверочными операторами. Заметим, что здесь на некотором этапе развития системы произошло переименование на "мнимаячасть" логического символа, который ранее использовался в качестве заголовка данного информационного элемента (более этот символ ни для чего не был использован). Поэтому и возникло такое несоответствие названия функциональной нагрузке элемента.

Наконец, последняя из трех ветвей - она целиком относится к приемам с заголовком "свертка". Напомним, что эти приемы используются в задачах на доказательство. Консеквент их теоремы представляет собой кванторную импликацию K, консеквент которой идентифицируется с условием задачи, а антецеденты - с посылками задачи. В решателе заголовок "свертка" имеют приемы доказательства по индукции.

В список x12 установок на идентификацию заносится символ "свертка" и всевозможные пары (корень v), где v - вхождение антецедента либо консеквента указанной выше кванторной импликации K, не содержащих точку привязки. Переменной x14 присваивается список вхождений антецедентов в теорему приема. Далее выполняются действия, аналогичные имевшим место в первых двух ветвях: вводятся элемен-

ты (конец ...)и создаются установки на идентификацию антецедентов списка x14, соответствующие указателям приема. Для всех оставшихся антецедентов вводятся установки (корень ...); создаются информационные элементы (антецедент ...), (блокпроверок ...). После всех этих действий - откат к переходу через "ветвь 1", где продолжается рассмотрение приемов с заголовком "свертка". Здесь корректируется список фильтров приема: если точка привязки была выбрана в антецеденте импликации K, то добавляется фильтр "посылка"; если она была выбрана в консеквенте - добавляется "условие". Кроме того, для всех переменных x связывающей приставки импликации K, не выделенных указателем "кортежпеременных", добавляется фильтр "переменная(x)". Наконец, если точка привязки соответствовала корневому вхождению, в накопитель программы программного блока x11 заносится оператор "корень(x2)".

Возвращаемся к рассмотрению основной линии компиляции, откаты к которой происходят после обработки каждой из приведенных выше трех ветвей, соответствующих приемам с различными заголовками.

Эта линия начинается с пункта оглавления программ "Ввод дополнительных инф.элементов прогр.блока при просмотре элементов описания приема и обращении к справочнику ПРОГРБЛОК". Как видно из названия пункта, здесь просматриваются элементы x13 описания приема (фактически - указатели), и для каждого из них предпринимается обращение к справочнику "прогрблок" на логическом символе - заголовке указателя. Так как число различных типов указателей, порождающих те или иные специфические информационные элементы программного блока, весьма велико, то процедуры ввода таких элементов разнесены по соответствующим сравнительно маленьким программам данного справочника. Такая организация облегчает развитие компилятора при появлении новых типов указателей. Прием справочника "прогрблок" получает следующие входные данные: x1 - текущий указатель; x2 - заголовок приема; x3 - описание приема; x4 - программный блок; x5 - корень идентифицируемого подтерма; x6 - набор установок на идентификацию; x7 - теорема приема. Значение, возвращаемое справочником, несущественно, так как все необходимые изменения программного блока он выполняет сам.

Для примера, рассмотрим здесь одну типичную программу справочника "прогрблок" - она обрабатывает указатель "единица($ax_1 \dots x_n$)". Напомним, что этот указатель разрешает переменным $x_1, \dots, x_n$ принимать вырожденное "единичное" значение a. Входим в просмотр программы символа "единица" и, перемещаясь по "главному стволу" обращений, находим оператор "обращение(прогрблок)", с которого начинается работа рассматриваемого справочника. Напомним, что значением переменной x1 здесь является указатель "единица(...)". Переменной x8 присваивается символ единицы a; далее начинается просмотр переменных указателя, причем текущая переменная есть x10. Для заданного x10 переменная x12 перечисляет подтермы теоремы приема, которые должны быть идентифицированы. В каждом из таких подтермов x12 просматриваются вхождения x13 переменной x10. Если x13 расположено внутри функциональной переменной, то значением x14 становится вхождение этой функциональной переменной; иначе x14 равно x13. Рассматривается вхождение операции x15, операндом которой служит x14. Проверяется, что эта операция по операнду x14 имеет единичное значение x8. Если это так, то создается новый информационный элемент программного блока - (единица x15 x14 x8), указывающий на вхождение операции x15 и на ее операнд x14, который может принимать вырожденное единичное значение x8. Если x15 имеет лишь два операнда, то одновременно

с этим вводится также информационный элемент (заголовок x15), означающий, что заголовок подтерма, идентифицируемого с x15, может быть произвольным. Далее, анализируется особый случай - когда по x15 расположена коммутативная операция $f(xy)$; обе переменные x, y выделены рассматриваемым указателем "единица(...)", причем x15 само является операндом внешней некоммутативной операции x18, имеющей для x15 ту же единицу x8. Тогда вводятся еще два элемента - (единица x18 x15 x8) и (заголовок x18).

После цикла обработки указателей приема с помощью справочника "прогрблок" - переход через "ветвь 1".

Здесь рассматривается случай указателя "входвтерм(uA)". Такой указатель, при обработке его справочником "прогрблок", создает информационный элемент программного блока (извлечение v), где v - вхождение в теорему приема корня идентифицируемого подтерма B, соответствующее указателю вхождения u. Применение приема начинается с того, что некоторый терм задачи t идентифицируется с A. Этот терм t должен содержаться внутри терма T, который будет идентифицирован с B, однако где именно он располагается - несущественно. Первоначальная идентификация A с t нужна лишь для определения переменных, имеющихся в A, а далее она "забывается", и B идентифицируется с термом T независимым образом. Во всяком случае, известно, что B будет идентифицроваться с текущим термом задачи, расположенным по вхождению x3. Поэтому в x12 находится установка (корень ...) на идентификацию B, в программу приема вводится оператор, присваивающий вхождение первого символа текущего терма задачи переменной x15, и найденная установка заменяется на (операнд v x15).

После перехода через "ветвь 1" попадаем в фрагмент программы, содержащий контрольную точку "прием(19)". В нем рассматривается случай приема замены, у которого заменяемый терм - дизъюнкция либо конъюнкция. Если эта дизъюнкция либо конъюнкция имеет своим операндом переменную-модификатор, то нужно предусмотреть ситуацию, когда она (либо ее отрицание) будет идентифицироваться с подмножеством антецедентов кванторной импликации. Для этого вводится информационный элемент (длялюбого v), ссылающийся на вхождение v в теорему приема заменяемого терма.

В следующем фрагменте программы (после очередного перехода через "ветвь 1") вводится информационный элемент (уровень A), перечисляющий в наборе A все упоминаемые в фильтрах приема возможные уровни его срабатывания.

Опять переход через "ветвь 1" - теперь к контрольной точке "прием(23)", в которой рассматривается указатель приема "контекст(A)". Он определяет дополнительную по отношению к теореме приема ветвь идентификации той ситуации, которая описывается списком A идентифицирующих термов, фильтров и указателей. Для программирования ее вводится установка на идентификацию (контекст идентификатор(A)). Заметим, что обычно сначала реализуется именно дополнительная идентификация, и лишь после этого - идентификация, определяемая теоремой приема.

Следующий фрагмент рассматриваемой цепочки обрабатывает указатель приема "вводтерма(...)". Этот указатель связан со вводом вспомогательного обозначения x, определяемого антецедентом вида $t = x$. Фрагмент пополняет описание приема указателем "новаяпеременная(x)", чтобы в нужный момент компилятор смог выбрать для x первую неиспользуемую в задаче переменную.

Следующий фрагмент, содержащий контрольную точку "прием(26)", вводит информационные элементы (контроль x A) для тех теоремных переменных x, которые

встречаются только в подтермах T идентифицируемых термов приема, выделенных указателями "подстановка(...)". При идентификации прообразы таких подтермов T могут отсутствовать - это приведет к приписыванию некоторым переменным теоремы вырожденных "нулевых" значений. Например, при идентификации многочлена какой-либо его член ax^n может отсутствовать, и тогда переменной a будет сопоставлено значение 0. Здесь нужно контролировать идентификацию переменной x - она должна появиться хотя бы в одном явно присутствующем члене многочлена. Чтобы следить за идентификацией таких переменных x, которые встречаются только в "необязательно имеющихся" подтермах T, и вводится информационный элемент (контроль x A). У него A - список вхождений всех T. По окончании рассмотрения всех этих вхождений компилятор будет проверять, что переменная x идентифицирована с каким-либо термом.

Далее - фрагмент с контрольной точкой "прием(27)". Он пополняет программный блок элементами (заголовок v), указывающими те вхождения v в идентифицируемые термы, на которых ассоциативно-коммутативная операция может выродиться из-за того, что все ее операнды, кроме одного, будут идентифицированы с единичными либо нулевыми значениями. Эти элементы отменят вставку компилятором операторов, фиксирующих заголовок терма, идентифицированного с данным вхождением в теорему. В фрагменте рассматриваются только случаи вырожденной идентификации, определяемые указателями "единица(...)" и "подстановка(...)".

Следующий фрагмент цепочки - анализ указателя "внутрпосылка(...)", для которого создается информационный элемент (внутрпосылка ...) аналогичного формата, но с явным указанием вхождений в теорему.

Наконец, цепочка действий по созданию исходного комплекта информационных элементов программного блока и списка x12 установок на идентификацию завершается (контрольная точка "прием(28)"). Следует заметить, что рассмотренный фрагмент компилятора заполнялся вставками, обрабатывающими указатели приема, достаточно бессистемно. Это происходило постепенно, по мере проработки различных разделов. Многие из таких вставок обслуживают крайне редко встречающиеся ситуации, а некоторые - попросту единичные случаи. Вероятно, при последующем развитии компилятора большинство из них будет перенесено в справочник "прогрблок".

Теперь имеются программный блок x11 и список установок на идентификацию x12. Для реализации этого списка установок применяется процедура "идентификатор(начало(x9) x12 x11 x4 x6 пустоеслово)". Это - одна из самых больших процедур компилятора, и ее описанию будет посвящен отдельный раздел. Процедура создает ту часть программы приема, которая обеспечивает усмотрение ситуации для его применения - во всем, кроме учета фильтров приема.

Чтобы учесть фильтры, далее располагается обращение к процедуре "блокпроверок(x7 x4 x6 x11 x2)". Она рассматривает фильтры, создает для каждого из них один или группу операторов, обеспечивающих его проверку, и вставляет эти фильтры в уже имеющуюся заготовку программы приема. Обычно операторы фильтра располагаются как можно ближе к началу программы - сразу же, как только оказываются определены все их входные переменные. Это обеспечивает ускоренное отсечение ветвей программы при сканировании. Впрочем, здесь имеется множество исключений, и подробнее о них будет рассказано в разделе, посвященном компиляции фильтров приема.

Если при обращении к компиляции была задана установка на сохранение информации о соответствии теоремных и программных переменных (логический символ "знач" во входном набор x5), то после контрольной точки "прием(8)" реализуется

процедура, вставляющая в конец программы приема фиктивные операторы "знач(i A)", где i - символьный номер теоремной переменной x_i; A - программное выражение для терма, идентифицированного с x_i. Эти фиктивные операторы сохраняются на период дальнейших преобразований программы. Если будет сдвигаться нумерация программных переменных (что обычно и происходит при оптимизации программы), то соответствующий сдвиг затронет и операторы "знач". Номер i теоремной переменной сохранится, а переменные в программном выражении A окажутся скорректированы. По завершении создания программы операторы "знач(...)" будут из нее удалены, а информация о соответствии теоремных переменных программным - зарегистрирована в терминале "переменные" узла приема.

Далее - переход через оператор "ветвь 1" к фрагменту с контрольной точкой "прием(6)". Здесь проверяется наличие указателя "оценка(...)"; если такой указатель имеется, то вводится обращение к оператору "оценка(...)", обеспечивающему работу приема в двухуровневом режиме (сначала определяется наименьшая из предлагаемых различными приемами оценок, затем применяется прием с наименьшей оценкой). После этого - переход через "ветвь 1".

Здесь происходит обращение к процедуре "преобразователь(x7 x4 x6 x11 x2)". Эта процедура обеспечивает компиляцию тех операторов программы приема, которые собственно реализуют преобразования задачи. Подробнее она будет описана в отдельном разделе.

Далее реализуется процедура "вставкафрагментов(x11)". Она осуществляет вставку в накопитель программы приема тех дополнительных фрагментов программы F, которые в процессе компиляции, по различным причинам, не были сразу вставлены в основной список фрагментов, а оказались занесены в информационные элементы (блокпрограммы i F). Здесь i - номер ссылки на дополнительный фрагмент из основных фрагментов программного блока. Такая ссылка организуется в виде псевдооператора вида "1(A i)" либо "2(i)". Первый из них преобразуется при вставке в оператор "A иначе j"; второй - в "ветвь j", где j - номер дополнительного фрагмента F после вставки его в общий список фрагментов.

Для завершающего редактирования программы приема вводится специальная структура данных x13, называемая программным модулем. Она представляет собой пару (K,P), где P - извлеченный из программного блока набор фрагментов программы; K - набор наборов комментариев по редактированию этих фрагментов. Изначально списки комментариев берутся пустыми. Собственно редактирование программы выполняется процедурой "завершениепрограммы(x13 x1 x3 x4 x6)". Это - достаточно большая процедура, существенным образом сжимающая исходную сырую версию программы. Она будет рассмотрена в отдельном разделе.

По завершении редактирования программы из нее удаляются псевдооператоры "знач(i A)". Напомним, что у такого псевдооператора i есть номер теоремной переменной; A - программное выражение для терма, идентифицированного с этой переменной. Удаленные псевдооператоры преобразуются в термы "знач(xi A)" и регистрируются в терминале "переменные" узла приема.

После перехода через "ветвь 1"реализуется завершающая компиляцию вставка фрагментов программы в блок программ. Ее осуществляет оператор "записьприема(...)". Он выбирает в уже имеющейся ветви программы логического символа привязки ту точку, где новая программа ответвляется от ранее созданных, и осуществляет регистрацию подветви приема. Описанию устройства этого оператора будет посвящен отдельный раздел.

19.3 Схема компиляции приема пакетного нормализатора

Приемы пакетного нормализатора компилируются программой логического символа "замена" при обращении к ней от справочника "новыйприем". Приведем краткое описание последовательности действий, выполняемых программой. Рекомендуется прослеживать эти действия, войдя в программу, например, через пункт ("Компилятор ГЕНОЛОГа", "Приемы нормализаторов") оглавления программ.

В отличие от приемов сканирования задачи, которые могли быть активизированы при сканировании произвольной задачи, приемы нормализатора активизируются лишь при обращении к конкретной программе ЛОСа - данному нормализатору. Архитектура программы нормализатора может варьироваться в зависимости от формата нормализатора. Однако, общие прицины здесь одни и те же. Преобразуемый терм является значением переменной x1; список посылок, относительно которых осуществляются преобразования, является значением переменной x2; список комментариев к нормализатору - значение переменной x3. Приемы нормализатора инициируются в одном цикле, начало которого обозначено оператором "повторение". До этого оператора определяется текущий уровень сканирования: некоторой программной переменной, используемой в качестве указателя уровня, присваивается единица. После оператора "повторение" располагается оператор "ветвь ...", при откатах к которому либо происходит увеличение уровня сканирования на единицу, либо, по достижении максимального уровня, выдается результат - текущий вид изменяемого в цикле преобразований терма x1. После оператора "ветвь ..." - дерево, образованное программами приемов нормализатора. Главный ствол дерева - операторы, фиксирующие значение текущего уровня. В рамках одного уровня сканирования компилятор вклеивает новые программы приемов, находя точки, где их начальные отрезки начинают отличаться от ранее зарегистрированных программ. После каждого срабатывания приема происходит изменение значения x1, уменьшение текущего уровня сканирования до единицы, и откат к оператору "повторение" для перехода к очередному циклу.

19.3.1 Предварительное преобразование теоремы приема

Прежде всего, проверяется наличие в теореме приема геометрических понятий, и если они есть, то применяется процедура "развязка". Эти действия совершенно аналогичны действиям при компиляции приема сканирования задачи, уже рассмотренным выше. Далее - переход через "ветвь 2".

Здесь рассматривается случай указателя "связка(x)". Напомним, что этот указатель относится к теореме, у которой некоторый квантор по единственной переменной x может идентифицироваться с квантором того же типа, имеющим несколько переменных. Одна из них идентифицируется с x, а остальные переносятся без изменений в формируемые кванторы с единственной переменной x.

Переменной x11 присваивается список всех вхождений в теорему приема кванторов по переменной x. Выбирается новая (не входящая в теорему и в описание приема) теоремная переменная y, присваиваемая программной переменной x13. Определяется набор x14 результатов вставки в связывающие приставки кванторов списка x11 переменной y после переменной x. Переменная y будет идентифицироваться с остатком переменных связывающей приставки, возникающим после идентификации x. Просматриваются термы описания приема, и входящие в них кванторы с x преобра-

зуются таким же образом. Ввод новой переменной связывающей приставки в теорему мог привести к необходимости коррекции указателей выделенных в описании приема вхождений. Реализуется цикл таких коррекций. Затем в теореме приема вхождения списка x11 заменяются на термы списка x14. Наконец, вводятся два новых элемента описания приема: "кортежпеременных(y)", означающий, что переменная y может идентифицироваться со списком переменных произвольной длины, а также "Входит(y)", означающий, что не должна предприниматься проверка невхождения переменных, идентифицированных с y, в термы кванторов по y. Так как в этих термах не были предусмотрены функциональные переменные, разрешающие зависимость от y, то без данного указателя компилятор вставил бы операторы проверки невхождений. Затем - откат к оператору "ветвь 1" для перехода к следующему этапу преобразований теоремы приема.

После метки "прием(2)" располагается ветвь, обеспечивающая преобразование теоремы приема, консеквент A которой отличен от равенства либо эквивалентности, к виду "равно(A истина)". Если A имеет вид "не(B)", то вместо этого вводится "равно(B ложь)". В случае приема, осуществляющего лишь изменение комментариев нормализатора, консеквент теоремы может иметь вид "контекст(A)". Тогда применение приема начинается с усмотрения терма A; для удобства компиляции консеквент здесь преобразуется к виду "равно(A пусто)". Разумеется, никакой замены подтерма A на символ пустого множества в этом приеме не произойдет.

После отката через "ветвь 2" - замена символов "Величина" в теореме приема (если они есть) на "величина". В этой же ветви вводится, если она нужна, переменная - модификатор. Все это делается так же, как для приемов сканирования задачи. Затем - откат к переходу через "ветвь 1".

19.3.2 Создание начального отрезка программы

Перед вводом программного блока, в рамках которого будет происходить основная часть процесса компиляции приема, создается начальный отрезок его программы. Эти действия начинаются с контрольной точки "прием(4)"; перейти к ней можно из пункта "Формирование в x13 ..." раздела "Приемы нормализаторов" оглавления программ.

Переменной x10 присваивается вхождение корня заменяемого подтерма; переменной x11 - заголовок нормализатора. Переменной x12 присваивается набор, определяющий формат нормализатора. Затем инициализируется набор x13 операторов начального отрезка программы. В него заносятся операторы "обращение(0)" и "метка(икс(N))", где N - номер первой не определенной при обращении к нормализатору переменной. Если x12 содержит элемент "контрольтитра", указывающий, что может понадобиться пошаговая трассировка его действий, причем в x12 отсутствует элемент "контрольнормализации", определяющий возможность такой трассировки средствами оператора "контрольнормализации", то в накопитель x13 заносится оператор "контрольтитра(x1 x2 x3 x11)". Далее x14 становится равно первой не определенной в программе переменной. Если в x12 входит элемент "группировка", то в x13 вводится оператор инициализации накопителя вариантов преобразований (см. справочную информацию для логического символа "замены"). Если в x12 имеется элемент "контрольнормализации", то вводятся операторы "контрольнормализации(x1 x2 x3 x11 x y)", "не(равно(y 0))", "замена(1 x)". Они обеспечивают проверку отсутствия в буфере готового результата нормализации и замену обрабатываемого терма на извлекаемый из буфера частичный результат. Как правило, его в буфере

нет, и тогда терм просто повторяется без изменений. При наличии в x12 элемента "уровень(...)" добавляется оператор "равно(x14 1)", инициализирующий указатель текущего уровня сканирования. Этот оператор не вводится, если нормализатор имеет лишь один уровень сканирования.

Далее вводится оператор "повторение" для откатов в цикле преобразований. Если нормализатор предназначен для единственного преобразования, что распознается по наличию символа "1" в x12, и имеет лишь один уровень срабатывания, то точка отката не создается. Сразу заметим, что данный случай практически нигде не встречается. Непосредственно после оператора "повторение" размещается оператор "меткаперехода(1 y)". Здесь "1" - ссылка на точку отката; y - первая не определенная в точке отката программная переменная. Этот оператор - временный; он будет исключен по завершении компиляции и нужен лишь для вычисления параметров операторов, обеспечивающих откат после очередного преобразования к оператору "повторение". Откат должен быть абсолютно точным, так как иначе возможно отключение от преобразований различных групп приемов и выдача промежуточного результата в качестве ответа.

Далее из описания приема x6 извлекается указатель "уровень($n_1 \dots n_k$)", перечисляющий уровни срабатывания приема, и вводится оператор "или(равно(u n_1) ... равно(u n_k))", проверяющий совпадение текущего уровня u с одним из этих уровней. Если в описании приема уровень вообще указан, то он берется равным 1.

После перехода через "ветвь 1" - начало формирования операторов, идентифицирующих вхождение корня заменяемого подтерма и его заголовок. Эти действия выполняются после контрольной точки "прием(5)", выйти на которую можно через подраздел ("Учет заголовка заменяемого подтерма", "Исходная точка").

Во всех рассматриваемых после контрольной точки "прием(5)" подслучаях вхождение корня заменяемого подтерма присваивается переменной y, номер которой на 1 больше переменной - значения x14. Значением x14 пока является программная переменная, используемая в качестве указателя текущего уровня сканирования. Заголовок корня заменяемого подтерма присвоен переменной x15.

Прежде всего, рассматривается подслучай, в котором заменяемый подтерм имеет вид функциональной переменной $f(\dots)$, где f выделено указателем "отображение(f)". В этом подслучае требуется также наличие указателя "корень", означающего, что преобразуется корневое вхождение. Тогда вводится оператор "равно(y левыйкрай(x1))". Если этот подслучай не имеет места, то переход через "ветвь 2", иначе - откат к переходу через "ветвь 1" для анализа заголовка приема.

Прежде всего, далее рассматривается случай наличия указателя "корень" в наборе x12, описывающем формат нормализатора. Этот указатель означает, что все преобразования нормализатора - только корневые. После контрольной точки "прием(7)" разбирается подслучай, когда преобразуемый терм имеет вид $f(x\ t)$; f - ассоциативная и коммутативная операция; x - переменная, выделенная указателем "единица". Как и в случае функциональной переменной, заголовок терма здесь определен неоднозначно. Поэтому ограничиваемся вводом оператора "равно(y левыйкрай(x1))" и откатываемся к началу анализа заголовка приема. Аналогичные действия имеют место в случае идентификации матрицы. В прочих подслучаях заголовок заменяемого подтерма есть x15, так что можно ввести сначала оператор "заголовок(x1 x15)", фиксирующий этот заголовок, а затем добавить оператор "равно(y левыйкрай(x1))". Специально рассматорен подслучай указателя "комплексное", в котором вводится дизъюнкция условий на заголовок x1, учитывающих как веществennозначную, так и комплексную версию символа x15.

При отсутствии в x12 логического символа "корень" - переход через "иначе 1" к фрагменту, начинающемуся с контрольной точки "прием(9)". Здесь корень заменяемого подтерма может находиться в произвольной точке преобразуемого терма. Для некоторых нормализаторов предусмотрен специальный вспомогательный оператор, перечисляющий в качестве точек привязки лишь подмножество вхождений. Его заголовок P определяется справочником "новпозиция" по заголовку нормализатора x11. Если вспомогательный оператор есть, то в программу приема добавляются операторы "P(левыйкрай(x1)y)", "символ(y x15)". Второй из них вводится лишь тогда, когда заголовок заменяемого подтерма определен однозначно. Если прием имеет указатель "корень", то после указанных операторов добавляется оператор "корень(y)", отсекающий некорневые точки привязки. При отсутствии специального оператора P перечисление вхождений предпринимается оператором "позиция(y x1)"; остальные действия здесь - те же, что и выше.

19.3.3 Анализ заголовка приема

После контрольной точки "прием(11)" процесс компиляции разветвляется в зависимости от первого символа заголовка приема (будем называть его для краткости далее подзаголовком). Выйти на точку разветвления можно через пункт ("Классификация приемов по заголовкам", "Исходная точка") оглавления программ.

Прежде всего рассматриваются случаи подзаголовков "первыйтерм", "второйтерм", "замечание".

Это - общий случай компиляции. Первые два подзаголовка соответствуют обычным преобразованиям замены; последний - указывает на действия с комментариями нормализатора.

Инициализация программного блока начинается с ввода накопителя x16 его информационных элементов. Если формат нормализатора предусматривает наличие списка посылок, то в x16 заносится элемент (списокпосылок x2), указывающий, что этот список является значением программной переменной x2. Для случая, когда заголовок заменяемой части теоремы приема есть переменная x, в x16 заносятся элементы (корень y) и (вхождение x (y)). Здесь y - все та же увеличенная на единицу переменная x14, что и в предыдущем подразделе. Элемент (вхождение ...) определяет программное выражение для вхождения терма, идентифицированного с переменной x. Если в накопителе программы x13 имелся оператор "меткаперехода(...)", то в x16 заносится элемент (меткаперехода 1), указывающий, что символ 1 уже использован для обозначения метки перехода.

Вводится накопитель x17 набора установок на идентификацию. Если заменяемая часть теоремы представляла собой переменную, то x17 инициализируется пустым словом. Иначе - в него заносится установка (операнд x10 y) для идентификации заменяемой части теоремы x10 с подтермом, вхождение которого задается программной переменной y. Установки на идентификацию антецедентов вводятся специальной процедурой "смантецеденты". Ее действия аналогичны разобранному выше случаю создания установок на идентификацию антецедентов в приемах сканирования задачи. После пополнения набора x17 такими установками - переход через "ветвь 2"

Здесь, после контрольной точки "прием(15)", рассматривается особый случай, когда заменяемая часть представляет собой функциональную переменную $f(x)$ либо $f(x_1 \ldots x_n)$, причем замена - корневая. В этом случае сразу вводится информационный элемент программного блока (функция f ...), определяющий идентифицированную функциональную переменную f, и исключается установка на идентификацию

заменяемого подтерма (операнд ...).

Наконец, после перехода через "ветвь 1", вводится заготовка x18 программного блока. В ней набор информационных элементов x16 пополнен стандартными элементами, характеризующими компилируемый прием в целом. Сразу после ввода программного блока добавляется его информационный элемент (операнд x10 t), определяющий программное выражение t для вхождения, идентифицированного с корнем x10 заменяемой части.

После контрольной точки "прием(17)" происходит ввод информационных элементов: (конец A), (блокпроверок A), (мнимаячасть A), аналогичных рассмтренным для сканирования задачи. Здесь A - списки вхождений корней антецедентов, к которым относится информация.

После контрольной точки "прием (18)" - попытка усмотреть в теореме приема переменную - модификатор, и ввод соответствующего ей информационного элемента.

После контрольной точки "прием(19)" - цикл обращений к справочнику "прогрблок" для ввода информационных элементов, соответствующих различным указателям приема. После перехода через "ветвь 1" - ввод информационных элементов (контроль ...), сопровождающих ранее введенные элементы (подстановка ...). Все эти действия совершенно аналогичны ранее рассмотренным для приемов сканирования задачи.

После контрольной точки "прием(20)" предпринимается обработка указателей приема "вход($f\ x_1 \dots x_n$)": из списка комментариев извлекается элемент ($fa_1 \dots a_n$), и переменные $x_1, \dots x_n$ идентифицируются с термами $a_1, \dots, a_n$. Результаты идентификации регистрируются в информационных элементах (терм $x_i\ A_i$), где A_i - программное выражение для a_i.

Далее компиляция снова разветвляется: если прием имеет указатель "набор(первыйтерм)", определяющий режим одношаговой "разгруппировки" всего набора операндов рассматриваемой в приеме двуместной ассоциативно-коммутативной операции, то применяется одна ветвь программы, иначе - другая.

Компиляция в случае режима разгруппировки

Это - очень редкий случай, когда теорема приема может быть представлена в виде $P(f(a,b)) = g(P(a), P(b))$; f, g - ассоциативно-коммутативные операции; a, b - переменные, не имеющие других вхождений в терм $P(\dots)$, кроме указанных; заменяемой частью является $P(f(a,b))$. Такая теорема, при наличии указателя "набор(первыйтерм)", определяет преобразование термов вида $P(f(a_1 \dots a_n))$ в термы $g(P(a_1), \dots, P(a_n))$.

Прежде всего, в заменяемой части находится вхождение x19 операции $f(a,b)$; теоремные переменные a, b присваиваются программным переменным x21 и x22. Вводится пара x23 термов $P(a), P(b)$, и проверяется, что заменяющая часть представима как $g(P(a), P(b))$. Создается копия x26 программного блока x18, к информационным элементам которой добавляется набор (параметры (a,b)). Он указывает, что переменные a, b в действительности идентифицироваться не должны: ведь число операндов у f может оказаться произвольным. Идентифицироваться будет лишь вхождение $f(a,b)$. Далее из списка установок на идентификацию удаляются все элементы (корень ...), ссылающиеся на антецеденты, содержащие a либо b; одновременно из информационных элементов программного блока x26 удаляются соответствующие наборы (антецедент ...). После этого (контрольная точка "прием(22)") происходит обращение

к процедуре "идентификатор" для создания той части программы приема, которая идентифицирует заменяемый терм (без конкретизации операндов у $f(\ldots)$), а также все антецеденты теоремы приема, не содержащие переменных a, b.

После перехода через "ветвь 3" начинается рассмотрение тех антецедентов теоремы приема, которые содержали a либо b. Прежде всего, проверяется, что существуют такие антецеденты и что нет антецедента, содержащего a и b одновременно. Затем создается вспомогательный программный блок x29, в который переносятся из x26 все информационные элементы, кроме элементов (выводимо A), определявших программные переменные либо выражения A для списков использованных при обосновании корректности утверждений. Этот блок сопровождается элементом (вхождение a y), указывающим программную переменную y, которая будет пробегать вхождения операндов операции $f(\ldots)$, последовательно идентифицируя их с a. Создается список x30 заготовок установок на идентификацию антецедентов, содержащих a, и для идентификации их предпринимается обращение к процедуре "идентификатор". По завершении идентификации определяется конъюнкция x30 всех операторов программы вспомогательного блока. Она нужна для проверки условий, которым должен удовлетворять каждый операнд операции $f(\ldots)$. В конце x30 добавляются операторы, пересылающие в накопитель утверждения, использованные при проверках. Далее с помощью x30 формируется кванторный оператор x31, реализующий просмотр операндов операции $f(\ldots)$ и выполнение соответствующих проверок. После вставки этого оператора в программу блока x26 - откат к переходу через "ветвь 1".

Здесь водится оператор "учетобоснований(...)", регистрирующий в комментарии нормализатора (выводимо ...) информацию о посылках, использованных для обоснования корректности замены.

После перехода через "ветвь 1" создается еще один вспомогательный программный блок x28. Он нужен для формирования списка операндов заменяющего терма $P(a_1), \ldots, P(a_n)$. В x28 заносится информационный элемент (вхождение a y), указывающий ту программную переменную y, которая будет последовательно пробегать вхождения операндов операции $f(\ldots)$, идентифицируемые с a. Программа этого блока инициализируется единственным оператором "операнд(t y)", где t - программное выражение для вхождения операции $f(\ldots)$. Затем переменной x29 присваивается теоремный терм $P(a)$, и операторное выражение "метаперевод(x29 x28)" дает в качестве своего значения x30 программное выражение, значением которого служит операнд $P(\ldots)$ заменяющего терма, соответствующий текущему операнду операции $f(\ldots)$. Далее переменной x31 присваивается набор всех новых программных переменных $z_1, \ldots z_k$, введенных в блоке x28 по сравнению с блоком x26. Переменной x32, после перехода через "ветвь 3", присваивается операторное выражение "выписка(z_1 $\ldots z_k$ R x30)", где R - конъюнкция всех операторов программы блока x28. Значением этого выражения и служит список термов $P(a_1), \ldots, P(a_n)$. Переменной x33 присваивается теперь программное выражение для заменяющего терма $g(P(a_1), \ldots , P(a_n))$.

Далее проверяется наличие информационного элемента (быстрпреобр ...), определяющего обработку заменяющего терма нормализаторами. Учет нормализаторов для его подтермов $P(a_i)$ уже произошел при использовании процедуры "метаперевод". В цикле просмотра нормализаторов, связанных с заменяющим термом, корректируется выражение x33, определяющее этот терм. По завершении этих действий - переход через "ветвь 1".

Для удобства дальнейших ссылок на заменяющий подтерм вводится оператор "равно(x34 x33)", присваивающий его вспомогательной переменной x34. Переменной

x35 присваивается программное выражение для результата преобразования. В случае корневого вхождения заменяемого подтерма оно совпадает с x34, иначе - является результатом замены в текущем терме x1 заменяемого подтерма на заменяющий. Далее просматриваются фильтры приема, для них создаются группы операторов программы, выполняющих проверку фильтра, и эти группы добавляются в конце созданной части программы приема. Затем - переход через "ветвь 1".

Если формат нормализатора содержит символ "1", означающим, что ответ выдается после первого же преобразования, то программа приема завершается оператором, выдающим ответ x35. Иначе - вводится оператор "контрользамены(...)", проверяющий целесообразность выполнения замены в контексте каких-либо внешних общих установок (например, при блокировке применения данного приема) и позволяющий войти в отладчик ЛОСа непосредственно перед заменой при трассировке. Если предусмотрено сопровождение срабатывания приема специальными пояснениями, то добавляется обеспечивающий работу с ними оператор "титр(...)" (переменной x36 присваивается до его вставки в программу либо "пустоеслово", либо одноэлементный набор из данного оператора). В случае приема, срабатывающего на уровне, большем единицы, добавляется оператор, уменьшающий текущий уровень до единицы. Наконец, программу приема завершают операторы "замена(1 x35)", изменяющий текущий терм x1 на x35, и "переходпометке(1)" - для отката к оператору "повторение", начинающему очередной цикл сканирования.

По завершении программы приема - откат к переходу через "ветвь 2". Здесь предпринимаются завершающая обработка программы и запись ее в блоке программ. Используются те же макрооператоры, что и в случае приемов сканирования задачи.

Компиляция в общем случае

Если прием не имел указателя "набор(первыйтерм)", то его компиляция продолжается с контрольной точки "прием(25)". Выйти на эту контрольную точку можно через пункт ("Приемы нормализаторов","Классификация приемов по заголовкам", "Приемы ПЕРВЫЙТЕРМ, ВТОРОЙТЕРМ, ЗАМЕЧАНИЕ", "Формирование идентифицирующей части приема") оглавления программ. Сразу после контрольной точки - обращение к процедуре "идентификатор", создающей идентифицирующую часть программы приема. Здесь x17 - набор установок на идентификацию; x18 - программный блок.

Если прием не имеет указателя "вывод(...)" и либо нормализатор является корневым, либо его формат не содержит элемента "коррекцияпосылок", то предпринимается добавление оператора "учетобоснований(...)", регистрирующего в комментарии (выводимо ...) использованные при замене посылки. В остальных случаях такой оператор будет добавляться позднее, так как оставшаяся часть программы в этих случаях может пополнять список использованных посылок. Затем - переход через "ветвь 1".

Здесь, после контрольной точки "прием(26)", осуществляется учет фильтров приема. Прежде всего, если в фильтрах встречается терм вида "контекст(... буфер(A t) ...)", то для логического символа A, указывающего заголовок используемого в качестве буфера комментария нормализатора, вводится оператор "равно(x пустоеслово)". Здесь x - программная переменная, используемая в качестве накопителя заносимых в буфер термов. Одновременно добавляется информационный элемент программного блока (буфер A x), с помощью которого будет происходить компиляция операторов, заполняющих данный накопитель.

Далее - переход через "ветвь 2", где переменной x20 присваивается список фильтров приема. Из него тут же исключаются фильтры, уже обработанные ранее - их список сохраняется в информационном элементе (комментарии ...). Фильтры x21 списка x20 просматриваются, и по каждому из них процедурой "фильтр(...)" создается оператор, реализующий x21. Конъюнктивные члены этого оператора вставляются процедурой "вставкафильтра" в уже созданную часть программы приема. По мере возможности, они размещаются ближе к началу программы - для наискорейшего отсечения. После обработки фильтров - откат к переходу через "ветвь 1".

Здесь начинается создание операторного выражения для заменяющего подтерма. Переменной x19 присваивается заменяющий терм теоремы. Если формат нормализатора имел элемент "группировка", то в x19 отбрасывается, если она есть, переменная - модификатор. Затем происходит обращение к процедуре "метаперевод", определяющей программное выражение x20 для заменяющего подтерма. Если уже имелся информационный элемент (результат ...), содержащий ранее найденное программное выражение для заменяющего подтерма, то x20 переопределяется согласно этому элементу. Затем - переход через "ветвь 2".

После контрольной точки "прием(28)" - учет указателя приема "дробь". Такой указатель означает, что при идентификации двуместной корневой операции заменяемого терма могут быть переставлены ее операнды; тогда аналогичная перестановка выполняется и для заменяющего терма. Прежде всего, переменной x21 присваивается вхождение заменяемого терма, а переменной x22 - его корневая двуместная операция. Если выражение x20 для заменяющего терма имеет вид "запись(x22 t_1 t_2)", то x20 преобразуется в условное выражение "вариант(P x20 запись(x22 t_2 t_1))". Здесь оператор P проверяет наличие перестановки при идентификации корневых операндов B_1, B_2 заменяемого терма теоремы. Отсутствие перестановки распознается по совпадению первого операнда заменяемого подтерма и того вхождения, которое было идентифицировано с B_1. Если заменяющий терм не имел того же заголовка f, что и заменяемый, то предусмотрена попытка искусственного представления его в этом виде, с использованием единицы операции f. После учета указателя "дробь" откат к переходу через "ветвь 1".

Здесь происходит учет другого указателя - "знаксуммы(...)", определяющего возможность одновременного изменения знаков у всех операндов некоторой операции. Роль знака играет некоторая одноместная операция s. Находится ранее созданный по этому указателю информационный элемент (знаксуммы ...) и проверяется, что идентификация с возможной заменой знаков предпринималась для корневой операции заменяемого подтерма. Тогда программное выражение x20 для заменяющего терма модифицируется так, чтобы его знак s изменялся одновременно с изменением знаков идентифицируемых операндов. При этом используется вспомогательная процедура "измзнака".

После перехода через "ветвь 1" x20 преобразуется к формату терма; в x16 сохраняется программная переменная, значением которой служит вхождение корня заменяемого подтерма; вводится информационный элемент (результат x20), указывающий программное выражение для заменяющего терма. Далее (контрольная точка "прием(29)")обрабатываются указатели приема, определяющие добавление или удаление комментариев. x21 - текущий такой указатель. Если его первый операнд не имеет вида "условие(...)", то процедура "транскоммент" определяет набор программных выражений для разрядов комментария. Затем вводится оператор, выполняющий добавление либо удаление этого комментария. Напомним, что в случае указателя "замечусловие(...)" копии уже имевшихся комментариев не регистрируются; в

случае указателя "замечание(...)" - регистрируются. Если первый операнд указателя имеет вид "условие(P)", то действие выполняется лишь при истинности фильтра P. Процедура "фильтр" находит оператор x24 проверки истинности P. Затем, как и в безусловном случае, вводится оператор для добавления или исключения, но уже не комментария, а определяемого условным выражением набора комментариев. Если x24 истинно, то этот набор одноэлементный, иначе - пустой.

При переходе через "иначе 3" рассматривается случай ввода нового комментария к текущей задаче. При переходе через "иначе 2" - учет указателей "нормализация(...)", "учетнормализации(...)", которые позволяют передавать комментарии внешнему нормализатору с заданным заголовком. В первом случае берутся все такие нормализаторы, во втором - лишь первый из них. С помощью оператора "смисточник" организуется просмотр источников текущего стэкового кадра интерпретатора ЛОСа. Обычным образом создаются программное выражение для комментария и оператор, проверяющий условие его ввода. Все это объединяется в одну кванторную импликацию, которая и присоединяется к программе.

После обработки указателей, определяющих действия с комментариями, - откат к переходу через "ветвь 1". Если прием имел заголовок "замечание", то на этом его программа и завершается: в ее конце размещается оператор "продолжение". После этого происходит откат к ветви, выполняющей редактирование программы и ее запись в блок программ. Иначе - следующий переход через "ветвь 1". Здесь происходит учет фильтров "длина(P)", сравнивающих заменяющий и заменяемый подтермы в смысле оценки, определяемой процедурой сравнения P. После такого учета - переход к фрагменту с контрольной точкой "прием(31)".

Если не имеет места случай преобразования особого типа, причем заменяемый подтерм не совпадает со всем преобразуемым термом, то переменной x20 переприсваивается программное выражение для результата замены вхождения x16 в преобразуемый терм x1 на заменяющий терм x20.

Далее сначала рассматривается подслучай, когда заменяющая часть теоремы имеет вид "разборслучаев(...)". Здесь вводится оператор "равно(x x20)", присваивающий новой программной переменной x фиктивный заменяющий терм "разборслучаев(...)". Вводится оператор "контрольслучаев(x...)", блокирующий повторное рассмотрение альтернатив. Вводятся операторы "входит(подчинено x3)иначе n ответ(x)", которые выдают в качестве ответа терм "разборслучаев(...)", если имеется комментарий "подчинено", свидетельствующий, что разбор случаев адресован внешнему нормализатору; в противном случае они организуют переход к новому, пока еще не зарегистрированному n-му фрагменту программы. Наконец, вводится этот фрагмент программы, содержащий операторы: "замена(6 0)" (обнуление текущего уровня сканирования для отката к процедуре, организующей выбор рассматриваемого подслучая); "подслучаи(...)" (регистрация установки на разбор подслучаев согласно терму "разборслучаев(...)"); "переходпометке(1)" (откат к началу цикла сканирования).

Если прием не был связан с разбором случаев, то переход через "иначе 1". Если нормализатор выполняет лишь однократное преобразование, то добавляется оператор для выдачи ответа. Иначе - переход через "иначе 1".

Если нормализатор не является нормализатором группировки, то сначала проверяется, является ли программное выражение x20 переменной; если не является, то вводится оператор "равно(x x20)", где x - новая программная переменная. Затем x20 переприсваивается эта переменная x. Это делается, чтобы избежать переписывания неоднобуквенного выражения x20, которое может оказаться входным данным не-

скольких вспомогательных процедур. Далее - переход через "ветвь 2".

Если прием имеет фильтр "контрольодз", то вставляется оператор "контрольодз(...)", проверяющий, что все подтермы нового терма имеют о.д.з., включающую область истинности посылок. Затем - переход через "ветвь 1", где после контрольной точки "прием(36)" предпринимается вставка оператора "контрользамены(...)", уже встречавшегося ранее при компиляции приема "разгруппировки". Он бывает нужен при блокировке срабатывания приема из соображений внешнего характера, а также для трассировки.

После контрольной точки "прием(44)" обрабатываются указатели "вывод(...)", определяющие добавление новых посылок. Для них вводятся операторы "учетвывода(...)", реализующие такое добавление. Если указатель сопровождается фильтром, ограничивающим область его действия, то x23 присваивается оператор проверки фильтра и вводится, вместо указанного выше, оператор "или(не(x23) учетвывода(...))". По окончании просмотра указателей "вывод" аналогичным образом обрабатываются указатели "удалениеусловия(...)". Затем - откат к переходу через "ветвь 1".

Если прием имеет указатель "вывод(...)", либо нормализатор не является корневым, но имеет комментарий (коррекцияпосылок ...), то переменной x21 присваивается набор наборов использованных посылок. Вводится оператор "учетобоснований(...)", регистрирующий эти посылки в комментарии нормализатора (выводимо ...). Далее - переход через "ветвь 1".

Если имеется информационный элемент (буфер $A\ x$), то вводится оператор, проверяющий непустоту набора x и переносящий все его элементы в накопитель N комментария к нормализатору ($A\ N$). Затем - переход через "ветвь 1". Здесь происходит разветвление процесса компиляции.

Сначала рассматривается случай, когда нормализатор не является нормализатором группировок. Если прием имеет указатель "стоп", то в его программу вставляется оператор прерывания "трассировка(стоп 0)", который будет вызывать выход в отладчик ЛОСа непосредственно перед реализацией замены. Далее - переход через "ветвь 2".

Переменной x22 присваивается программное выражение для скорректированного после преобразования списка посылок нормализатора, и переход через "ветвь 1".

Здесь рассматривается случай приемов, имеющих указатель "дизъюнктчлен(...)". Этот указатель выделяет антецеденты вида $A = B$, снабженные также указателем "идентификатор(...)". У них сначала формируется терм A, а затем происходят последовательные попытки идентификации B с дизъюнктивными членами терма A. Для каждого такого дизъюнктивного члена находится соответствующий вариант заменяющего терма, после чего фактический заменяющий терм строится как дизъюнкция этих вариантов. Информационный элемент (дизъюнктчлен $A_1\ A_2\ A_3\ A_4$) хранит: список A_1 вхождений антецедентов, выделенных указателями "дизъюнктчлен"; программную переменную A_2 для накопителя вариантов заменяющего терма; метку перехода A_3, выводящую на точку формирования накопителя A_2; набор A_4 меток перехода для откатов к продолжению рассмотрения дизъюнктивных членов. Переменной x26 присваивается программное выражение для дизъюнкции термов накопителя; к фрагменту, завершающему перечисление дизъюнктивных членов, добавляются операторы, реализующие замену и откат к началу цикла сканирования, а к текущему фрагменту - операторы пополнения накопителя термом x20 и отката к продолжению перечисления дизъюнктивных членов. Далее - откат к редактированию и регистрации программы. При отсутствии у приема указателя "дизъюнктчлен" - переход через "ветвь 1".

При наличии указателя "титр(...)" в список x23 заносится оператор "титр(...)", обеспечивающий выдачу поясняющего текста при срабатывании приема. Далее - переход через "ветвь 1", где формируется набор x24 операторов, заносимых в конец программы. Прежде всего, это оператор, корректирующий значение текущего уровня сканирования. Если прием имеет указатель "выход", определяющий обрыв преобразований, то текущий уровень заменяется на символьный номер "стандменьшеилиравно", заведомо превосходящий число уровней срабатывания нормализатора; иначе он заменяется на 1. Далее размещаются: список x23 (пустой либо содержащий оператор "титр(...)"; оператор, реализующий замену; оператор, корректирующий список посылок согласно x22. Список x24 присоединяется к концу программы, и добавляется оператор отката к началу сканирования. На этом формирование программы завершается, и далее - откат к ветви редактирования программы и ее регистрации.

Компиляцию редко встречающихся приемов нормализатора группировок, а также компиляцию простейших приемов "спускоперандов" и "лексупорядочение" рассматривать здесь не будем, так как эти случаи легко разобрать самостоятельно по тексту программы.

Для полного представления о компиляции нормализаторов полезно также самостоятельно разобрать случай приемов "окончание", обеспечивающих переключение текущего уровня сканирования. Начальная часть программы приема "окончание" воспроизводит общий начальный отрезок всех приемов данного нормализатора. Сразу после оператора "повторение", организующего цикл сканирования, происходит ответвление к продолжению этой программы. Кроме увеличения текущего уровня, проверки превышения им максимального значения и выдачи результата, она может выполнять также ряд других действий, определяемых форматом нормализатора. Заметим, что инициализация программы нормализатора должна начинаться только с приема "окончание". Если при работе с нормализатором на какой-то момент все его другие приемы удаляются, то необходимо удалить также и программу приема "окончание", снова ее после этого восстановив. Иначе добавление новых приемов будет происходить неправильно.

19.4 Схема компиляции приема проверочного оператора

Приемы проверочного оператора компилируются программой логического символа "спуск" при обращении к ней от справочника "новыйприем". Войти в начальную точку компиляции можно через пункт ("Компилятор ГЕНОЛОГа", "Приемы проверочных операторов", "Исходная точка") оглавления программ.

19.4.1 Предварительное преобразование теоремы приема

.

Обычно теорема приема проверочного оператора имеет вид кванторной импликации $\forall_{x_1 \ldots x_n}(A_1 \ldots A_m \to A_0)$, консеквент A_0 которой представляет собой результат подстановки некоторых термов в шаблон утверждений, проверяемых оператором. При применении приема эти термы идентифицируются с входными данными проверяемого отношения.

Компиляция начинается с проверки наличия указателя "не". Этот указатель означает, что прием нужен не для установления истинности рассматриваемого утверждения, а для обрыва действий оператора при установлении его ложности. Соответственно, консеквент A_0 теоремы приема будет иметь вид отрицания $\neg B_0$ утверждения B_0, полученного некоторой подстановкой в шаблон. Однако, идентификацию проверяемого утверждения здесь нужно будет проводить не с консеквентом A_0, а с его отрицанием B_0. Поэтому после контрольной точки "прием(21)" и происходит замена консеквента на B_0.

Далее, после перехода через "ветвь 2", для геометрических приемов предпринимается обращение к оператору "развязка". На этом предварительные преобразования теоремы приема и завершаются.

19.4.2 Создание начального отрезка программы

После модификации теоремы приема переменной x9 присваивается вхождение ее консеквента. Вводятся накопители: x10 - первая неиспользуемая программная переменная; x11 - набор пар (входная переменная оператора - соответствующее ей вхождение в консеквент теоремы приема); x12 - число уровней срабатывания оператора. Накопители x11, x12 сначала инициализируются нулями, а затем доопределяются с помощью справочников "легковидеть" и "проверка", соответствующих обычным и усиленным проверочным операторам. Чтобы воспользоваться таким справочником, фиксируется корневое вхождение x9 проверяемого утверждения, а затем просматриваются расположенные внутри x9 вхождения x13 различных логических символов x14. Как только результат обращения к справочнику на символе x14 для вхождений x9,x13 дает тройку, начинающуюся с названия проверочного оператора, указанного в заголовке приема, из этой тройки извлекается информация, необходимая для определения значений x11, x12. Далее - переход через "ветвь 1".

После контрольной точки "прием(2)" начинается формирование начального отрезка x14 программы приема. Предварительно переменной x13 присваивается копия исходной версии набора x11, так как сам этот набор будет впоследствии изменяться. В набор x14 сначала заносятся лишь два оператора - "программа" и "метка(икс(N))", где номер N первой не определенной программной переменной получается увеличением x10 на 4. Это вызвано тем, что кроме набора входных переменных проверяемого отношения, оператор получает еще 3 входных данных - список посылок, список комментариев и номер той выходной переменной, которой будет присвоен набор использованных при проверке посылок.

После ввода накопителя x14 начального отрезка программы создается набор x15, составленный из всех входных переменных оператора (без единственной его выходной переменной). С помощью x15 формируется обращение к оператору "контрольбуфера($\ldots x\ y$)", используемому для поиска в буфере готового результата по данному входному набору. В x14 заносятся это обращение и оператор $x = 0$, проверяющий, что поиск в буфере ничего не дал. Лишь тогда имеет смысл переходить к попытке реализации приема. Заметим, что для использования результата, найденного в буфере, имеется специальный прием пакета, заголовок которого - "контрольбуфера". Его программа ответвляется от программ обычных приемов при проверке условия $x = 0$.

Так как обычно прием "контрольбуфера" компилируется перед "основными" приемами пакета, имеющими заголовок "спуск", то фактически в программе находится оператор "не($x = 0$)", после которого идут операторы приема "контрольбуфера",

выдающие найденный результат. Ответвление по "иначе" после "не($x = 0$)" ведет к ветви программ основных приемов. У всех этих программ оператор $x = 0$ оказывается автоматически отброшен при их вклейке в общую программу пакета. Далее - переход через "ветвь 1" и коррекция номера первой не использованной в x14 программной переменной x10. Если число x12 уровней срабатывания пакета более одного, то на этот момент x10 - переменная, которая будет использоваться в пакете как номер текущего уровня срабатывания. К x14 добавляется оператор "или(равно(x10 1)... равно(x10 N))", последовательно перечисляющий уровни срабатывания от 1 до максимального уровня N. Затем находится фактический уровень срабатывания приема x16, определяемый его фильтрами, и вставляется оператор "равно(x10 x16)", фиксирующий этот уровень. Заметим, что в отличие от нормализаторов, для проверочных операторов не нужно будет обеспечивать откаты к точке перечисления уровней после срабатывания приемов, так как каждое срабатывание означает получение окончательного результата.

После перехода через "ветвь 1" - контрольная точка "прием(3)", связанная с учетом симметричности ("коммутативности") проверяемого отношения. Если отношение симметрично и прием не имеет указателя "коммутативно", блокирующего попытку перестановки операндов при идентификации, то проверяется симметричность консеквента. Если он не симметричен относительно входных выражений, то к x14 добавляется оператор "или(и(равно(x x1)равно(y x2))и(равно(x x2)равно(y x1)))", реализующий попытку перестановки входных данных x1,x2 при идентификации. Далее в качестве входных данных будут выступать уже переменные x, y, и в наборах x11, x13 предпринимается замена x1,x2 на эти переменные. После анализа симметричности - переход через "ветвь 2". Здесь к концу набора x14 добавляются операторы, присваивающие очередным программным переменным корневые вхождения входных термов. В наборе x11 ссылки на программные переменные для входных термов заменяются ссылками на переменные для корневых вхождений. Далее - откат к переходу через "ветвь 1".

Здесь вводится накопитель x15 информационных элементов программного блока. Последовательно просматриваются вхождения x16 в набор x11. В этом просмотре элементы набора x11 играют роль предварительных установок на идентификацию входных подтермов консеквента: часть из них будет идентифицирована сразу же (их позиции в наборе x11 заменятся на 0), а на основе остальных впоследствии будут введены обычные установки на идентификацию. Для текущего вхождения x16 в накопитель x15 заносится элемент (выражение ...), указывающий программную переменную для соответствующего входного подтерма. Затем - переход через "ветвь 2". Переменной x17 присваивается текущая пара набора x11. Конец пары x17 - вхождение x18 в консеквент теоремы приема подтерма, соответствующего текущему входному данному. Начало пары x17 - программная переменная x для корневого вхождения, с которого нужно начинать идентификацию данного подтерма. Для x18 рассматриваются следующие случаи:

1. По вхождению x18 расположена переменная x19. Проверяется наличие в x15 для этой переменной элемента, указывающего, что она уже идентифицирована с некоторым вхождением. Если такой элемент есть, то к x14 присоединяется оператор, проверяющий равенство подтермов по обоим вхождениям. Иначе - в x15 заносится элемент (вхождение x19 x), указывающий, что x19 идентифицирована с подтермом по вхождению x. При этом в набор x11 по вхождению x16 заносится ноль, указывающий, что для данного входного терма идентификация

уже проведена.

2. По вхождению x18 расположена функциональная переменная "значение(f y)", где f выделено указателем приема "отображение(...)". В x15 заносится информационный элемент (функция f (y) 0 (вхождение x)), определяющий идентификацию данной функциональной переменной. По вхождению x16 регистрируется ноль.

3. По вхождению x18 расположен терм "префикс(A B)". Он определяет выделение некоторого элемента A в рассматриваемом наборе. Здесь анализируются следующие подслучаи:

   а) Данный терм выделен указателем приема "список(...)", означающим, что определяемый термом набор идентифицируется без учета порядка своих элементов. В свою очередь, здесь анализируются подслучаи:

   а1) B - переменная; A само имеет вид "префикс(C D)" и выделено указателем "список(...)". Тогда к x14 добавляются операторы "символ(x набор) операнд(x y) символ(y набор)", а к x15 - элемент (терм B остатокнабора(x (y))), идентифицирующий B как остаток набора x после выделения в нем элемента - набора y. По вхождению x16 размещается пара (y v), где v - вхождение подтерма "префикс(C D)". Эта пара позволит продолжить идентификацию подтерма A.

   а2) B - переменная, причем A не имеет вида, указанного в предыдущем подслучае. К x14 добавляются операторы "символ(x набор) операнд(x y) символ(y f)", где f - логический символ, являющийся заголовком терма A. К x15 присоединяется элемент (терм B остатокнабора(x (y))). По вхождению x16 размещается та же пара, что и в предыдущем подслучае.

   а3) A и B - переменные. Тогда к x14 добавляются операторы "символ(x набор) операнд(x y)". Если в x15 уже имелся элемент (терм A ...), то добавляется оператор, проверяющий равенство термов, идентифицированных с A по старому и новому вхождениям. В любом случае, к x15 присоединяются элементы (терм A подтерм(y)) и (терм B остатокнабора(x (y))), а по вхождению x16 регистрируется 0.

   б) Имеется указатель приема (префикс ...), ссылающийся на вхождение x18. Это означает, что подтерм "префикс(...)" должен идентифицироваться буквально, т.е. с термом вида "префикс(...)".

   в) A, B - переменные, причем набор идентифицируется с учетом порядка элементов. К x14 добавляется оператор "символ(x набор)", а к x15 присоединяются элементы (терм A первыйтерм(x)), (терм B остатокнабора(x (первыйоперанд(x)))). По вхождению x16 регистрируется 0.

4. По вхождению x18 расположен терм "отображение(...)", выделенный указателем "развертка(...)". К x14 присоединяется оператор "символ(x набор)".

5. По вхождению x18 расположен терм "набор(...)". К x14 добавляется оператор, фиксирующий число элементов этого набора.

6. Остаточный случай, реализуемый, если предыдущие случаи не имели места. В этом случае к x14 добавляется оператор "символ(x f)", где f - заголовок

подтерма по вхождению x18. Если этот подтерм представляет собой логический символ, то по вхождению x16 регистрируется 0.

По окончании просмотра вхождений x16 в набор x11 - переход через "иначе 1". Здесь, после контрольной точки "прием(5)", происходит регистрация в x15 информационных элементов (замечание ...) и (списокпосылок ...), сохраняющих для дальнейшего использования ссылки на программные переменные, значениями которых служат список комментариев и список посылок проверочного оператора.

Далее - переход через "ветвь 1", и после контрольной точки "прием(6)" вводится накопитель x16 установок на идентификацию. Он формируется из элементов (операнд v x) для всех пар (x v), имеющихся в наборе x11. Если какое-либо вхождение v выделено указателем приема "развертка(...)", то соответствующая установка заменяется на (развертка v x 0).

После перехода через "ветвь 1" - контрольная точка "прием(7)". Если прием не имеет указателя "дистрибразвертка(...)", требующего специального режима компиляции, то предпринимается обращение к процедуре "смантецеденты" для пополнения списка x16 установками на идентификацию антецедентов. Попутно в список x15 информационных элементов программного блока заносится набор (конец ...), перечисляющий вхождения всех антецедентов, обработка которых отнесена на конец программы приема. Затем - переход через "ветвь 1", где из набора x11 исключаются все разряды, равные нулю.

19.4.3 Ввод программного блока

Вводится заготовка x17 программного блока. Кроме перечисленных в наборе x15, информационными элементами этого блока становятся стандартные общие элементы (теорема ...), (точкапривязки ...), (логсимвол ...) и т.п. Добавляются элементы (подтерм ...), определяющие корневые вхождения идентифицируемых подтермов консеквента приема, отличных от подтерма, содержащего точку привязки. Наконец, добавляются элементы (антецедент ...), перечисляющие корневые вхождения непосредственно идентифицируемых антецедентов. В программном блоке регистрируется уже созданный начальный фрагмент x14 программы приемы. Если набор x11 непуст, то первая его пара определяет точку привязки приема, и тогда добавляется информационный элемент (операнд ...), указывающий для точки привязки программную переменную, имеющую своим значением вхождение этой точки привязки.

После контрольной точки "прием(9)" предпринимается регистрация информационного элемента (параметры ...), перечисляющего те переменные теоремы, для которых по различным причинам идентификация не проводится. Например, она отменяется для переменных консеквента при наличии указателя "дистрибразвертка". В этом случае консеквент имеет двуместную ассоциативно-коммутативную операцию $f(x, y)$, которая фактически будет идентифицироваться с операцией $f(...)$ от произвольного числа операндов, т.е. указывать конкретные значения для x, y бессмысленно. Далее - переход через "ветвь 1".

После контрольной точки "прием(10)" расположен цикл обращений к справочнику "прогрблок" на указателях приема x19. Этот справочник, как уже говорилось выше, осуществляет пополнение списка информационных элементов программного блока, необходимых для компиляции. После контрольной точки "прием(11)" - регистрация в информационных элементах программного блока списков антецедентов, обрабатываемых проверочными либо усиленными проверочными операторами. Далее - переход через "ветвь 1".

Если имеется информационный элемент (заголовок v), указывающий, что при идентификации подтерма теоремы по вхождению v возможно различие заголовков, а некоторый оператор уже введенной части программы проверяет совпадение данных заголовков, то этот оператор удаляется.

19.4.4 Компиляция идентифицирующей части приема и его фильтров

После контрольной точки "прием(13)" выполняется обращение к процедуре "идентификатор(...)", создающей идентифицирующую часть программы приема. Здесь же, после контрольной точки "прием(14)" происходит создание операторов для проверки фильтров приема и вставка их в программу. Затем - переход через "ветвь 1".

После контрольной точки "прием(15)" рассматривается случай приемов, имеющих указатель "дистрибразвертка(K)". Теорема таких приемов представима в виде $\forall_{y_1 \ldots y_n}(P(\ldots a \ldots)\ \&\ P(\ldots b \ldots) \rightarrow P(\ldots h(a\ b) \ldots))$, где h - ассоциативно-коммутативная операция. K - указатель вхождения в теорему подтерма $h(a\ b)$. Указатель "дистрибразвертка" определяет режим реализации приема с обобщением на произвольное число операндов у операции h.

Прежде всего, проверяется наличие у приема указателя "спуск". Он означает, что при неудачной попытке применения приема (после идентификации его входных данных) должен происходить выход из проверочного оператора по истинностному значению "ложь". Соответственно, после контрольной точки "прием(19)" вводится фрагмент программы x19, обеспечивающий такой выход при откате. Фрагмент x19 будет включен в программу приема перед обращением к ее редактированию, а пока он регистрируется в информационном элементе (блокпрограммы 1 x19). Точка перехода к нему обозначается в программе псевдооператором "2(1)", который впоследствии окажется преобразован в оператор "ветвь". Далее - переход через "ветвь 2".

Здесь определяется вхождение x19 выделенной указателем "дистрибразвертка" операции $h(\ldots)$. Находится тот входной терм t проверочного оператора, внутри которого расположено x19. Выбирается новая переменная x, и определяется результат x22 замены в терме t вхождения x19 на переменную x. Терм x22 будет служить шаблоном для антецедентов теоремы приема, возникающих при переходе от многоместной операции $h(a_1 \ldots a_n)$ консеквента к ее операндам $a_1, \ldots, a_n$. В программный блок x17 заносится информационный элемент (вхождение $x\ y$), определяющий программную переменную y, которая будет пробегать вхождения операндов операции h. После этого находится программное выражение x23 для терма x22 - оно будет служить одним из входных данных рекурсивного обращения к тому же самому проверочному оператору. Другие входные данные этого обращения берутся из набора x13; накопителем входных термов обращения служит набор x24. По окончании его заполнения переменная x25 становится равна той входной переменной проверочного оператора, значением которой служит список посылок, а x26 - переменной, значением которой служит список комментариев. Если прием имел указатель "комментарий(...)", определяющий ввод некоторого комментария при рекурсивном обращении, то значение x26 заменяется на программное выражение, дающее пополненный список комментариев. После этого - переход через "ветвь 1", где переменной x27 присваивается программное выражение для рекурсивного обращения к проверочному оператору. Наконец, переменной x29 присваивается список операторов, содержащий оператор "длялюбого(...)", реализующий цикл просмотра операндов операции h и обращений к оператору x27. Этот список присоединяется к программе приема. Программная

переменная "начало(x17)", играющая в операторах из x29 роль накопителя ссылок на использованные при проверках посылки, регистрируется в информационном элементе (выводимо ...). Далее - откат к оператору "ветвь 1" перед ветвью, относящейся к указателю "дистрибразвертка". Переход через этот оператор продолжает основное русло компиляции программы приема.

При наличии указателя приема "стоп" в программу вставляется контрольная точка "трассировка(стоп 0)", вызывающая выход в отладчик ЛОСа после выполнения всех необходимых для срабатывания приема проверок.

После контрольной точки "прием(16)" находится программное выражение x21 для списка всех использованных приемом посылок. Если прием не имел указателя "не", то его программа завершается операторами "блок(...)", "учетвбуфере(...)", "результат(...)", "выход". Первый из этих операторов выполняет блокировку применения приема, если она задана специальным комментарием, а также регистрирует в комментариях к посылкам исходной задачи факт срабатывания приема, если установлен режим работы решателя с регистрацией срабатываний. Второй оператор регистрирует результат обращения к проверочному оператору в буфере - для непосредственного извлечения его оттуда при повторном обращении. При наличии текстформульного сопровождения вставляется реализующий его оператор "титр(...)". Если прием имел указатель "не", то программа завершается операторами "блок(...)", "учетвбуфере(...)", "внешобрыв(...)", "стоп". В обоих случаях далее предпринимается завершающее редактирование программы и регистрация ее в блоке программ.

19.5 Схема компиляции приема синтезатора

Приемы синтезатора компилируются программой логического символа "значение" при обращении к ней от справочника "новыйприем". Войти в начальную точку компиляции можно через пункт ("Компилятор ГЕНОЛОГа", "Приемы синтезаторов", "Исходная точка") оглавления программ. Схема компиляции представляет собой некоторое усложнение схемы компиляции проверочных операторов.

19.5.1 Предварительный анализ приема

В случае синтезаторов единственное предварительное преобразование теоремы приема - обращение к оператору "развязка" в случае геометрических приемов. Далее переменной x9 присваивается вхождение консеквента теоремы, переменной x11 - название синтезатора, переменной x12 - набор (T вход($x_1 \dots x_n$) выход($y_1 \dots y_k$) A_1 $\dots A_m$). Здесь T - реализуемое синтезатором утверждение; $x_1, \dots, x_n$ - список входных переменных синтезатора, $y_1, \dots, y_k$ - список выходных переменных, $A_1, \dots, A_m$ - список термов, задающих формат синтезатора. Утверждение T присваивается переменной x13. Значениями переменных x14 и x15 становятся списки переменных $x_1, \dots x_n$ и $y_1, \dots, y_k$ соответственно. Проверяется, что консеквент теоремы приема представим как результат некоторой подстановки в терм T. Переменной x19 присваивается набор пар (входная программная переменная синтезатора - вхождение в консеквент теоремы приема, идентифицируемое со значением этой переменной). После этого значением x18 становится программная переменная для списка посылок синтезатора. Значение x18 сохраняется для последующего использования в переменной x20. Затем оно увеличивается на 2 и становится равно первой выходной переменной синтезатора. Создается набор x21 пар (выходная программная переменная синтезатора

- вхождение в консеквент теоремы того терма, который определяет ее значение). После этого значением x18 становится выходная программная переменная для списка использованных синтезатором посылок. Данное значение сохраняется в переменной x22. Наконец, x18 увеличивается на единицу и становится первой не определенной переменной на начальный момент выполнения программы синтезатора.

19.5.2 Создание начального отрезка программы

После контрольной точки "прием(3)" переменной x23 присваивается набор операторов начального отрезка программы синтезатора: "программа", "метка(икс(...))", "синтезатор(0)". Оператор "синтезатор(0)" нужен для трассировки на уровне срабатываний приемов.

Переменной x24 присваивается список входных программных переменных синтезатора. После контрольной точки "прием(20)" осуществляется пополнение списка x23 оператором Q, проверяющим наличие комментария (вход $fA_1 \dots A_n$), где f - название синтезатора, $A_1, \dots A_n$ - его входные данные. Наличие такого комментария означает появление цикла при рекурсивных обращениях; в этом случае оператор Q ложен, и цикл разрывается. Если комментария нет, то оператор Q вводит его, блокируя таким образом зацикливание при последующих рекурсивных обращениях.

Если в формате синтезатора имеется символ "перечисление", определяющий режим перечисления его выходных значений, то добавляется также оператор, исключающий ранее введенные комментарии (перечисление ...) и вводящий новый такой комментарий (перечисление пустоеслово). В этом комментарии будут сохраняться наборы значений выходных переменных - для предотвращения повторений в последовательности выходных наборов.

Далее - откат к переходу через "ветвь 2", где реализуется добавление оператора "контрольбуфера(...)", предпринимающего попытку усмотреть готовый результат обращения в буфере. После него размещается оператор "равно(x18 0)", определяющий отсутствие в буфере нужного результата - лишь тогда нужен компилируемый прием. Если же результат в буфере найден, то для выдачи его, как и в случае проверочных операторов, имеется отдельный прием с заголовком "контрольбуфера". Значение x18 снова устанавливается на первую неиспользуемую программную переменную, и откат к переходу через "ветвь 1".

Здесь вводится накопитель x24 информационных элементов программного блока. Для каждой входной программной переменной x синтезатора добавляется оператор "равно(y левыйкрай(x))", присваивающий новой переменной y корневое вхождение входного терма; в парах набора x19 вместо переменных x помещаются соответствующие переменные y. После этого x18 по-прежнему указывает первую неиспользованную программную переменную.

После контрольной точки "прием(5)" расположен цикл предварительной идентификации корневых точек входных термов. Просматриваются вхождения x25 в набор x19 пар (программная переменная для корневого вхождения во входной терм - вхождение v в консеквент теоремы приема подтерма, идентифицируемого с данным входным термом). Переменной x26 присваивается вхождение v; переменной x27 - символ по этому вхождению.

Если этот символ предствляет собой переменную x, то проверяется наличие в накопителе x24 информационного элемента (вхождение x ...), указывающего ранее идентифицированное ее вхождение. Если такого элемента нет, то он вводится, иначе - добавляется оператор, проверяющий совпадение термов по старому и новому вхожде-

ниям, идентифицированным с x. В обоих случаях разряд x25 набора x19 заменяется на ноль, означающий, что дальнейшая идентификация для данного входного терма не требуется.

Если по вхождению x27 расположена не переменная, то переход через "иначе 2". Здесь сначала анализируются особые случаи, аналогичные описанным ранее для проверочных операторов: идентификация x27, имеющего вид "префикс(x y)", а также случай вхождения v, выделенного указателем "развертка(...)". При отсутствии особых случаев - вводится оператор "символ(...)", фиксирующий совпадение заголовка входного терма с заголовком соответствующего теоремного подтерма. Если по вхождению v был расположен однобуквенный терм, то дальнейшая идентификация здесь не нужна, и разряд x25 заменяется на 0.

После просмотра всех пар набора x19 - откат к переходу через "ветвь 1". Здесь, после контрольной точки "прием(6)", происходит регистрация в накопителе x24 информационных элементов (замечание ...) и (списокпосылок ...), указывающих программные переменные для списка комментариев и списка посылок. По оставшимся ненулевыми элементам набора x19 формируется накопитель x25 списка установок на идентификацию. При наличии указателей приема "развертка(...)", выделяющих корневые вхождения входных термов, установки (операнд ...) на идентификацию этих вхождений преобразуются в установки (развертка ...).

Далее - переход через "ветвь 1", где после контрольной точки "прием(7)" предпринимается пополнение списка x25 установками на идентификацию антецедентов. Как и в случае других пакетных операторов, для этого используется процедура "сантецеденты". Затем в накопителе x24 регистрируются информационные элементы, ссылающиеся на антецеденты, для обработки которых применяются проверочные либо усиленные проверочные операторы. После отката к переходу через "ветвь 1" из x19 исключаются все нулевые разряды; в нем остаются только пары, определяющие входные термы, для которых требуется продолжение идентификации в соответствии с установками набора x25.

19.5.3 Ввод программного блока

После контрольной точки "прием(9)" вводится программный блок x26. Кроме элементов набора x24, в него заносятся стандартные информационные элементы, характеризующие контекст компиляции. Далее создаются информационные элементы (операнд ...), связывающие корневые вхождения подлежащих дальнейшей идентификации входных термов с программными переменными для этих вхождений. После контрольной точки "прием(10)" - цикл обращений к справочнику "прогрблок" для указателей приема, пополняющих список информационных элементов программного блока. После контрольной точки "прием(13)" - создание информационного элемента (параметры ...), перечисляющего теоремные переменные, для которых идентификация не нужна. Затем - откат к переходу через "ветвь 1". Здесь просматриваются информационные элементы (заголовок ...), указывающие корневые вхождения входных термов, для которых заголовок определен неоднозначно. Если уже были введены операторы программы, фиксирующие эти заголовки, то они отбрасываются.

19.5.4 Компиляция основной части программы приема

После контрольной точки "прием(15)" - обращение к процедуре "идентификатор", создающей идентифицирующую часть программы приема. Далее заполняются накопитель x27 программных выражений для операндов завершающего прием оператора "результат(...)", а также накопитель x28 программных выражений для выходных термов синтезатора. Здесь используются пары набора x21, связывающие выходные программные переменные с вхождениями в теорему приема выходных термов. Программные выражения для этих термов даются процедурой "метаперевод". После заполнения накопителей x27, x28 - переход через "ветвь 1".

Начиная с контрольной точки "прием(16)" - создание операторов, реализующих фильтры приема, и вставка их в программу.

Если синтезатор перечисляющий, то добавляется оператор, проверяющий отсутствие в комментарии (перечисление ...) текущего набора x28 значений выходных переменных и регистрирующий этот набор в данном комментарии. Таким образом предотвращается повторная выдача одного и того же набора.

При наличии указателя приема "стоп" вставляется контрольная точка "трассировка(стоп 0)", обеспечивающая выход в отладчик ЛОСа перед выдачей результата.

После контрольной точки "прием(17)" предпринимается вставка завершающих операторов программы. Прежде всего, формируется программное выражение x32 для списка использованных приемом посылок. Затем добавляются операторы "блок(...)" и "учетвбуфере(...)", играющие ту же роль, что и в программах проверочных операторов. При необходимости добавляется оператор "титр(...)". Наконец, добавляются операторы "результат(...)" и "выход" (у перечисляющих синтезаторов последний отсутствует). Далее выполняются: редактирование программы; вставка в ее начальной части оператора, фиксирующего уровни срабатывания приема, указанные в его фильтрах; регистрация программы в блоке программ.

19.6 Схема компиляции приема анализатора

Приемы анализатора компилируются программой логического символа "внутрвывод" при обращении к ней от справочника "новыйприем". Войти в начальную точку компиляции можно через пункт ("Компилятор ГЕНОЛОГа", "Приемы анализаторов", "Исходная точка") оглавления программ.

Обращения к основным пакетным операторам - нормализаторам, проверочным операторам, синтезаторам и анализаторам - можно рассматривать как упрощенные версии задач, соответственно, на преобразование, доказательство, описание и исследование. Функционирование анализатора, по аналогии с решением задачи на исследование, состоит в выводе следствий из заданного списка посылок в соответствии с заданной целевой установкой. Наиболее ценные из них отбираются и передаются внешней задаче. При обращении к анализатору задаются значения трех программных переменных: x1 - текущая задача Z, x2 - максимальный уровень применяемых приемов анализатора, x3 - целевая установка вспомогательной задачи анализатора Z'. В последнюю автоматически передается из задачи Z цель, перечисляющая неизвестные, и эту цель в списке x3 указывать не нужно.

Вспомогательная задача Z' создается помещаемым в начале программы анализатора оператором "анализатор(x1 x2 x3 x4 x5)". Значения переменных x1,x2,x3 - те же, что в обращении к анализатору. Переменной x4 присваивается вспомогательная

задача Z', а переменной x5 - лимит трудоемкости, отведенный для работы анализатора. Для организации сканирования анализатора, в зависимости от его типа, используется одна из процедур "теквхожд", "корнвхожд", "Корнвхожд". Обращения к ним имеют вид "теквхожд(x1 x2 x3 x4)", "корнвхожд(x1 x2 x3 x4)", "Корнвхожд(x1 x2 x3)". Входные данные суть: x3 - вспомогательная (внутренняя) задача анализатора и x4 - текущий уровень сканирования. x1 перечисляет вхождения логических символов в посылки задачи x3, а x2 - вхождения этих посылок в список посылок. В первом случае рассматриваются только посылки, вес которых не превосходит x4. Во втором случае - посылки, имеющие вес x4, причем в качестве x1 берутся только корневые вхождения. В третьем случае тоже берутся только корневые вхождения, но вес посылки игнорируется. Тип используемого анализатором сканирования определяется по его названию справочником "анализатор". Если справочник выдает 1, используется оператор "корнвхожд", если он выдает 2, используется "теквхожд", и если выдает 3, то используется "Корнвхожд".

Приемы анализатора осуществляют пополнение списка посылок внутренней задачи на исследование. Те следствия, которые по завершении работы анализатора должны быть перенесены в список посылок внешней задачи, снабжаются комментарием "внешвывод". Вывод следствий прекращается либо при исчерпании возможностей применения приемов, либо при исчерпании лимита на число шагов работы интерпретатора, либо при реализации приема, усматривающего получение необходимого результата и реализующего выход из анализатора.

В дополнение к приемам вывода следствий, введены также приемы анализатора, осуществляющие эквивалентные либо тождественные преобразования термов внутренней задачи. Эти приемы имеют указатель "внутрпреобр".

19.6.1 Создание начального отрезка программы

Как и в случае других пакетных операторов, компиляция приема анализатора начинается с предварительного преобразования теоремы приема. Для геометрического приема используется обращение к процедуре "развязка"; других преобразований теоремы пока не предусмотрено.

После перехода через "ветвь 2" переменной x9 присваивается вхождение консеквента теоремы; переменной x10 - название анализатора; переменной x11 - указатель типа используемого анализатором сканирования. После контрольной точки "прием(37)" расположено обращение к процедуре "точкапривязки", выбирающей точку привязки в теореме приема. Затем инициализируется накопитель x14 начального отрезка программы. В него заносятся операторы "программа", "метка(икс(4))", "анализатор(x1 x2 x3 x4 x5)", "лимит(x5)", "равно(x6 0)", повторение", "меткаперехода(1 x7)", "теквхожд(...)" либо "корнвхожд(...)" либо "Корнвхожд(...)", "символ(x7 A)". Здесь A - символ привязки, выбранный процедурой "точкапривязки". Оператор "анализатор(...)" нужен для создания внутренней задачи и организации трассировки по шагам работы анализатора. Он же определяет по входным данным лимит x5 трудоемкости обращения к анализатору. После того, как этот лимит будет превышен, произойдет откат к ветви программы анализатора, которая позднее будет вставлена между операторами "анализатор(...)" и "лимит(x5)". Эта ветвь представляет собой отдельный прием, завершающий работу анализатора. Заголовком завершающего приема служит символ "внешвывод", фиктивной его теоремой - терм "анализатор(S)", где S - название анализатора. Переменная x6 имеет своим значением текущий уровень сканирования. Оператор "повторение" представляет со-

бой точку отката после реализации очередного приема. Чтобы адресовать переход к этой точке, на период компиляции после нее размещается фиктивный оператор "меткаперехода(...)".

Находится указатель приема "уровень(...)", перечисляющий допустимые уровни его срабатывания, и к набору x14 добавляется оператор, проверяющий совпадение x6 с одним из этих уровней. Затем - переход через "ветвь 1".

После контрольной точки "прием(2)" вводится накопитель x15 информационных элементов программного блока, в который заносится ряд стандартных общих характеристик компилируемого приема. Здесь же вводится накопитель x16 установок на идентификацию. Переменной x17 присваивается набор корневых вхождений антецедентов теоремы приема. Просматриваются указатели приема, определяющие специальную обработку антецедентов, и по ним предпринимается заполнение набора x16 установками на идентификацию. Соответствующие разряды набора x17 обнуливаются. Если прием имеет указатель "внутрпреобр", а также указатель "теквхожд(...)", определяющий точку привязки не в заменяемом терме, а в одном из антецедентов теоремы, то в x16 заносится установка на идентификацию (теквхожд ...). После перехода через "ветвь 1" переменной x18 присваивается список всех ненулевых элементов набора x17, т.е. всех еще не учтенных в установках на идентификацию корневых вхождений антецедентов. Если прием не имеет указателя "теквхожд(...)", адресованного антецеденту, выделенному указателем "усм", то в x16 заносится установка на идентификацию (операнд ...) для точки привязки. В накопитель x15 заносятся информационные элементы, характеризующие точку привязки. По элементам набора x18 создаются установки на идентификацию (корень ...).

19.6.2 Ввод программного блока

После контрольной точки "прием(7)" вводится программный блок x19.

Если прием имеет указатели "конец(...)", откладывающие обработку некоторых антецедентов на конец программы приема, то создается информационный элемент программного блока (конец ...), перечисляющий корневые вхождения таких антецедентов. Предпринимается регистрация в информационных элементах ссылок на те антецеденты, для обработки которых используются проверочные либо усиленные проверочные операторы. Затем - откат к переходу через "ветвь 1".

Здесь реализуется цикл обращений к справочнику "прогрблок" для указателей приема, в процессе которого создаются необходимые дополнительные информационные элементы. Далее - снова переход через "ветвь 1".

Если прием имеет указатель "вход($A\ x_1 \ldots x_n$)", определяющий идентификацию теоремных переменных $x_1, \ldots, x_n$ из комментария ($A\ t_1 \ldots t_n$), то добавляются оператор, извлекающий данный комментарий, а также информационные элементы (терм $x_i\ T_i$), идентифицирующие переменные x_i с термами t_i.

Если прием имеет указатель "внутрпреобр", то добавляется оператор, проверяющий, что текущее утверждение относится к буферу анализатора, а не к посылкам внешней задачи.

19.6.3 Компиляция основной части программы приема

После контрольной точки "прием(11)" - обращение к процедуре "идентификатор", создающей идентифицирующую часть программы приема. Как и для пакетов рассматривавшихся выше типов, далее создаются и вставляются в программу операторы

проверки фильтров приема; при наличии указателя "стоп" - добавляется контрольная точка "трассировка(стоп 0)". После этого переменной x20 присваивается подтерм теоремы приема, определяющий выводимое утверждение (при наличии указателя "внутрпреобр" - заменяющее утверждение). Оператор "метаперевод" находит программное выражение x21 для этого подтерма. Если x21 отлично от переменной, то добавляется оператор, присваивающий формируемый приемом новый терм вспомогательной переменной, и x21 заменяется на эту переменную. Затем - переход через "ветвь 3".

После контрольной точки "прием(14)" происходит учет указателей "замечусловие(...)", "замечание(...)". По этим указателям к программе добавляется оператор, вводящий новый комментарий анализатора. В первом случае ("замечусловие") дублирующий комментарий не регистрируется; во втором случае дублирование допускается.

После контрольной точки "прием(27)" проверяется наличие указателей приема "новаяпосылка(...)", определяющих уменьшение весов посылок при срабатывании приема. По этим указателям создаются соответствующие операторы.

Далее компиляция разветвляется. Отдельно рассматриваются случаи приема вывода, распознаваемого по отсутствию указателя "внутрпреобр", и приема замены, обладающего таким указателем. В качестве операторов, реализующих преобразования вспомогательной задачи, используются те же операторы "вывод" и "заменавхождения", что и для приемов сканирования задач. Формирование их входных данных происходит аналогично тому, как это делалось ранее.

19.7 Схема компиляции приема оператора фильтра

Приемы оператора фильтра компилируются программой логического символа "контекст" при обращении к ней от справочника "новыйприем". Войти в начальную точку компиляции можно через пункт ("Компилятор ГЕНОЛОГа", "Приемы оператора фильтра", "Исходная точка") оглавления программ.

Обращение к оператору фильтра имеет вид $P(x_1 \dots x_n\ y_1 \dots y_m)$, где $x_1, \dots, x_n$ - входные переменные; $y_1, \dots, y_m$ - выходные переменные. Обычно $m = 0$. Однако, число входных переменных у программы оператора P может быть больше n, так как дополнительно ей сообщается информация о контексте срабатывания приема, использующего оператор фильтра. Например, для приема, срабатывающего при сканировании задачи, дополнительно передаются значения первых 5 переменных контекста сканирования; для приема нормализатора - значения первых 3 переменных, определяющих обращение к нормализатору. Дополнительные программные переменные размещаются перед основными переменными $x_1, \dots, x_n, y_1, \dots, y_m$, так что нумерация последних сдвигается. Тип внешнего контекста считается для оператора фильтра фиксированным, он определяется в формате оператора.

Теорема приема оператора фильтра вырожденная (логический символ "блок") и на экран не выводится. Компиляция происходит таким образом, что в качестве этой теоремы можно брать терм $P(x_1 \dots x_n\ y_1 \dots y_m)$.

Прием оператора фильтра, не имеющего выходных переменных, проверяет истинность условия на текущий контекст, определяемого фильтрами этого приема. Если условие истинно, то происходит выход из оператора фильтра по значению "истина". Если ни один прием не сработал, то оператор фильтра оказывается ложным. Фактически, оператор фильтра реализует дизъюнкцию условий, задаваемых его приемами.

19.7.1 Предварительный анализ приема

Переменной x9 присваивается заголовок оператора фильтра; переменной x10 - терм "условие(A)", у которого A - конъюнкция фильтров приема. Переменной x11 присваивается набор всех указателей приема. Затем вводится вспомогательный терм "идентификатор(B)", у которого B представляет собой набор всех фильтров и указателей приема. Этот терм будет впоследствии рассматриваться как фильтр "контекст(B)" и обрабатываться программой справочника "блокпроверок" на символе "контекст". Отличие от обработки обычного фильтра "контекст(...)" будет иметь место лишь на завершающем этапе компиляции. Во вспомогательном программном блоке, создаваемом при компиляции фильтра "контекст(...)", возникнет группа операторов, реализующих проверку условий B. Вместо ввода внешнего квантора существования по новым переменным для этой группы операторов, она будет перенесена без каких-либо изменений в основной программный блок приема.

Переменной x13 присваивается набор указателей формата оператора фильтра. Значением переменной x15 далее становится набор однобуквенных термов - типов входных данных. Возможны следующие типы: "терм", "вхождение", "логсимвол", "переменная". Переменной x16 присваивается набор идущих по возрастанию номеров переменных, число которых равно общему числу $m + n$ входных и выходных переменных оператора фильтра. На первом месте в этом наборе стоит переменная с номером 1. Далее - переход через "ветвь 2".

Переменной x18 присваивается число дополнительных входных переменных, определяющих внешний контекст. Логический символ "решить" в формате оператора указывает на режим сканирования задачи; символ "замена" - на наличие внешнего нормализатора.

Далее вводится заготовка x19 набора информационных элементов программного блока. С учетом константы сдвига x18, в нее заносятся элементы, определяющие идентификацию теоремных переменных $x_1, \ldots x_n$ (т.е. входных переменных оператора).

19.7.2 Ввод программного блока и завершение компиляции

Вводится программный блок x20; в его накопителе программы имеются всего два оператора - "программа" и "метка(икс(...))". Если оператор фильтра используется приемом нормализатора, то вводится информационный элемент (списокпосылок ...), указывающий на список посылок этого нормализатора как на логический контекст приема оператора фильтра. Затем предпринимается обращение к процедуре "учетнормализаторов(...)", создающей информационные элементы (быстрпреобр ...) для имеющихся у приема указателей нормализации. Эти элементы будут использоваться процедурой "метаперевод" при определении программных выражений для новых термов (такие термы могут встречаться в фильтрах приема).

Наконец, происходит обращение к справочнику "блокпроверок" на логическом символе "контекст" для обработки введенного ранее терма "идентификатор(B)". Этот терм был присвоен переменной x12. Фактически, это основная часть процесса компиляции приема. По завершении ее в накопитель программы программного блока x20 будут занесены операторы, проверяющие истинность фильтров приема.

Если оператор фильтра не имеет выходных переменных, то его программа завершается оператором "выход". Иначе - переход через "иначе 2". Здесь вводятся набор x23 выходных "теоремных" переменных приема и набор x25 типов их значе-

ний. Определение значений выходных переменных оператора фильтра осуществляется теми фильтрами приема, которые представляют собой идентифицирующие термы. После обращения к справочнику "блокпроверок" эти значения уже должны быть определены, быть может, в формате, отличающемся от указанного в наборе x25. С помощью процедуры "прогрвыражение" находятся программные выражения для данных значений в нужном формате, регистрируемые в наборе x26 поочередно с выходными программными переменными оператора. По набору x26 создаются завершающие программу операторы "результат(...)" и "выход".

По завершении программы приема - откат к переходу через "ветвь 1" для редактирования и записи ее в блок программ.

19.8 Компиляция приемов вычислительных пакетов

Компиляция вычислительных пакетов находится в стадии развития. Приемы пакета продукций компилируются справочником "новыйприем" на логическом символе "продукция"; приемы технического анализатора - справочником "новыйприем" на логическом символе "знач".

19.9 Информационные элементы программного блока

После того, как выше были приведены общие схемы компиляции приемов различных типов, перейдем к более подробному описанию основных блоков компилятора, использованных в этих схемах. На различных этапах компиляции интенсивно будут использоваться информационные элементы программного блока, так что прежде всего приведем достаточно представительный перечень типов таких элементов. Опускаем здесь уже приводившиеся выше информационные элементы, несущие общую информацию о приеме. Так как список типов информационных элементов постоянно пополняется, а некоторые редко используемые уже имеющиеся элементы ниже не приведены, то для получения более полной информации рекомендуется использовать справочное оглавление, достижимое из редактора программ по Ctr-п.

Приводимый в этом разделе материал имеет чисто справочный характер. Перечисляемые здесь информационные элементы оторваны от того контекста, в котором они появляются и используются. Этот контекст будет восстанавливаться по мере описания блоков компилятора, работающих с данными элементами.

19.9.1 Идентификация теоремных переменных

В процессе компиляции программы приема находятся программные выражения, определяющие, прямо либо косвенно, значения теоремных переменных. Чтобы этими программными выражениями можно было воспользоваться на последующих этапах компиляции, они регистрируются в информационных элементах программного блока. Перечислим основные такие информационные элементы, а также элементы, задающие различные ограничения и пожелания, учитываемые при идентификации теоремных переменных. Каждый пункт начинается с указания вида элемента, после чего идут пояснения.

1. (вхождение x A). x - теоремная переменная; A - программное выражение для вхождения терма, идентифицированного с x.

2. (терм x A). x - теоремная переменная; A - программное выражение для терма, идентифицированного с x. Обычно теоремные переменные идентифицируются с вхождениями подтермов, так как при этом не требуется переписывать подтерм. Идентификация с термом означает, что подтерм переписывается. Иногда это приходится делать из-за того, что он представлен в неявном виде (например, как подмножество операндов ассоциативно-коммутативной операции); иногда - в процессе формирования заменяющего терма выписывается подтерм, заданный до этого своим вхождением, и тогда элемент (вхождение x ...) дополняется элементом (терм x ...). При наличии такой избыточности компилятор выбирает в каждом конкретном случае элемент, для которого получаются более простые операторы программы.

3. (унисборка x A B). x - теоремная переменная; A - логический символ; B - программное выражение для набора термов. Переменная x идентифицируется с термом, определяемым программным выражением "унисборка(A B)". Элемент используется в тех случаях, когда удобнее работать со списком корневых операндов терма, идентифицированного с x.

4. (наборчленов A x B). x - теоремная переменная; A - логический символ; B - программное выражение для набора A - членов терма, идентифицированного с x. Незначительная модификация предыдущего информационного элемента, у которого B могло быть одноэлементным набором из терма, имеющего заголовок A.

5. (переменная x A). Теоремная переменная x идентифицирована с переменной либо логическим символом, являющимися значением программного выражения A.

6. (значение x A). A есть программное выражение для значения теоремной переменной x, введенной некоторым идентифицирующим термом фильтра приема. Элемент используется в тех случаях, когда тип объекта, идентифицированного с x, неясен (например, при идентификации разряда комментария).

7. (наборпеременных x A). x есть теоремная переменная; A - программное выражение, определяющее набор идентифицированных с x переменных. Используется, например, при идентификации связывающих приставок кванторов и описателей, обозначенных одной переменной.

8. (выч x A). x есть теоремная переменная; A - программное выражение для идентифицированного с x числа, представленного в формате "с плавающей запятой".

9. (пустоеслово x). Элемент указывает, что теоремная переменная x может определяться пустым набором операндов той ассоциативно-коммутативной операции с единицей, под которой она расположена в идентифицируемой части теоремы.

10. (неизвестная x). Элемент указывает, что теоремная переменная x должна идентифицироваться с неизвестной текущей задачи.

11. (известно x). Теоремная переменная x должна идентифицироваться с выражением, не содержащим неизвестных текущей задачи.

12. (неизвестные x). Теоремная переменная x должна идентифицироваться с выражением, содержащим хотя бы одну неизвестную текущей задачи.

13. (параметры A). A есть список теоремных переменных, для которых идентификация не осуществляется. Такая ситуация может складываться для переменных, не используемых в создаваемых приемом новых термах, либо при обобщении компилятором теоремы приема, после которого исходные переменные пропадают. Например, вместо двух переменных под исходной ассоциативно-коммутативной операцией после обобщения возникает набор из произвольного числа переменных.

14. (пересечение x A). Теоремная переменная x идентифицируется только из рассмотрения подтермов теоремы, вхождения которых образуют набор A.

15. (содержится x A). A есть список таких переменных, идентифицируемых под квантором либо описателем по теоремной переменной x, для которых компилятор не создает операторы, проверяющие их независимость от x.

16. (результподст x A). Теоремная переменная x идентифицирована как набор термов, используемых в функциональных конструкциях в качестве набора операндов. A - программное выражение для этого набора термов.

17. (свобоперанд x y). Терм, идентифицированный с теоремной переменной x, не содержит свободных вхождений переменной, идентифицированной с теоремной переменной y.

18. (связка A). A - программное выражение, определяющее одну из переменных связывающей приставки идентифицируемого квантора либо описателя.

19.9.2 Идентификация вхождений

Перечислим основные типы элементов, определяющих идентификацию вхождений в теорему приема либо несущих информацию, используемую при идентификации вхождений.

1. (операнд A B). A - вхождение в теорему; B - программное выражение для вхождения, идентифицированного с A. Заметим, что, вообще говоря, такой элемент может быть введен не для всех вхождений в теорему. Это происходит из-за того, что иногда вхождение идентифицируется с подтермом, полученным некоторыми простыми преобразованиями (например, группировкой) рассматриваемого терма задачи.

2. (набороперандов A B). A - вхождение в теорему терма $f(\ldots)$, где f - ассоциативный и коммутативный логический символ. B - программное выражение, значением которого является набор термов, перечисляющий еще не идентифицированные операнды той операции, которая идентифицируется с A.

3. (базавхождения A B). A - вхождение в теорему; B - программное выражение для подтерма, идентифицированного с A. В отличие от элемента (операнд ...), этот элемент используется крайне редко.

4. (ассоциативно A B). A - вхождение в теорему терма $f(\ldots)$, где f - ассоциативный, но не коммутативный символ. B - программное выражение, значением которого является вхождение последнего идентифицированного операнда операции, идентифицируемой с A.

5. (Выч v A). v - вхождение операнда, для которого найдено программное выражение A, определяющее его численное значение в формате с плавающей запятой.

6. (длялюбого A). Разрешается идентифицировать конъюнкцию либо дизъюнкцию, корневое вхождение которой в теорему есть A, с подмножеством антецедентов кванторной импликации.

7. (подтерм A). A - корневое вхождение в теорему одной из подлежащих идентификации частей, отличной от той части, в которой находится точка привязки.

8. (обрыв x). x - программная переменная, имеющая своим значением корневое вхождение вспомогательного подтерма, введенного в процессе идентификации.

9. (количествоопе рандов A). A - вхождение в теорему. Элемент указывает на то, что в программу уже введен оператор, проверяющий совпадение чисел операндов операции A и операции, идентифицированной с A.

10. (смравно A). A - набор вхождений в теорему, выделенных указателем "смравно(...)". Эти вхождения идентифицируются либо непосредственно, либо с переходом через имеющиеся в контексте идентификации тождества для идентифицирующего подтерма.

19.9.3 Учет посылок, используемых приемом

Обычно процедуры, выполняющие преобразование приема, получают в числе своих входных данных список использованных приемом посылок. Он составляется как объединение подсписков, определяемых по мере продвижения от начала программы приема к концу. Для сохранения в процессе компиляции ссылок на программные выражения, значениями которых служат эти подсписки, вводятся информационные элементы различных типов. Перечислим основные типы таких элементов, а также некоторых других элементов, связанных со списком утверждений, образующих контекст применения приема (эти утверждения называем посылками приема).

1. (выводимо x). x - программная переменная либо программное выражение, значением которого становится подсписок использованных посылок. Подсписки, определяемые такими программными выражениями, могут пересекаться.

2. (посылка x). x - программная переменная, значением которой служит либо 0, либо некоторая использованная приемом посылка.

3. (проверка x). x - программная переменная, значением которой служит пара (выводимо A), где A - подсписок использованных посылок.

4. (списокпосылок x). x - программная переменная либо программное выражение для всего списка посылок, определяющего контекст применения приема.

5. (коррекцияпосылок x). x - программная переменная для комментария (коррекцияпосылок A) к использованному приемом нормализатору. Такие комментарии указывают выведенные из посылок нормализатора утверждения, которые могут понадобиться для пополнения контекста срабатывания приема (например, для сопровождения по о.д.з. новых выражений, возникших после применения нормализатора). Набор A состоит из троек (B_1 B_2 B_3), где B_3 - посылки, из которых вытекает утверждение B_1; B_2 - либо 0, либо некоторое подмножество списка B_3, являющееся следствием утверждения B_1 и остатка утверждений списка B_3.

6. (посылки A B). B есть набор дополнительных посылок, используемых при обработке нормализаторами теоремного терма A.

7. (чисткапосылок A x B). B есть фильтр, отбирающий утверждения, исключаемые из списка посылок при обработке нормализаторами теоремного терма A. x - переменная, обозначающая в фильтре B исключаемое утверждение.

8. (занесениепосылки v A). v - вхождение антецедента теоремы приема, при обработке которого вводится дополнительная посылка, определяемая теоремным термом A.

9. (удалениепосылок v x A). v - вхождение антецедента, при обработке которого происходит удаление посылок, отбираемых фильтром A. В этом фильтре удаляемая посылка обозначается переменной x.

19.9.4 Списки переменных

Перечислим информационные элементы, указывающие различные используемые в процессе компиляции списки переменных.

1. (новаяпеременная A). A - программное выражение, значением которого служит набор переменных, использованных на текущий момент в контексте срабатывания приема. Это выражение используется при выборе новых переменных создаваемых приемом термов.

2. (переменные A B). A - вхождение в теорему квантора либо описателя; B - программное выражение для связывающей приставки этого квантора либо описателя.

3. (квантор A). A - программное выражение для набора переменных, идентифицированного с некоторой переменной из связывающей приставки идентифицируемого квантора либо описателя.

4. (списокпеременных A). A есть список теоремных переменных, используемых на текущий момент компиляции; в этот список включаются также переменные фильтров и указателей приема. Элемент нужен для выбора новых переменных во вспомогательных логических конструкциях, вводимых при компиляции. В основном, это имеет место при обработке сложных программно реализуемых антецедентов.

5. (свобвхождение x A). Теоремная переменная x идентифицируется с термом, не имеющим свободных вхождений переменных из набора переменных, определяемого программным выражением A.

19.9.5 Антецеденты и указатели дополнительной идентификации

Перечисляются информационные элементы, связанны с обработкой антецедентов и указателей дополнительной идентификации. Многие из них дублируют соответствующие указатели приема - это делается для того, чтобы вместо косвенной ссылки на антецедент через указатель его вхождения получить непосредственную ссылку на само вхождение.

1. (антецедент A B). A - вхождение в теорему непосредственно идентифицируемого антецедента. B - сначала 0, а затем программное выражение для утверждения из контекста срабатывания приема, идентифицированного с этим антецедентом.

2. (блокпроверок A). A - набор вхождений антецедентов теоремы, для которых предусмотрено обращение к проверочному оператору.

3. (мнимаячасть A). A - набор вхождений антецедентов теоремы, для которых предусмотрено обращение к усиленному проверочному оператору. Название элемента ранее было другим, однако изменилось из-за переименования логического символа, который в некоторый момент развития системы оказался неиспользуемым по своему прямому назначению.

4. (вхождениевзадачу A B C). A - вхождение в теорему; C - программная переменная либо программное выражение, указывающие на идентификацию A с подтермом посылки текущей задачи (значение C равно 0) либо с подтермом условия (это значение равно 1). B - программное выражение либо программная переменная, указывающие вхождение рассматриваемой посылки либо условия в список посылок (условий).

5. (позиция x A B C). A - программное выражение, определяющее задачу. C - программное выражение, определяющее вхождение идентифицированной с теоремной переменной x посылки при $B = 0$ либо идентифицированного с x условия при $B = 1$ в соответствующий список задачи A.

6. (идентификатор A). A есть вхождение антецедента теоремы, выделенного указателем "идентификатор(...)".

7. (начало A B). A есть корневое вхождение антецедента, обработка которого начинается сразу же, как только определяются значения теоремных переменных списка B.

8. (конец A). A есть список вхождений антецедентов, компилируемых после обработки идентифицирующих антецедентов.

9. (копия A). A есть список термов, используемых внутри указателя дополнительной идентификации "контекст(...)" в конструкциях "вид(... A)", "подтерм(A)".

10. (примечпосылки v). v есть вхождение антецедента, идентифицируемого с посылкой задачи.

11. (замечусловие v). v есть вхождение антецедента, идентифицируемого с условием задачи.

12. (спуск A). При компиляции приема проверочного оператора после обработки антецедентов, корневые вхождения которых перечислены в списке A, вводится дополнительный фрагмент программы, обрывающий при откате процесс применения этого оператора.

13. (откат v). v есть вхождение антецедента, после обработки которого проверочным оператором располагается ветвь программы, инициирующая обрыв внешнего перечисления при откате.

14. (сброс A). A есть вхождение антецедента теоремы приема пакетного оператора, такое, что при неустановлении его истинности происходит выход из оператора по значению "ложь".

15. (входвтерм v). v есть вхождение антецедента, выделенного указателем приема "теквхожд(...)".

16. (повтор v A). Непосредственно идентифицируемый антецедент теоремы приема проверочного оператора, вхождение которого есть v, идентифицируется без учета комментариев (исключение ...), блокирующих повторное использование антецедентов в цепочках рекурсивных обращений. A - сначала 0, а затем программная переменная для утверждения, идентифицированного с v.

17. (смпосылка A_1 A_2). A_1 есть вхождение антецедента, непосредственная идентификация которого не начинается до тех пор, пока не идентифицированы переменные списка A_2.

18. (транзитоперанд A). A - вхождение антецедента, имеющего вид $E(a\ b)$, где E - отношение эквивалентности, для идентификации операндов которого используется оператор "транзитоперанд".

19. (дизъюнктчлен A_1 A_2 A_3 A_4). A_1 - набор вхождений антецедентов, выделенных указателем "дизъюнктчлен(...)". A_2 - сначала 0, а затем программная переменная для накопителя дизъюнктивных членов заменяющего терма (следующая после нее программная переменная используется в качестве накопителя информационных элементов оператора "заменавхождения"). A_3 - сначала 0, а затем метка отката при обрыве идентификации. A_4 - сначала 0, а затем набор меток отката для продолжения перечислений идентифицируемых дизъюнктивных членов; длина его равна длине набора A_1.

19.9.6 Идентификация функциональных переменных

Функциональные переменные $f(x_1 \dots x_n)$, встречающиеся в теореме приема, обычно выделяются указателями "отображение(f)" и используются для идентификации некоторого терма A, содержащего переменные $y_1, \dots, y_n$, идентифицированные с $x_1, \dots, x_n$. Далее A начинает играть роль шаблона, по которому определяются выражения $f(t_1 \dots t_n)$ с новыми термами $t_1, \dots t_n$. Допускаются также некоторые другие типы функциональных переменных (например, при идентификации многочленов). Перечислим основные типы информационных элементов программного блока, связанные с функциональными переменными.

1. (отображение A B). B есть вхождение в теорему приема выражения "значение(A x)", выделенного указателем "отображение(...)".

2. (функция f A B C). f есть теоремная переменная для функции; A - набор теоремных переменных - аргументов функции, либо программная переменная, значением которой служит этот набор; B - либо 0, либо программное выражение для утверждения, задающего область определения f. Для C имеется одна из следующих возможностей:

   а) $C = 0$;

   б) C - пара (вхождение D), где D - программное выражение, определяющее вхождение выражения для значения функции f;

   в) C - пара (терм D), где D - программное выражение, определяющее выражение для значения функции f;

   г) C - тройка (функподст D_1 D_2), где D_1 - программное выражение для набора переменных либо нулей, D_2 - программное выражение для терма, причем выражение для значения функции f определяется операторным выражением "функподст(D_2 A D_1 T)". Этот случай встречается крайне редко.

   Заметим, что практически всегда B равно 0, так как при идентификации функциональной переменной $f(x)$ область определения функции f не фиксируется. Однако, есть одно исключение - определение функциональной переменной f из равенства f = отображение(...). Здесь область определения может быть найдена, и для нее вводится программное выражение B.

3. (заменатермов f A_1 A_2 A_3). Элемент вводится при наличии указателя приема "заменатермов(...)", выделяющего подтерм "значение(f A)". A_1 - вхождение в теорему приема данного подтерма. A_2, A_3 - сначала нули; затем A_2 - программное выражение для терма, идентифицированного с A_1, A_3 - программное выражение для набора вхождений в него терма, идентифицированного с A.

4. (смногочлен x A). Теоремная переменная x идентифицируется с многочленом. A - сначала 0, а затем программное выражение для набора коэффициентов многочлена, представленных в формате термов и расположенных по возрастанию степеней. В этот набор включены нулевые коэффициенты.

5. (видмногочлена v). Выражение "суммавсех(...)" по вхождению v идентифицируется как многочлен.

6. (вхождениетерма f A). f - функциональная переменная, выделенная указателем приема "вхождение(f)". A - программное выражение для вхождения подтерма, идентифицированного с теоремным выражением "значение(f B)".

7. (новоперанд f A). A есть набор программных выражений для вхождений операндов терма, идентифицированного с теоремным выражением $f(A_1, \ldots, A_n)$, выделенного указателем "симметрично". Элемент используется для упорядочения операндов новых термов вида $f(B_1 \ldots B_n)$, создаваемых приемом. Это упорядочение согласуется с исходным упорядочением для $A_1, \ldots A_n$.

8. (функподст f A_1 A_2). Переменная f, выделенная указателем "функподст(f g)", идентифицирована с термом, определяемым программным выражением A_1, причем программное выражение A_2 определяет набор вхождений в данный терм термов "значение(g ...)".

9. (Набор f). Функциональная переменная f идентифицируется с набором функций.

19.9.7 Идентификация наборов

Перечисляем основные информационные элементы программного блока, используемые при идентификации конечных наборов термов.

1. (набор A B). A есть вхождение в заменяемый терм теоремы приема ассоциативно-коммутативной операции, расформируемой в набор термов согласно указателю приема "набор(первыйтерм)". B - сначала 0, а затем программное выражение для вхождения A.

2. (перечень A B C D). A есть вхождение в заменяющий терм ассоциативно-коммутативной операции, являющейся результатом группировки при выполнении приема с указателем "набор(второйтерм)". B - вхождение подтерма заменяемого терма, используемого в качестве шаблона при идентификации группируемых термов; C - переменная из B, варьируемая при группировке; D - сначала 0, а затем программное выражение для терма, идентифицированного с A.

3. (список A). A - список вхождений в теорему, определяющих наборы, элементы которых идентифицируются без учета порядка.

4. (развертка A). A - набор подтермов теоремы приема, выделенных указателями приема "развертка(...)". Эти подтермы идентифицируются либо создаются с переходом от функциональных конструкций к конечным наборам (например, вместо "сигмы" рассматривается обычная сумма).

5. (пусто A). A - выделенное указателем "развертка(...)" вхождение в теорему приема подтерма, который может идентифицироваться с пустым списком операндов.

6. (элементыматрицы f A m n). f - функциональная переменная, идентифицируемая по указателю приема "матрица(...)". A,m,n - программные выражения, соответственно, для набора наборов (строк) элементов матрицы, заданных термами; для числа строк и числа столбцов матрицы, также заданных термами.

7. (матрица A). A - вхождение в теорему приема выражения "отображение(...)", идентифицируемого либо формируемого как прямоугольная матрица.

8. (модифсборка v w). v - вхождение в теорему терма "набор(...)", при формировании которого предпринимается такая же перестановка операндов, как при идентификации задающего набор терма по вхождению v.

9. (вектор x A). x - функциональная теоремная переменная, идентифицируемая с помощью кванторной импликации в антецеденте как набор термов. A - программная переменная для накопителя этих термов.

10. (длинанабора x A). x - функциональная теоремная переменная, идентифицируемая как набор термов. A - программное выражение для длины этого набора, представленной в формате терма.

11. (функции x A). x - функциональная теоремная переменная, идентифицируемая с помощью кванторной импликации в антецеденте как набор функциональных переменных. A - программная переменная для набора пар (набор аргументов функциональной переменной - выражение для значения функциональной переменной).

19.9.8 Вычисления

При идентификации теоремных переменных им могут сопоставляться в качестве значений не логические структуры данных (термы), а используемые в реализуемых приемом вычислениях технические структуры данных. Для такой идентификации, ориентированной на вычисления, предусмотрены специальные информационные элементы программного блока. Ограничимся здесь указанием основного типа таких элементов. Это элемент (транслвыражения x A t). Значение теоремной переменной x в формате t вычислительных данных ГЕНОЛОГа определяется программным выражением A. Типы используемых для вычислений форматов перечислены в справочной информации логического символа "тип".

19.9.9 Особенности идентификации

Для указателей приема, определяющих особенности идентификации, обычно вводятся двойники в информационных элементах программного блока. Они имеют непосредственные ссылки на вхождения в теорему приема (вместо указателей вхождений "фикс(...)") и сохраняют программные выражения, определяющие идентификацию.

1. (единица A B C). Элемент вводится по указателю приема "единица(...)". A - вхождение в теорему некоторой операции f; B - вхождение операнда этой операции. При идентификации учитывается возможность равенства данного операнда логическому символу C, являющемуся единицей операции f.

2. (отрицание A). Элемент вводится по указателю "отрицание(...)". A есть вхождение в теорему выражения $f(x)$, такого, что для одноместной операции f в ее области определения выполняется тождественно $f(f(x)) = x$. При идентификации $f(x)$ с термом B в качестве x может быть взято $f(B)$.

3. (заголовок A). Элемент указывает на то, что при идентификации подтерма, расположенного по вхождению A в теорему приема, заголовок идентифицирующего терма не обязан совпадать с заголовком идентифицируемого. Этот элемент вводится в самых различных ситуациях - при наличии операнда двуместной операции, который может обратиться в единицу; при идентификации с указателем "отрицание", и т.д. При компиляции он блокирует ввод оператора, проверяющего совпадение заголовков идентифицируемого и идентифицирующего термов.

4. (заменазнака A B x D C). Элемент вводится по указателю приема "заменазнака(...)". A есть вхождение в теорему терма $f(\ldots x \ldots)$. Если он идентифицируется с термом вида $B(f(\ldots t \ldots))$, то последний представляется как $f(\ldots D(t) \ldots)$. C - сначала 0, а затем программное выражение для вхождения идентифицирующего терма - $B(f(\ldots t \ldots))$ либо $f(\ldots t \ldots)$.

5. (пересечениесписков A). A есть вхождение в теорему подтерма, при идентификации с которым будет использоваться справочник "пересечениесписков". При этом заголовки идентифицирующего и идентифицируемого термов могут оказаться различными.

6. (знаксуммы A_1 A_2 A_3). Элемент вводится по указателю приема "знаксуммы(A_1 B)". A_2 - вхождение в теорему приема, определяемое по указателю вхождения B. A_3 - сначала 0, а затем программная переменная, являющаяся индикатором замены знака A_1 у группы операндов по данному вхождению. Этот индикатор принимает значение 0, если знак не изменен, и значение 1, если изменен.

7. (нормзнака A B). A есть вхождение в теорему операнда $C(x)$ ассоциативно-коммутативной операции D, идентифицируемого как набор D - операндов терма, идентифицированного с x, у которых "знак" C изменен на противоположный. B - сначала 0, а затем программное выражение для указанного набора операндов с измененным знаком.

8. (логарифмы A B). Элемент вводится по указателю приема "группировка(x)". A - вхождение в идентифицируемую часть теоремы приема терма $f(x\ C)$. B - указатель этапов обработки указателя в процессе компиляции, принимающий последовательно значения 0,1 и 2. Заголовок элемента ранее был другим, но впоследствии изменился из-за переименования логического символа.

9. (группировки A x B). Элемент вводится по указателю приема "группировки(x C)". A есть вхождение в заменяемый терм выражения $f(x\ y)$, определяемое указателем. B - сначала 0, а затем программное выражение для набора участвовавших в идентификации $f(x\ y)$ термов.

10. (подстановка A x B). Элемент вводится по указателю прием "подстановка(...)". A есть вхождение в заменяемый терм операнда ассоциативно-коммутативной операции, содержащего переменную x. При идентификации допускается отсутствие такого операнда, и тогда x идентифицируется с однобуквенным термом, состоящим из логического символа B.

11. (свертка x A B C). Элемент вводится по указателю приема "свертка(...)". Теоремная переменная x идентифицируется с термом, определяемым программным выражением A, определяемым в процессе компиляции (вначале A равно 0), и порождает вспомогательное обозначение согласно указателю. Это обозначение представляет собой переменную, являющуюся значением программной переменной C (вначале C равно 0). B - утверждение, определяющее дополнительные посылки при вводе вспомогательного обозначения.

12. (контроль x A). Вхождения переменной x в идентифицируемую часть теоремы приема расположены только внутри подтермов, выделенных указателями приема "подстановка(...)". $A = (A_1, \ldots, A_n)$ - набор вхождений первых символов этих подтермов. При идентификации очередного A_i учитывается возможность того, что x еще не определена, т.е. соответствующая программная переменная для терма, идентифицируемого с x в цикле просмотра вхождений A_i, равна 0. По окончании этого цикла проверяется, что x идентифицирована с некоторым термом.

13. (общаястепень A_1 A_2 A_3 A_4 A_5). Элемент вводится по указателю "общаястепень(...)". A_1 есть заголовок вспомогательного оператора, используемого для выделения общего подтерма из заданного набора термов; A_2 - теоремная переменная, идентифицируемая с выделенным общим подтермом; A_3 - набор переменных, идентифицируемых с остатками термов исходного набора; A_4 - набор вхождений в теорему, соответствующих термам исходного набора. A_5 - сначала набор из нулей, имеющий ту же длину, что A_4, затем - набор программных выражений для термов исходного набора.

14. (учетзнака A). A есть заменяющий терм для приемов, имеющих указатель "знаксуммы(...)".

15. (второйоперанд A x). A есть вхождение в теорему такой ассоциативно-коммутативной операции, что теоремная переменная x либо функциональная переменная "значение(x B)" идентифицируется как ее непосредственный операнд.

16. (альтернатива A B). По вхождениям в теорему, перечисленным в списке A, могут находиться (обязательно сразу по всем этим вхождениям) альтернативные логические символы, перечисленные в наборе B.

17. (дробь A_1 A_2 A_3 A_4). Элемент вводится по уазателю "дробь(...)". A_1 - набор вхождений двуместных некоммутативных операций либо отношений, при идентификации либо формировании которых допустима (одновременная для всех вхождений) перестановка операндов. A_2, A_3 - сначала нули, а затем программные выражения для вхождения, идентифицированного с одной из этих операций и вхождения, идентифицированного с ее первым операндом. A_4 - набор теоремных термов, при формировании которых предпринимается синхронная с A_1 перестановка операндов их корневой двуместной операции.

18. (схемаоперандов v A_1 A_2). v - вхождение в теорему символа операции, допускающей частичные перестановки своих операндов. A_1 - древовидная структура, определенная операторным выражением "схемаоперандов(...)" для возможных перестановок операндов данной операции. A_2 - набор троек (1 B_1 B_2), у которых B_1 - набор из дерева A_1 либо вхождение операнда операции v; B_2 - программное выражение для B_1. A_2 указывает на уже проведенную идентификацию операндов операции v.

19. (вычеркивание A). При идентификации термов, перечисленных в списке A, не используется справочник "пересечениесписков".

20. (первыйтерм A). A есть подтерм, идентифицируемый, как первый операнд внешней ассоциативно-коммутативной операции.

21. (внешнийквантор v). v - вхождение квантора общности либо существования, идентифицируемого без учета возможного перехода к отрицанию.

22. (обобщподст v). v - вхождение квантора, для которого при идентификации не предпринимается проверка невхождения переменных связывающей приставки в термы, идентифицированные с переменными, не входящими в эту приставку.

23. (коммутативно v). v - вхождение коммутативной неассоциативной операции, которая при идентификации рассматривается как некоммутативная.

24. (циклупорядочение v A). При идентификации операции по вхождению v допускаются перестановки операндов. Если $A = 0$, то такие перестановки - произвольные циклические; если $A = 1$, то допускается лишь одна циклическая перестановка, при которой перечисление начинается со второго операндов; если $A = 2$, то допускается не циклическая перестановка операндов, а изменение их порядка на противоположный. Если $A = 3$, то допускаются произвольные перестановки операндов. Если $A = 4$, то допускаются произвольные циклические перестановки и изменение порядка операндов на противоположный.

25. (подобны v x). Сопровождает элемент (циклупорядочение v 4), указывая программную переменную x - индикатор прямого ($x = 0$) либо обратного ($x = 1$) направления перечисления операндов.

26. (вид v). При идентификации подтерма по вхождению v блокируется применение справочника "вид".

27. (делители v). При идентификации операции по вхождению v используется справочник "делители".

28. (замены v A). Подтерм по вхождению v может иметь при идентификации любой из альтернативных заголовков списка A.

29. (коэффициент v x t). Слагаемое по вхождению v идентифицируется как сумма всех членов, представимых в виде произведения на выражение, определяемое теоремным термом t (без перемножения степеней с одинаковым основанием). Теоремная переменная x при этом идентифицируется как сумма соответствующих коэффициентов при t.

30. (сдвиг v A t X). При идентификации подтерма по вхождению v допускается циклическая перестановка его корневых операндов, символьные номера которых образуют список A. t - терм, при формировании которого осуществляется синхронная перестановка операндов. X - сначала 0, а затем программная переменная, значением которой является величина циклического сдвига.

31. (перегруппировка A). A - набор вхождений в теорему двуместных отношений, идентификация которых происходит с группировкой всех "ненулевых" членов в одной части.

32. (Отрицание v). v есть вхождение эквивалентности, которая должна идентифицироваться с возможным переходом к отрицаниям своих частей.

19.9.10 Нормализаторы приема

Нормализаторы приема, определяемые термами "быстрпреобр(...)" из описания приема, в процессе компиляции порождают информационные элементы программного блока (быстрпреобр A_1 A_2 A_3 A_4 A_5). Здесь A_1 - терм, подлежащий обработке нормализаторами (т.е. пакетными операторами либо вспомогательными задачами). Заголовки применяемых к A_1 операторов либо термы "задача(...)", определяющие вспомогательные задачи, собраны в набор A_2. Они расположены в нем с сохранением порядка обработки. A_3 - набор той же длины, что и A_2. Его элементы суть наборы программных выражений, определяющих остальные входные данные пакетных нормализаторов, следующие за обрабатываемым термом (т.е. список посылок и список

комментариев, либо только список комментариев, либо пустой набор - в зависимости от формата оператора). A_4 - набор той же длины, что и два предыдущих. Его элементы - сопровождающие обращение к нормализатору термы "условие(...)", "вариант(...)", "лимит(...)". A_5 сначала равно 0, в процессе компиляции становится равно 1, а затем заменяется на программное выражение для результата нормализации.

Информационный элемент (быстрпреобр ...) создается лишь после того, как компиляция определяет всю необходимую для получения его программных выражений информацию. Создание этих элементов осуществляется процедурой "учетнормализаторов(...)", обращения к которой расставлены в различных точках компилятора. При очередном обращении дополнительно вводятся те элементы (быстрпреобр ...) для которых информация уже есть.

19.9.11 Комментарии, изменяемые приемом

Для работы с комментариями компилятор использует следующие информационные элементы:

1. (коммент A). Уже учтен указатель приема A, связанный с обработкой комментариев.

2. (примечание A). x есть программная переменная для накопителя комментариев к формируемому либо изменяемому утверждению. Этот накопитель представляет собой одноэлементный набор, состоящий из комментариев.

3. (комментарий A B x). A есть идентифицирующий терм "комментарий(...)" либо "комментарийпосылки(...)"; x - программная переменная, идентифицируемая с соответствующим комментарием. B - в случае комментария к посылке либо условию есть пара (программное выражение для вхождения данной посылки либо условия - программное выражение для указателя на посылку (0) либо на условие (1)). В случае комментария либо комментария посылок B представляет собой указатель, равный 0 для комментария к посылкам, 1 - для комментария к задаче, и 2, если тип комментария определяется типом задачи: для задачи на исследование - комментарий к посылкам, иначе - комментарий к задаче.

4. (замечание x). x есть программная переменная, определяющая список комментариев пакетного оператора, к которому относится компилируемый прием.

5. (буфер A x). x есть программная переменная, используемая в качестве накопителя элементов, заносимых приемом нормализатора перед выполнением замены в набор B комментария (A ... B).

6. (внутрвывод x). x есть программная переменная для вспомогательного списка комментариев, используемого при обращении к нормализаторам из приема пакетного анализатора.

19.9.12 Дополнительные преобразования

Для учета действий с посылками и условиями задачи, дополняющих основное преобразование приема, используются следующие информационные элементы:

1. (удалениеусловия x). x есть программная переменная, значением которой служит набор - накопитель удаляемых условий задачи на описание.

2. (удалениепосылки x). x есть программная переменная, значением которой является набор - накопитель удаляемых посылок задачи.

3. (учетвывода x). x есть программная переменная, значением которой является набор пар (A_1 A_2) входных данных оператора "вывод" либо "выводусловия", используемого при сопровождении преобразований: A_1 - добавляемое утверждение; A_2 - набор сопутствующих данных.

4. (ключвывода x). x есть программная переменная, значением которой служит набор (вывод A_1 A_2 A_3), заносимый в список сопутствующих данных применяемого преобразователя для вывода дополнительного утверждения A_1.

19.9.13 Информация о задачах

Для сохранения информации о рассматриваемых в контексте применения приема задачах используются следующие информационные элементы:

1. (текущаязадача A). A есть программное выражение для текущей задачи.

2. (текущийуровень A). A есть программное выражение для текущего уровня текущей задачи.

3. (исходнаязадача A). A есть программное выражение для исходной задачи.

4. (надзадача A_1 A_2 A_3 A_4). A_1, A_2, A_3, A_4 суть программные выражения, определяющие внешнюю задачу A_1 и координату (A_2, A_3, A_4) вхождения в нее преобразуемого терма. При этом A_2 - однобуквенное выражение, состоящее из переменной.

5. (вход A). Текущая задача - на исследование. A есть программное выражение для внешней задачи на описание.

19.9.14 Разное

В заключение приведем еще ряд информационных элементов, не попавших в предыдущие разделы.

1. (определено A x). Идентификация подтерма A не должна начинаться до тех пор, пока не идентифицирована переменная x. Элемент определяется указателями приема, уточняющими последовательность идентификации. Такое уточнение может быть необходимо для ускорения отсечения при попытке применения приема.

2. (видпеременной x). Обработка выделенных указателем "идентификатор(...)" антецедентов - равенств с переменной x в одной из своих частей - не начинается до тех пор, пока не идентифицировано значение переменной x.

3. (прогртерм A B). B есть программное выражение для создаваемого приемом теоремного терма A. Элемент бывает полезен, если терм A имеет несколько вхождений в теорему приема.

4. (блокпрограммы n A). n - символьный номер ссылки на фрагмент программы A, который пока не зарегистрирован в основном списке фрагментов программы программного блока. Регистрация этого фрагмента происходит непосредственно перед обращением к редактированием программы. Она выполняется процедурой "вставкафрагментов". Ссылка на дополнительный фрагмент, сохраненный в информационном элементе (блокпрограммы n A), имеет вид размещенного в другом фрагменте псевдооператора "1(P n)" либо "2(n)". Первый из них заменяется процедурой "вставкафрагментов" на оператор "P иначе N"; второй - на "ветвь N".

19.10 Создание идентифицирующей части программы

Основную часть работы по созданию программы приема выполняет процедура "идентификатор", реализующая идентифицирующую часть программы. В этой части определяются объекты текущего контекста срабатывания приема, соответствующие теоремным переменным. Исключение составляют новые переменные, выбираемые приемом при срабатывании. Здесь же происходит и обработка антецедентов теоремы приема. Обращения к процедуре "идентификатор" имеются не только из рассмотренных выше процедур, реализующих общие схемы компиляции приемов различных типов, но и из многих вспомогательных процедур, и даже из самой процедуры "идентификатор". Последнее объясняется необходимостью обработки внутренних логических контекстов, возникающих, например, в циклах просмотра различных списков термов.

Процедура "идентификатор" по своему объему значительно превосходит все остальные процедуры компилятора ГЕНОЛОГа. Обращение к ней имеет вид "идентификатор(x1 x2 x3 x4 x5 x6)". Здесь x1 - корневое вхождение в теорему идентифицируемого подтерма (если таких подтермов несколько, то берется подтерм, содержащий точку привязки); x2 - набор установок на идентификацию; x3 - программный блок; x4 - заголовок приема; x5 - описание приема; x6 - дополнительная информация (обычно список x6 пуст). В описание приема входят: терм "условие(и($A_1 \ldots A_n$))", перечисляющий его фильтры $A_1, \ldots, A_n$, указатели приема, а также нормализаторы приема - термы "быстрпреобр($t \ldots$)", где t - явно заданный нормализуемый подтерм. Если идентифицирующая часть программы успешно синтезирована, то выход из оператора происходит по значению "истина", иначе оператор ложен.

При описании преобразований, выполняемых процедурой "идентификатор", мы часто будем использовать термины "идентифицируемый терм" и "идентифицирующий терм". Первый представляет собой некоторый подтерм теоремы приема; второй - сопоставляемый ему программой приема терм, встречающийся в условии либо посылке задачи, в преобразуемом терме нормализатора, в посылке пакетного оператора, и т.п.

19.10.1 Предварительные преобразования

Переходим к рассмотрению программы оператора "идентификатор". Начальная точка этой программы может быть найдена либо переходом из главного меню к корневому фрагменту программы символа "идентификатор", либо из пункта оглавле-

ния программ ("Формирование идентифицирующей процедуры", "Начало идентификации ...").

При работе с программным блоком часто будут использоваться операторы "новоператор(x1 x2)", "прогинф(x1 x2)" и "прогинфы(x1 x2)". Первый добавляет к концу последнего фрагмента программы, зарегистрированному в программном блоке x1, новый оператор либо набор операторов x2. Второй - добавляет новый информационный элемент x2, последний - набор x2 таких элементов.

Прежде всего, вводится, если его еще не было, новый информационный элемент (левыйкрай v), сохраняющий ссылку на корневое вхождение идентифицируемого подтерма. Он нужен для того, чтобы можно было отличить первое, основное обращение к процедуре "идентификатор" от вспомогательных обращений к ней, реализуемых внутри основного обращения. Заметим, что иногда при вспомогательных обращениях переменная x1 может принимать значение 0.

Если имеется ровно одна установка на идентификацию, причем она имеет вид (операнд v x), где v - вхождение логического символа, не имеющее операндов и совпадающее с x1, то идентификация не нужна (проверка того, что по вхождению v расположен данный символ, уже проведена перед обращением к процедуре). Поэтому здесь сразу происходит выход по "истина". Если указанный случай не имеет места, то переход через "ветвь 2".

После контрольной точки "прием(3)" выполняется просмотр установок на идентификацию (корень v), которые указывают на корневые вхождения v непосредственно идентифицируемых антецедентов теоремы. Обычно к числу таких антецедентов при обращении к процедуре "идентификатор" добавлены также антецеденты, обрабатываемые проверочными операторами. Для них установки (корень v) заменяются на (извлекается v) либо, если используется усиленный проверочный оператор, на (мнимаячасть v). Соответственно, исключаются информационные элементы (антецедент v).

После контрольной точки "прием(4)" происходит сохранение в списке информационных элементов программного блока копий установок на идентификацию (идентификатор ...). Обработка соответствующих этим установкам антецедентов заключается в построении некоторых вспомогательных термов, преобразуемых пакетными операторами и вспомогательными задачами, и либо в сравнении полученных термов между собой, либо в идентификации их с заданными подтермами теоремы.

После оператора "входит(корень x5)" рассматривается крайне редкий случай приемов замены подтермов задачи на преобразование, имеющих указатель "корень". Заменяемая часть таких приемов представляет собой переменную (обычную либо функциональную), идентифицируемую с условием задачи. Предпринимается коррекция элемента (корень ...), который теперь будет указывать на корневое вхождение условия задачи; в случае функциональной переменной вводится либо доопределяется информационный элемент (функция ...), в который также заносится ссылка на условие задачи. Далее - переход через "ветвь 1".

Здесь инициализируется нулем переменная x7. Значение этой переменной является текущим уровнем, регулирующим срабатывание приемов компилятора при сканировании списка установок на идентификацию. Максимально допустимое значение текущего уровня равно 9; при превышении его происходит выход из процедуры "идентификатор" по значению "ложь". После каждого срабатывания приема значение переменной x7 изменяется на 0, и далее повторно просматривается все дерево приемов. Каких-либо весов, определяющих "теневые" зоны сканирования, в компиляторе не предусмотрено.

После контрольной точки "прием(6)" инициализируется список x8. Он будет состоять из тех вхождений в теорему приема, для которых идентификация уже проведена. Изначально в список включаются все вхождения v, упомянутые в установках на идентификацию (операнд v t). Они являются отправными точками, идентифицированными до обращения к процедуре "идентификатор". При инициализации x8 учитывается, что в редких случаях установки на идентификацию (операнд v t) могут определять вхождения не в теорему приема, а во вспомогательные термы "теквхожд(A)"; "внешсимвол(s A)". Тогда к списку x8 присоединяются не требующие специального рассмотрения вхождения корневых операндов s,A.

После контрольной точки "прием(7)" выполняется учет указателей приема "заменазнака(...)", относящихся к отправным точкам идентификации. Просматриваются информационные элементы (заменазнака A B x D C), у которых выделенное вхождение A имеется в определяющей отправную точку идентификации установке (операнд A t). Напомним, что t - программная переменная для вхождения v, идентифицированного с A. Для каждого такого элемента в программу приема добавляется оператор "альтернатива(...)", анализирующий ту операцию w, операндом которой служит v. Если w имеет заголовок B, то добавленный оператор присваивает первой свободной программной переменной X, определяемой как первый разряд набора x3, вхождение w, иначе - вхождение v. После этого переменная t заменяется в установке (операнд A t) на X. Такая же замена делается в информационном элементе (операнд A t). Смысл этих замен заключается в том, чтобы сопоставить теоремному вхождению A то вхождение в задачу, с которым оно фактически будет идентифицироваться, согласно указателю "заменазнака(...)". После данного цикла коррекций - переход через "ветвь 1".

Здесь, после контрольной точки "прием(8)", происходит учет указателей приема "начало(...)", выделяющих антецеденты, обработка которых должна происходить сразу же по идентификации заданных в указателе переменных. Создаются дублирующие информационные элементы (начало ...), в которых косвенные ссылки на антецеденты заменены явным указанием их вхождений. Далее - переход через "ветвь 1".

19.10.2 Начало основного цикла идентификации

После контрольной точки "прием(9)" располагается оператор "повторение", который является началом основного цикла процедуры "идентификатор". Откаты к этому оператору будут происходить после каждого срабатывания приема, обрабатывающего ту или иную установку на идентификацию. Цикл просмотра установок на идентификацию размещен несколько дальше; до перехода к нему после каждого отката будет выполняться текущий анализ сложившейся ситуации. Он необходим из-за того, что указатели приема могут предопределять вставку в программу приема некоторых дополнительных операторов сразу же, как только в процессе компиляции окажутся выполнены определенные условия.

Прежде всего, отметим здесь указатели "ключ(...)", "первыйключ(...)", "попытка(...)", "Попытка(...)", по которым должны вводиться операторы, играющие роль "защелки" - они проверяют отсутствие заданного комментария и после этого вводят его. Указатель "ключ" определяет ввод комментария к задаче сразу после проверки его отсутствия; указатель "первыйключ" - аналогичным образом поступает с комментарием к текущему терму задачи; указатели "попытка", "Попытка" суть модификации первых двух указателей, вводящие комментарии лишь при откате после

неудачной попытки применения приема.

После контрольной точки "прием(10)" начинается просмотр тех указателей x9 одного из выделенных выше типов, для которых отсутствует информационный элемент (коммент x9), означающий, что данный указатель уже был реализован в операторах программы. Определяется набор x10 корневых операндов указателя, задающих разряды комментария, следующие за его заголовком. Если набор x10 непуст и завершается термом t вида "посылки(...)", то этот терм не относится собственно к комментарию, а лишь вводит ограничение на момент учета указателя: реализующие его операторы размещаются в программе после того, как обработаны антецеденты, номера которых перечислены в терме t. Терм t исключается из набора x10 (последний будет использован при формировании комментария), и проверяется наличие установок на идентификацию, относящихся к антецедентам, перечисленным в t. Если хотя бы одна такая установка имеется, то обработка указателя преждевременна и обрывается. Иначе - откат к переходу через "ветвь 3" для продолжения обработки указателя.

Здесь инициализируется пустым словом накопитель x11 программных выражений, определяющих разряды комментария, следующие после его заголовка. Для заполнения этого накопителя последовательно просматриваются термы x12 набора x10, соответствующие разрядам комментария. Здесь рассматриваются два случая:

а) Терм x12 имеет вид "теквхожд" либо "вхождение(теквхожд)", т.е. ссылается на заменяемый подтерм либо на его вхождение. Тогда проверяется, что все вхождения в заменяемую часть теоремы приема уже зарегистрированы в наборе обработанных вхождений x8, а все ее переменные идентифицированы. После этого находится информационный элемент (корень X), у которого X - программная переменная для вхождения корня заменяемого подтерма, и по нему определяется регистрируемое в накопителе x11 программное выражение.

б) Терм x12 не имеет указанного выше вида. Тогда проверяется, что все его переменные уже идентифицрованы. После этого, в зависимости от того, имел ли x12 заголовок "вхождение" или нет, создается программное выражение для очередного разряда комментария, задающее вхождение терма либо сам терм. Если x12 имел заголовок "фикс", то он был введен только для задания списка переменных, которые должны быть идентифицированы к моменту обработки указателя. В этом случае новый разряд комментария по x12 не определяется.

Если при рассмотрении какого - либо терма x12 выявляется преждевременность обработки указателя, то продолжение обработки отменяется. Иначе, по завершении просмотра всех элементов набора x10, переход через "иначе 1".

В этом фрагменте просматриваются все программные выражения набора x11. Если какое-либо из них не является переменной, то для него вводится вспомогательная переменная, которой присваивается значение данного выражения. Она заменяет рассматриваемое выражение в списке x11. По окончании просмотра списка x11 - переход к следующему фрагменту.

Вводится информационный элемент (коммент x9), означающий, что указатель x9 обработан. После этого реализуется вставка в программу операторов, соответствующих определяемым указателем действиям. Сначала рассматривается случай приема, выполняемого при сканировании задачи. Для указателей "первыйключ" либо "ключ" вставляются два оператора: первый проверяет отсутствие данного комментария, второй - вводит этот комментарий. Указатель "первыйключ" определяет рассмотрение комментариев к текущему терму задачи, указатель "ключ" - к самой задаче (в случае задачи на исследование - к ее списку посылок). Для указателей "проверка",

"Проверка" в программе приема используются операторы "проверка(...)", "Проверка(...)", которые являются фиктивными перечисляющими операторами. При первом прохождении такой оператор проверяет отстутствие комментария, при откате - вводит этот комментарий. Аналогичным образом рассматривается случай приема пакетного оператора, хотя здесь остаются лишь две возможности - указатели "ключ" и "проверка".

После обработки указателей "ключ", "первыйключ", "попытка", "Попытка" - откат к переходу через оператор "ветвь 2", расположенный перед контрольной точкой "прием(10)". Здесь, после контрольной точки "прием(12)", проверяется, что уже обработаны основные установки на идентификацию - т.е. установки типов "контекст", "операнд", "идентификатор", а также (в случае приемов групповой замены) установки "корень". Если это имеет место, то выполняется обращение к процедуре "учетнормализаторов". Она подготавливает информационные элементы, необходимые для создания программных выражений, определяющих обрабатываемые нормализаторами новые термы приема. Далее - переход через "ветвь 1".

Здесь, после контрольной точки "прием(153)", предпринимается анализ указателя приема "лимит(N $A_1 \dots A_n$)". Если все переменные термов $A_1, \dots, A_n$ уже идентифицированы, то находится программное выражение x11 для ограничителя N числа шагов, отводимых на попытку применения приема начиная с данного момента, и в программу приема вставляется оператор "лимит(x11)".

Наконец, последний фрагмент рассматриваемой цепочки связан только с приемами вывода следствий в посылках задачи. Если применение такого приема предполагает обращения к нормализаторам, причем уже определились переменные, при которых становится возможно хотя бы одно из этих обращений, то вводятся вспомогательные накопители (выводимо пустоеслово) и (коррекцияпосылок пустоеслово), которые будут передаваться нормализаторам в списках комментариев. По завершении преобразования из этих накопителей будут извлекаться списки использованных нормализаторами посылок и списки дополнительно вводимых приемом посылок, необходимых для сопровождения по о.д.з.

Далее - откат к переходу через "ветвь 1", расположенному непосредственно после оператора "повторение".

19.10.3 Цикл просмотра установок на идентификацию

После контрольной точки "прием(13)" - цикл просмотра установок на идентификацию x9. Это - основной цикл работы процедуры "идентификатор", а вместе с тем и всего компилятора. В нем происходит анализ текущей установки x9, завершающийся применением одного из многочисленных приемов обработки этих установок. Фактически, здесь расположен корень очень большой древовидной программы, распадающейся на независимые приемы. После применения каждого из них происходит откат к упомянутому выше оператору "повторение", и далее - снова выход в цикл просмотра установок. Описание приемов обработки установок разобьем на подразделы, связанные с различными типами установок. Соответствующие начальные точки программы компилятора находятся через пункт ("Компилятор ГЕНОЛОГа", "Формирование идентифицирующей процедуры", "Типы установок на идентификацию") оглавления программ.

19.10.4 Установка на идентификацию "операнд"

Это - наиболее часто встречающийся тип установок, и для него возникло наибольшее число приемов. Началом обработки установки служит контрольная точка "прием(14)". Напомним, что установка имеет вид (операнд v X), где v - идентифицируемое вхождение в теорему; X - программная переменная для вхождения терма задачи (пакетного оператора, и т.п.), сопоставляемого вхождению v. Будем называть такое вхождение идентифицирующим.

Прежде всего, переменной x10 присваивается вхождение v. Логический символ по этому вхождению к данному моменту уже был идентифицирован, и речь идет лишь об идентификации, во-первых, операндов этого вхождения, и, во-вторых, о переходе к надтерму, операндом которого служит x10, если таковой расположен в идентифицируемой части теоремы. При рассмотрении каждого из этих новых вхождений (несколько "внутренних" и одно "наружное") будет происходить и фиксация расположенных на них логических символов.

Далее проверяется наличие информационного элемента (делители x10), указывающего, что текущее вхождение x10 является вхождением двуместной операции $f(a\ b)$, операнды a, b которой идентифицируются с помощью специальной процедуры, определяемой по символу f справочником "делители". Это крайне редкий случай, возникавший лишь однажды - для перечисления делителей одночленов в элементарной алгебре. Соответственно, справочник "делители" имеет лишь один фрагмент программы, выдающий по символу "умножение" название "Делители" процедуры перечисления. Компиляция в данном случае весьма проста, и описание ее мы опускаем.

За оператором "равно(x7 0)" следует ветвь приемов, срабатывающих на нулевом уровне сканирования. После оператора "стандарт(x10)" располагаются приемы, анализирующие операнды вхождения x10; если ни один из них не срабатывает, то выполняется переход через "ветвь 5" к рассмотрению приемов, анализирующих непосредственный надтерм вхождения x10. Рассмотрим сначала первую группу приемов. Здесь переменной x11 присваивается логический символ, расположенный в теореме приема по вхождению x10.

Предварительный учет указателей "заменазнака"

Если указатель приема "заменазнака(...)" допускает возможность появления по идентифицирующему вхождению некоторого логического символа f одноместной операции, который должен быть отнесен к внутренней переменной подтерма x10, то необходимо скорректировать идентифицирующее вхождение: если на нем расположена операция f, то она пропускается, и v идентифицируется с ее операндом. Более подробно, выполняются следующие действия. Находится информационный элемент (заменазнака x10 f x g p). Здесь x - переменная, на расположенное внутри x10 вхождение которой будет переноситься внешний знак f; g - символ одноместной операции, на который будет изменен при перемещении символ f; p - изначально равно 0, а должно стать программным выражением для вхождения, идентифицируемого с x10. Найденный информационный элемент присваивается переменной x12, причем проверяется, что у него p равно 0. Затем выделяются два подслучая: если в процессе идентификации компилятор пришел к вхождению x10 "снизу", двигаясь от его надтерма и не рассматривая ранее операндов вхождения x10, то действия продолжаются после контрольной точки "прием(16)". Если выход на x10 имел место при движении "наружу", от некоторого операнда x10, то - переход через "иначе 7".

Рассмотри сначала первый подслучай. Здесь рассматривается вхождение x10. Если на нем уже расположен символ f, то переменной x13 присваивается вхождение первого операнда операции x10, иначе x13 равно x10. Таким образом, предпринимается пропуск операции f со стороны идентифицируемого терма. В конце рассматриваемой ветви программы установка на идентификацию x9 будет изменена, так что x13 станет новым идентифицируемым вхождением. Переменной x14 присваивается оператор, проверяющий наличие символа f по идентифицирующему вхождению. Указатель приема "комплексное" модифицирует этот оператор для учета комплекснозначного аналога символа f. После перехода через "ветвь 8" создается оператор x15, проверяющий наличие по новому идентифицирующему вхождению того же символа, который имеется на новом идентифицируемом вхождении x13. Новое идентифицирующее вхождение будет присвоено (см. ниже) первой свободной программной переменной, т.е. переменной "начало(x3)". Как и выше, предпринимается коррекция x15 на комплекснозначный случай.

После перехода через "ветвь 1" вводится оператор, определяющий новое идентифицирующее вхождение (пропускается операция f по старому идентифицирующему вхождению, если она была). Здесь используется ранее созданный подтерм x14. Кроме того, добавляется оператор x15, сравнивающий заголовки новых идентифицируемого и идентифицирующего подтермов. Так как в особых случаях, определяемых прочими указателями приема (наличие "единичных" операндов и т.п.), эти заголовки совпадать не обязаны, то оператор x15 вставляется лишь после проверки отсутствия таких случаев. Далее происходит замена p на однобуквенный терм, образованный программной переменной "начало(x3)" для нового идентифицирующего вхождения; в установке на идентификацию корректируются v и X. Соответственно изменяются x10 и x11. Если на новом идентифицируемом вхождении расположен неоднобуквенный терм, то откат к переходу через "ветвь 6", где будет продолжена обработка установки x9. Иначе - идентификация подтерма завершается с помощью оператора "учетоперанда". Затем установка x9 исключается из списка x2, и откат к началу основного цикла идентификации.

Возвращаемся к рассмотрению подслучая, когда выход на x10 произошел при движении "наружу". После перехода через "иначе 7" попадаем на контрольную точку "прием(17)". Здесь добавляется оператор, рассматривающий внешнюю операцию для идентифицирующего вхождения; если эта операция есть f, то первой неиспользуемой программной переменной "начало(x3)" присваивается ее вхождение, иначе значением данной переменной оказывается само идентифицирующее вхождение. В результате значением переменной "начало(x3)" оказывается вхождение, которое фактически будет идентифицироваться с v, согласно указателю "заменазнака(...)". Оно регистрируется вместо старой версии идентифицированного с v вхождения в информационном элементе (операнд v ...), а также в информационном элементе (заменазнака ...), где в качестве p берется однобуквенный терм, образованный переменной "начало(x3)". Установка на идентификацию здесь никаких изменений не претерпевает, так как при отброшенной внешней операции f вхождение v должно сравниваться именно со "старой" версией идентифицирующего терма. Далее - откат к переходу через "ветвь 6".

Учет указателя "общаястепень"

После контрольной точки "прием(18)" рассматривается особый случай, когда вхождение v выделено указателем "общаястепень(p x A)". Этот указатель относится к

группе вхождений в теорему приема выражений $f(x\ y_1), \ldots, f(x\ y_n)$. Идентификация данных выражений происходит с помощью специального оператора p, предпринимающего попытку выделить из идентифицирующих термов $A_1, \ldots, A_n$ возможно большую "общую часть" и сопоставить ее переменной x. Например, такая общая часть может предствлять собой наибольший общий делитель одночленов, находящихся в показателях степени термов $A_1, \ldots, A_n$. Тогда в качестве p берется оператор "общаястепень", определивший выбор названия для данного указателя. После идентификации x переменные (обычные либо функциональные) $y_1, \ldots, y_n$ идентифицируются с "остатками" выражений $A_1, \ldots, A_n$; эти остатки находятся оператором p одновременно с выделением общей части.

Сразу заметим, что указатель "общаястепень" используется крайне редко, и при предварительном ознакомлении с компилятором данный подраздел можно опустить.

По указателю "общаястепень($p\ x\ A$)" при формировании программного блока был создан информационный элемент (общаястепень $p\ x\ y\ B\ C$). Здесь y - набор переменных $y_1, \ldots, y_n$; B - набор вхождений в теорему термов $f(x\ y_1), \ldots, f(x\ y_n)$; C - сначала набор из нулей, а затем набор программных выражений для термов, идентифицируемых с элементами B. Этот элемент присваивается переменной x12. Проверяется, что v содержится в списке B. Затем просматриваются вхождения из списка B, и все они регистрируются в наборе x8 идентифицированных вхождений. Позиция списка C, соответствующая вхождению v из текущей установки (операнд $v\ X$), заменяется на программное выражение "подтерм(X)". Если в наборе C после этого остаются нули, то переход через "ветвь 2", где предпринимается исключение текущей установки из списка x2 и откат к началу цикла идентификации.

Если же в наборе C нулей не остается, то появляется возможность идентифицировать переменные $x, y_1, \ldots, y_n$ с помощью оператора p. Этот оператор имеет две входные переменные - набор выражений, из которых будет выделяться "общая часть", и контекст замены. В случае приема, применяемого при сканировании задачи, такой контекст представляет собой четверку (задача - координаты вхождения в нее символа привязки приема); в случае пакетного оператора он имеет вид одноэлементного набора, состоящего из списка посылок. Оператор имеет две выходные переменные: первая - для набора значений переменных $y_1, \ldots, y_n$, вторая - для переменной x. Компилятор присваивает переменной x15 программное выражение для контекста замены, формирует обращение x16 к оператору p и регистрирует его в программе. Далее вводятся информационные элементы, в которых регистрируются термы, идентифицированные с переменными x, $y_1, \ldots, y_n$. После регистрации - откат к переходу через "ветвь 2", где текущая установка на идентификацию исключается из списка x2 и выполняется возвращение к началу цикла идентификации.

Начало идентификации не коммутативного и не ассоциативного символа операции

Расположенный перед контрольной точкой "прием(18)" оператор "ветвь 1" переводит к подразделу программы компилятора, в котором рассматривается случай идентификации операции, не обладающей специальной симметрией - не коммутативной, не ассоциативной и не допускающей перестановок операндов согласно справочнику "схемаоперандов" либо согласно указателям приема. Проверки отсутствия такого рода симметрии размещены вплоть до контрольной точки "прием(19)". Далее компиляция разветвляется. Через "иначе 1" - переход к случаю идентификации описателей "класс", "отображение", "выписка", "перечисление", "суммавсех" (отсекается

случай, когда "суммавсех" представляет собой не описатель, а символ обычной одноместной операции, применяемой к функции). Через "иначе 2" - переход к идентификации кванторов. Через "ветвь 3" - переход к общему случаю идентификации. Наконец, после "ветвь 3" начинается идентификация функциональных переменных. Описание перечисленных случаев выносим в нижеследующие подразделы.

Общий случай идентификации не ассоциативного и не коммутативного символа

После контрольной точки "прием(220)" рассматривается еще один особый подслучай - наличие указателя "перегруппировка(...)". Этот указатель определяет идентификацию некоторого двуместного отношения с перенесением всех его "ненулевых" членов в одну часть, по аналогии с идентификацией неравенств и уравнений. Проверяется, что текущее вхождение v в теорему приема упоминается в наборе A информационного элемента (перегруппировка A), т.е. должно идентифицироваться указанным образом. Предпринимается обращение к справочнику "перегруппировка", определяющему по текущему символу двуместного отношения f тройку (название двуместной ассоциативно-коммутативной операции g, члены которой будут переноситься из одной части в другую - знак, изменяемый при перенесении члена - единица a операции g). Далее проверяется, что по вхождению v расположено утверждение вида $f(t\ a)$. Оба корневых операнда операции v регистрируются в списке x8 идентифицированных вхождений, и в программу заносится оператор "перегруппировка(...)", выполняющий требуемую перегруппировку членов идентифицирующего отношения. После этой перегруппировки последнее примет вид $f(T\ a)$, причем вхождение операнда T будет присвоено новой программной переменной "начало(x3)". Вводится информационный элемент (операнд ...), сопоставляющий вхождению операнда t в идентифицируемый терм программное выражение "начало(x3)"для соответствующего вхождения T.

Реализуется обращение к процедуре "учетоперанда", осуществляющей текущую обработку найденного соответствия вхождений t и T. Эта программа добавляет в программу приема операторы, фиксирующие заголовок подтерма T; если t является переменной, то вводит информационные элементы, идентифицирущие эту переменную; определяет список новых установок на идентификацию (например, для операндов операции t); в особых случаях она может также предложить список дополнительно идентифицированных вхождений для регистрации их в списке x8. Обращение к такой процедуре обычно завершает действия приема, обрабатывающего установку на идентификацию. Наконец, текущая установка на идентификацию x9 исключается из списка x2, и выполняется откат к началу цикла идентификации.

Если с вхождением v не был связан указатель "перегруппировка", то переход через "ветвь 1" к контрольной точке "прием(22)". Здесь переменной x12 присваивается набор вхождений корневых операндов операции f, расположенной по вхождению v. Если операция f может иметь переменное число операндов, то в программу вставляется оператор, проверяющий совпадение этого числа в идентифицируемом и идентифицирующем термах; вводится информационный элемент (количествооперандов v), указывающий на наличие такого оператора. Далее - переход через оператор "ветвь 1".

Если число операндов операции v более 4, то переход через "иначе 1" - в этом случае для идентификации операндов придется использовать специальную их адресацию с помощью операторного выражения "операндномер(...)". Этот случай рас-

смотрим ниже. В противном случае операнды можно адресовать, используя стандартные ЛОСовские термины "первыйоперанд", "второйоперанд", "предпоследоперанд", "последнийоперанд". Переход к основной части программы будет выполняться далее через "ветвь 2". Однако, до этого перехода рассматриваются несколько специальных случаев.

Прежде всего, после контрольной точки рассматривается подслучай идентификации выражения "вариант(...)" при наличии указателя приема "извлечениеварианта". Здесь теорема приема представляет собой тождество вида "вариант(A t_1 t_2) $= t$", причем все переменные выражений t_1, t_2 встречаются в утверждении A. Прием применяется для преобразования условного выражения "вариант(B s_1 s_2)", которое идентифицируется не с теоремным термом "вариант(A t_1 t_2)", а с выражением "вариант(A $F(t_1)$ $F(t_2)$)". Это означает, что после идентификации A с B и определения всех переменных теоремного условного выражения предпринимается попытка представить s_1,s_2 как результаты подстановки t_1,t_2 в одно и то же выражение $F(x)$. После этого заменяющий терм строится как $F(t)$. Для определения F программа приема использует специальный оператор "извлечениеварианта(...)", которому сообщаются в качестве входных данных термы s_1,s_2, а также (после подстановки в них идентифицированных значений переменных) термы t_1,t_2,t.

Следующий особый подслучай - идентификация выражения "суммавсех(отображение(...))", выделенного указателем "видмногочлена(...)". В этом подслучае по вхождению v расположен терм вида

$$\sum_{i=1}^{n} f(i)A^{i-1},$$

идентифицируемый с многочленом от заданного выражения A. При идентификации значением переменной f становится терм "набор($c_1 \dots c_n$)", где $c_1, \dots, c_n$ - коэффициенты многочлена, упорядоченные по возрастанию степеней, а значением n - выражение, равное увеличенной на единицу степени многочлена. Компилятор проверяет, что идентифицируемый терм имеет указанный вид, и организует обращение к оператору "смногочлен" (при наличии указателя "комплексное" - "Смногочлен"), который и выполняет идентификацию многочлена. После регистрации значений f, n в информационных элементах программного блока и удаления обработанной установки на идентификацию - откат к началу цикла идентификации.

Наконец, последний из особых подслучаев, рассматриваемых в данной цепочке подфрагментов, - наличие указателя "частнпроизв". Такой указатель вводится, если теорема приема содержит выражение "частнпроизв(f n a)"для производной n - го порядка функции одного переменного f в точке a. Он обеспечивает автоматическое приведение к виду "частнпроизв(f 1 a)" идентифицирующих выражений "производная(f a)". Компилятор последовательно просматривает вхождения f, n, a, адресуемые с помощью операторных выражений "первыйоперанд", "второйоперанд", "последнийоперанд". f и a идентифицируются одинаковым образом как для терма "производная(f a)", так и для терма "частнпроизв(f n a)". Для идентификации n вводится оператор "альтернатива", который в случае символа "производная" сопоставляет этой переменной терм "1", а в случае символа "частнпроизв" - второй операнд идентифицирующего терма.

Возвращаемся к рассмотрению "главного ствола" процедуры, обрабатывающего идентифицируемые операции с не более чем 4 операндами. Он начинается с оператора "ветвь 2", расположенного после оператора "длинаменее(x12 5)".

Прежде всего, проверяется наличие указателя "единица", выделяющего операнды w текущей операции v, которые могут обращаться в единицу e. Это делается путем выделения соответствующего информационного элемента (единица v w e), присваиваемого переменной x13.

Если такой указатель имеется, то далее сначала рассматривается подслучай, в котором вхождение v выделено еще и указателем "дробь", допускающим перестановку операндов. Началом рассмотрения служит контрольная точка "прием(144)". После нее переменной x14 присваивается информационный элемент (дробь A_1 A_2 A_3 A_4), где A_1 - набор вхождений двуместных операций, при идентификации либо создании которых допустима (одновременная для всех) перестановка операндов. A_2, A_3 - на момент начала идентификации нули, а затем программные выражения для вхождения одной из этих операций и вхождения, идентифицированного с ее первым операндом. Эти выражения нужны для определения того, имела ли место перестановка операндов.

Компилятор просматривает еще не идентифицированные вхождения x15 операции v. Если A_2 и A_3 уже определились из ранее выполненной идентификации другой операции набора A_1, то переход через "иначе 6". В противном случае создается оператор x16, выбирающий некоторый операнд u идентифицирующей операции C, отличный от ее операндов, уже сопоставленных операндам v. Этот операнд присваивается первой неиспользуемой программной переменной "начало(x3)". Напомним, что вхождение идентифицирующей операции C является значением программной переменной "конец(x9)". Далее x16 включается в оператор x17, определяющий для x15 идентифицирующий терм с учетом элемента (единица v w e). Если идентифицирующая операция C невырождена и имеет тот же заголовок, что и v, то для x15 берется подтерм по вхождению u. Иначе рассматриваются два подслучая: x15 равно w, и тогда для него строится новый терм e, либо x15 не равно w, и тогда для него берется весь подтерм по вхождению C. Оператор x17 включается в программу приема. После этого создается информационный элемент (операнд x15 ...), ссылающийся на сопоставленное x15 идентифицирующее вхождение, а программное выражение, имеющее своим значением сопоставленный x15 идентифицирующий терм, присваивается переменной x18. Если A_2 равно 0, то оно заменяется на программное выражение, дающее вхождение идентифицирующей операции C, либо, в вырожденном случае, вхождение некоторого фиктивного терма. Если x15 является вхождением первого операнда, то после перехода через "ветвь 1" доопределяется также A_3. Наконец, найденный для x15 идентифицирующий терм x18 регистрируется с помощью оператора "учетоперанда", и откат к выбору очередного вхождения x15. По окончании просмотра всех x15 - откат к началу цикла идентификации.

Если в процессе просмотра A_2 и A_3 уже определились, то они позволяют однозначно определить вхождение u операнда идентифицирующей операции C, соответствующее x15. Это делается оператором x16. Как и в предыдущем случае, x16 включается далее в оператор x17, учитывающий указатель "единица".

Если информационный элемент "единица" не дополняется элементом "дробь", то после обнаружения его выполняется переход через "ветвь 4", к контрольной точке "прием(40)". Здесь рассматривается лишь случай, когда число операндов операции v равно 2 - других не ассоциативных и не коммутативных операций с единицей просто не возникало. Последовательно просматриваются не идентифицированные операнды операции v; вхождение текущего такого операнда присваивается переменной x14. Вводится оператор x15, проверяющий совпадение заголовков идентифицирующей и

идентифицируемой операций. При наличии указателя "комплексное" он заменяется на дизъюнкцию с оператором, учитывающим возможную комплекснозначную версию заголовка. Если вхождение x14 выделено указателем "единица", то к программе присоединяется оператор, определяющий идентифицирующий терм для операнда x14 в зависимости от того, истинно ли x15 или ложно; во втором случае этот терм является указанной в информационном элементе (единица ...) константой. Если же x14 не выделено указателем "единица", то ему сопоставляется не идентифицирующий терм, а идентифицирующее вхождение - это экономнее, так как не требует специального переписывания подтерма. В обоих случаях значением x16 становится программное выражение для идентифицирующего терма либо вхождения; значением x17 - индикатор терма, равный 1 в первом случае и 0 во втором. Для второго случая вводится информационный элемент (операнд x14 x16). Далее предпринимается завершающее рассмотрение текущего операнда x14 обращение к процедуре "учет-операнда". По окончании просмотра операндов x14 - откат к началу цикла идентификации.

Если вхождение v не выделено указателем "единица", то переход через "иначе 3", где проверяется, имеется ли информационный элемент (отрицание v). Такой элемент означает, что по вхождению v расположена одноместная операция $f(x)$, для которой выполнено тождество $f(f(x)) = x$. Если он есть, то после контрольной точки "прием(41)" предпринимается вставка в программу оператора, определяющего идентифицирующий терм t для x. Если на идентифицирующем вхождении C, соответствующем теоремному вхождению v, расположена операция f, то t полагается равным ее операнду. Иначе t получается вводом внешней операции f для подтерма по вхождению C. После определения t происходит обращение к оператору "учетоперанда" и откат к началу цикла идентификации.

Наконец, после контрольной точки "прием(42)" рассматриваются указатели с заголовком "дробь".

Сначала рассматривается случай однобуквенного указателя "дробь". Он означает, что в приеме замены заменяющий и заменяемый термы имеют корневые двуместные операции, причем допустима одновременная перестановка операндов обеих операций. Проверяется, что v есть корневое вхождение заменяемого терма. Затем просматриваются еще не идентифицированные операнды x13 операции v. Пусть C обозначает идентифицирующее вхождение для v. В программу заносится оператор, определяющий идентифицирующее вхождение для x13 как произвольное еще не использованное при идентификации вхождение операнда операции C. По этому вхождению создается информационный элемент (операнд x13 ...) и предпринимается обращение к оператору "учетоперанда". После просмотра операндов x13 - откат к началу цикла идентификации.

Затем (после перехода через "ветвь 1") рассматривается случай неоднобуквенного указателя "дробь", явно задающего вхождения с переставляемыми операндами. Этот указатель порождает упоминавшийся выше информационный элемент (дробь A_1 A_2 A_3 A_4). Проверяется, что v имеется в списке A_1 вхождений, выделенных указателем. Дальнейшие действия в этом подслучае - те же, что при одновременном выделении вхождения v указателями "единица" и "дробь", с пропуском тех шагов, которые относились к учету указателя "единица".

На этом цепочка фрагментов, связанных с анализом выделяющих вхождение v информационных элементов, завершается. После нее - откат к оператору "ветвь 2", приводящему к контрольной точке "прием(43)". Это - общий случай идентификации

операндов операции v, наиболее простой из перечисленных. Переменной x13 присваивается набор логических символов, адресующих операнды операции v. Если число этих операндов равно 3 либо 4, то используются все элементы набора x13; иначе - только первые один либо два. Затем синхронно просматриваются не идентифицированные операнды операции v и соответствующие им символы набора x13. Создается программное выражении x16 для идентифицирующего вхождения текущего операнда, и выполняется обращение к оператору "учетоперанда". По окончании просмотра - откат к переходу через "ветвь 1". Здесь, после контрольной точки "прием(44)", проверяется отсутствие внешнего для v (т.е. такого, чьим операндом служит v), еще не идентифицированного, но подлежащего идентификации вхождения. Если его нет, то установка x9 исключается из списка установок, и откат к началу цикла идентификации. В противном случае - откат переходу через "ветвь 5" в фрагменте, начинающемся с контрольной точки "прием(13)". Этот переход приводит к попытке идентификации внешней операции вхождения v.

В заключение данного подраздела рассмотрим ранее отложенный случай, когда операция по вхождению v имеет более 4 операндов. Он начинается с перехода через "иначе 1" после оператора "длинаменее(x12 5)", приводящего к контрольной точке "прием(218)". Переменной x13 присваивается 0 - эта переменная будет хранить символьный номер текущего идентифицируемого операнда операции v - после рассмотрения очередного операнда x14 она увеличивается на 1. Переменной x16 присваивается операторное выражение "операндномер(...)", определяющее идентифицирующее вхождение для текущего операнда. Оно используется в информационном элементе (операнд ...), вводимом для текущего операнда, а также при обращении к процедуре "учетоперанда". По окончании просмотра операндов - проверка того, что внешнее вхождение для v уже идентифицировано, и удаление в этом случае текущей установки x9 из списка установок x2.

Идентификация описателей

Точка выхода на ветвь программы, в которой происходит компиляция процедуры идентификации описателя, уже была указана выше. Эта ветвь начинается с контрольной точки "прием(46)"; перейти к ней можно также из пункта ("Типы установок на идентификацию", "Установка на идентификацию (операнд ...)", "Не коммутативный и не ассоциативный символ", "Идентификация описателей ...").

Прежде всего, проверяется наличие информационного элемента (матрица v). Этот элемент означает, что по вхождению v расположено выражение "отображение(i j и(принадлежит(i номера(1 m)) принадлежит(j номера(1 n))) значение(A набор(i j)))", идентифицируемое с прямоугольной матрицей A размера $m \times n$. Для регистрации результатов идентификации будет использован информационный элемент (элементыматрицы A T M N), где T, M, N - соответственно, программные выражения для набора наборов (строк) элементов матрицы, заданных термами; для числа строк и числа столбцов (также заданных термами). Компилятор вводит в программу приема оператор "элементыматрицы(...)". Этот оператор по заданному вхождению идентифицирующего терма проверяет, что он имеет вид "строки(...)" либо "столбцы(...)" и определяет прямоугольную матрицу. Трем выходным переменным оператора присваиваются указанные выше значения для T, M, N. Если информационный элемент (элементыматрицы A ...) уже имелся, то вводятся операторы, проверяющие, что вновь найденные значения T, M, N совпадают со старыми. Иначе создается новый информационный элемент (элементыматрицы ...). Установка на идентификацию x9

исключается из списка x2. Далее находятся теоремные переменные m, n для размерности матрицы. Если значение какой-либо из них уже было идентифицировано, то вводится оператор, проверяющий совпадение вновь найденного значения со старым. Иначе - вводятся информационные элементы (терм ...) и (транслвыражения ...), определяющие идентифицированное значение переменной. Первый из них представляет это значение в формате терма, второй - в формате десятичного числа. После перечисленных действий - откат к началу цикла идентификации.

При отсутствии информационного элемента (матрица v) - переход через "ветвь 1". Здесь проверяется наличие не идентифицированных вхождений связывающей приставки описателя. Если их нет, то переход через "ветвь 1". Иначе - переменной x12 присваивается набор переменных связывающей приставки.

Сначала рассматривается подслучай, когда имеется указатель "кортежпеременных(...)", разрешающий идентификацию единственной переменной x связывающей приставки описателя не с одной переменной, а с набором переменных. Тогда вводятся информационные элементы (наборпеременных x ...), (переменные v ...), (терм x ...). Первый из них определяет список переменных $y_1, \ldots, y_n$, с которым идентифицируется x; второй - указывает на тот же самый список переменных как на связывающую приставку описателя по вхождению v; третий - определяет терм для x, т.е. выражение "набор($y_1 \ldots y_n$)" при $n > 1$ и однобуквенный терм y_1 при $n = 1$.

Если указанный подслучай не имеет места, то переход через "иначе 2". Переменной x13 присваивается символьное число переменных связывающей приставки. Если оно равно 1, то переход через "иначе 1". Иначе - вводятся два оператора, первый из которых присваивает связывающую приставку первой неиспользуемой программной переменной, а второй - проверяет, что ее длина равна x13. Если имеются указатели "элемент(...)", определяющие идентификацию переменных связывающей приставки без учета их порядка, то эти переменные x последовательно просматриваются, и для каждого вводится оператор, извлекающий значение x из остатка связывающей приставки. Найденные значения регистрируются в информационных элементах (переменная x ...). При отсутствии указателей "элемент(...)" либо по их исчерпании - переход через "ветвь 2", где оставшиеся переменных связывающей приставки идентифицируются согласно своим номерам. По окончании все вхождения связывающей приставки заносятся в x8.

Наконец, если связывающая приставка состояла из одной переменной (см. выше), то после перехода через "иначе 1" вводится оператор, проверяющий, что число операндов описателя равно 2 (описатель "класс") либо 3 (прочие описатели). Если связанная переменная уже была идентифицирована, то проверяется ее совпадение с новым значением, иначе - вводится информационный элемент (переменная ...) с результатом ее идентификации.

После идентификации связывающей приставки - откат к переходу через "ветвь 1". Здесь просматриваются операнды x12 вхождения v, не относящиеся к связывающей приставке. В случае описателя "класс" это один лишь последний операнд; в прочих случаях - последний и предпоследний. Вводятся программные выражения "последнийоперанд(...)", "предпоследоперанд(...)" для соответствующих идентифицирующих вхождений. Эти выражения регистрируются в информационных элементах (операнд x12 ...), затем выполняется обращение к процедуре "учетоперанда". По окончании просмотра операндов x12 - проверяется, идентифицированы ли все операнды вхождения x10 и его внешнее вхождение. Если это так, то установка на идентификацию x9 исключается, и откат к началу цикла идентификации.

Идентификация кванторов

Точка выхода на ветвь программы, в которой происходит компиляция процедуры идентификации квантора, была указана выше. Эта ветвь начинается с контрольной точки "прием(45)"; перейти к ней можно также из пункта ("Типы установок на идентификацию", "Установка на идентификацию (операнд ...)", "Не коммутативный и не ассоциативный символ", "Идентификация кванторов", "Исходная точка").

После контрольной точки "прием(322)" предпринимается попытка идентифицировать переменные сязывающей приставки для тех случаев, когда это можно сделать однозначно или почти однозначно. Здесь - лишь предварительное рассмотрение связывающей приставки; основное ее рассмотрение будет происходить после идентификации подкванторных утверждений. Выделены следующие случаи:

а) Связывающая приставка x12 состоит из единственной переменной, для которой имеется указатель "кортежпеременных(...)". Тогда она идентифицируется со списком всех переменных связывающей приставки. Вводятся определяющие идентификацию информационные элементы (наборпеременных ...), (переменные ...), (терм ...). Если v есть корень заменяемого подтерма, то вводится также элемент (квантор ...).

б) Связывающая приставка состоит из не более чем двух переменных, каждая из которых должна быть идентифицирована с единственной переменной. Если переменная одна, то она идентифицируется однозначно; если их две, то в программу приема вводятся перечисляющие операторы, организующие рассмотрение обеих альтернатив идентификации.

Далее (вне зависимости от того, удалось ли идентифицировать переменные связывающей приставки) - откат к переходу через "ветвь 1". Здесь, после контрольной точки "прием(323)", продолжается предварительное рассмотрение простейших случаев идентификации. Проверяется отсутствие информационного элемента (количествоопердандов v), означающего, что уже введен оператор, фиксирующий требуемое количество подкванторных утверждений. Отбрасывается особый случай идентификации квантора общности $\forall_x F(x)$, выделенного указателем "импликация", у которого $F(x)$ - функциональная переменная. Такая идентификация встречается в приемах вывода теорем. Наконец, отбрасывается случай идентификации заменяемого квантора в приеме с указателем "кванторнаясвертка" - для него упрощенная схема идентификации не подходит. Для простейшей идентификации кванторов выделяются два подслучая - квантор существования $\exists_{x_1 \ldots x_n} A$, у которого заголовок утверждения A отличен от конъюнкции, а также (переход через "иначе 3") квантор общности $\forall_{x_1 \ldots x_n}(A \to B)$, выделенный указателем "внешнийквантор" и имеющий единственный антецедент. Напомним, что указатель "внешнийквантор" запрещает учет при идентификации кванторной импликации возможной контрапозиции консеквента с одним из антецедентов, а без такого указателя, по умолчанию, компилятор учитывает эту возможность. В первом случае происходит идентификация A как последнего операнда идентифицирующего квантора существования; во втором - идентификация A как конъюнкции всех антецедентов идентифицирующей кванторной импликации. В каждом из случаев далее реализуется откат к началу цикла идентификации. Если ни один из случаев не имел места, то после перехода через "ветвь 2" выполняется попытка создания оператора, фиксирующего число подкванторных утверждений. Проверяется, что среди подкванторных утверждений теоремы приема нет переменных либо функциональных переменных - тогда число подкванторных утверждений идентифицирующего квантора было бы заранее не определенным. Затем находится

число x12 подкванторных утверждений идентифицируемого квантора. Для квантора существования оно равно числу конъюнктивных членов подкванторного утверждения; для квантора общности - числу антецедентов. Наконец, в программу приема вставляется оператор, сравнивающий понимаемое указанным образом число подкванторных утверждений идентифицирующего квантора с x12; факт наличия такого оператора отмечается вводом информационного элемента (количествоопердандов v).

Если действия, выполнявшиеся после контрольной точки "прием(323)", не вызвали отката к началу цикла идентификации, то реализуется переход через оператор "ветвь 1" в начале фрагмента, содержащего данную точку. Здесь, после оператора "равно(x11 существует)", предпринимается рассмотрение еще одного особого случая - выделенный указателем "внешнийквантор" квантор существования $\exists_{x_1 \ldots x_n} A$ имеет выделенный указателем "развертка" конъюнктивный член утверждения A. Тогда делается один промежуточный шаг идентификации - находится последний операнд идентифицирующего квантора, его вхождение присваивается первой неиспользуемой программной переменной y, и вводится информационный элемент (операнд w y), идентифицирующий корневое вхождение w утверждения A.

Наконец, после отката к оператору "ветвь 2", расположенному перед оператором "равно(x11 существует), начинается идентификация подкванторных утверждений в общем случае. Она отмечена контрольной точкой "прием(324)". Просматриваются вхождения x12 атомарных подкванторных утверждений идентифицируемого кванторного утверждения. В случае кванторной импликации $\forall_{x_1 \ldots x_n}(A_1 \,\&\, \ldots \,\&\, A_n \to A_0)$ это антецеденты $A_1, \ldots, A_n$ и консеквент A_0; в случае квантора существования $\exists_{x_1 \ldots x_n}$ $(A_1 \,\&\, \ldots \,\&\, A_n)$ - конъюнктивные члены $A_1, \ldots A_n$. Текущее утверждение x12 сопровождается индикатором x13, равным 0 в случае антецедента и 1 в случае консеквента. Для квантора существования x13 равно 0, так как после преобразования его отрицания к виду кванторной импликации утверждения A_i попадают в антецеденты.

После определения значений x12 и x13 в компиляторе размещена еще одна вставка, обрабатывающая особый случай. Она начинается после оператора "равно(x11 длялюбого)" и относится к ситуации, когда идентифицируемый квантор имеет вид $\forall_x F(t)$; F - функциональная переменная, выделенная указателем приема "импликация(F)". Такая ситуация встречается при компиляции приемов с заголовком "логвывод". Выражение $F(t)$ идентифицируется со вспомогательным термом "то(A_0 $A_1 \ldots$ A_n)", где A_0 - консеквент идентифицирующей кванторной импликации; $A_1, \ldots, A_n$ - ее антецеденты. Соответственно, компилятор вводит оператор, присваивающий первой неиспользуемой программной переменной указанный вспомогательный терм, и регистрирует ее значение как идентифицирующее для $F(t)$.

Если указанный особый случай не имеет места, то переход через "ветвь 1". Здесь переменной x14 присваивается вхождение в теорему, полученное из x12 отбрасыванием отрицания, если таковое имеется. При отбрасывании отрицания значение x13 меняется на противоположное. Если идентифицируемый квантор выделен указателем "внешнийквантор", то при идентификации не будут предприниматься попытки контрапозиций - перестановок местами консеквента и одного из антецедентов, с переходом к их отрицаниям. Они не будут выполняться также и для приема с заголовком "теорема", не имеющего указателя "обобщконсеквент", выделяющего квантор. В обоих этих случаях - переход через "иначе 1". Иначе - реализуется компиляция процедуры, предпринимающей данные попытки.

После оператора "символ(x14 значение)" анализируется подслучай, когда по вхождению x14 расположена еще не идентифицированная функциональная переменная $F(t)$, причем все остальные атомарные подкванторные утверждения рассматри-

ваемого квантора уже идентифицированы. Сначала рассматривается подслучай, в котором вхождение v выделено указателем "обобщконсеквент", означающим, что при идентификации квантора допускаются произвольные контрапозиции антецедентов с его консеквентом. Если консеквент уже идентифицирован, то для идентификации $F(t)$ применяется оператор "обобщантецеденты", определяющий конъюнкцию тех антецедентов, которые возникли после использованной контрапозиции. Заметим, что фактически указатель "обобщконсеквент" использовался лишь в единственном приеме, причем относящемся к логическому выводу в базе теорем. Поэтом основной случай здесь - это переход через "ветвь 3". Переменной x16 присваивается набор вхождений уже идентифицированных атомарных подкванторных утверждений с отброшенными отрицаниями. На основе этого набора создается вставляемый в программу приема оператор "импликанты(...)", определяющий набор вхождений еще не идентифицированных подкванторных утверждений, а также конъюнкцию этих утверждений, преобразованных контрапозициями в антецеденты либо в консеквент - согласно индикатору антецедента x13. Данная конъюнкция и идентифицируется с функциональной переменной $F(t)$.

Если подслучай идентификации функциональной переменной $F(t)$ "по остаточному принципу" не имел места, то переход через "ветвь 2". Здесь снова проверяется наличие указателя "обобщконсеквент", выделяющего вхождение v. Если он есть, то переход через "иначе 1", где идентификация предпринимается с помощью оператора "обобщконсеквент". В противном случае в программу приема вставляется оператор "импликант(...)", перечисляющий все вхождения атомарных подкванторных утверждений с отброшенными отрицаниями, которые можно преобразовать контрапозициями в антецеденты либо в консеквент, согласно индикатору x13. После него размещаются операторы, отбрасывающие уже идентифицированные ранее вхождения. Текущее утверждение x14 идентифицируется с таким вхождением.

Вернемся к рассмотрению указанной выше ветви, в которой идентификация квантора происходит без попыток контрапозиций. Она начинается с контрольной точки "прием(230)". Переменной x15 присваивается заголовок рассматриваемого атомарного подкванторного утверждения x14 с отброшенным отрицанием (если оно было). Если x15 - переменная, то переход через "иначе 1".

Если x15 - логический символ, то сначала рассматривается особый случай - наличие указателя "консеквент(...)", выделяющего идентифицируемую кванторную импликацию. Этот указатель означает, что кванторная импликация теоремы идентифицируется, вообще говоря, не с единственной кванторной импликацией, а с группой всех имеющихся в текущем контексте кванторных импликаций, отличающихся друг от друга только консеквентами. Такие группы импликаций могут возникать в условиях и посылках задач как результат преобразования к виду конъюнкции импликаций какой-либо исходной импликации с конъюнкцией в консеквенте. Если есть указатель "консеквент(...)", то в качестве идентифицирующего утверждения для консеквента теоремной импликации берется конъюнкция всех консеквентов импликаций указанной группы. Сразу заметим, что данный указатель используется пока в единственном приеме - доказательстве индукцией по длине набора. Обработка указателя компилятором начинается с проверки того, что все антецеденты импликации уже идентифицированы. Это означает, что ранее была выбрана какая-то одна из импликаций идентифицирующей группы, причем программное выражение для нее было зарегистрировано в информационном элементе (антецедент v ...). Теперь добавляются два новых оператора программы приема, первый из которых определяет

список всех импликаций с такими же антецедентами (кроме выбранной ранее), а второй - формирует конъюнкцию всех консеквентов импликаций идентифицирующей группы.

При отсутствии указателя "консеквент" - переход через "ветвь 2". Здесь проверяется, что x14 не является функциональной переменной. Если это не так, то переход через "иначе 1". В противном случае в программу приема вводятся операторы, перечисляющие в качестве идентифицирующих для x12 вхождения подкванторных утверждений (т.е. антецедентов, консеквента, конъюнктивных членов утверждения, связанного квантором существования), еще не использованные ранее для идентификации. При этом консеквент идентифицируется с консеквентом, а антецедент - с одним из антецедентов.

Если x14 является функциональной переменной, то попадаем на контрольную точку "прием(231)", соответствующую пункту ("Идентификация кванторов", "Идентификация функциональной переменной в антецеденте либо консеквенте", "Исходная точка"). Прежде всего, проверяется, не является ли данная функциональная переменная f уже идентифицированной. Это распознается по наличию информационного элемента (функция $f\ A\ B\ C$), у которого C не равно 0. Здесь A - набор теоремных переменных - аргументов f. Рассматриваемый элемент возник при идентификации вхождения в теорему терма $f(A)$, в то время как теперь идентифицируется некоторый терм $f(A')$. Если какие - либо аргументы U в наборе A' отличаются от соответствующих аргументов u набора A, то для создания операторов, проверяющих представимость текущего идентифицирующего терма в виде $f(A')$, необходимо иметь уже идентифицированными переменную u и все переменные из U. Поэтому выполняется проверка того, что все указанные переменные идентифицированы. Если это не так, то обработка текущего x12 и остальных подкванторных утверждений откладывается; в противном случае - откат к переходу через "ветвь 1".

Если x12 - вхождение консеквента кванторной импликации, либо корневое вхождение утверждения под квантором существования, то идентификация x12 происходит однозначным образом - с последним операндом идентифицирующих кванторной импликации либо квантора существования. В противном случае - переход через "иначе 1".

Здесь сначала рассматривается подслучай кванторной импликации. Если неидентифицированным остался лишь один ее антецедент x12, то он идентифицируется с конъюнкцией остатка антецедентов идентифицирующей импликации. Если таких антецедентов более одного, то переход через "иначе 2". Далее рассматриваются следующие подслучаи:

1. Функциональная переменная x14 выделена указателем "антецедент", определяющим ее идентификацию с единственным антецедентом идентифицирующей импликации K. Эта ситуация распознается по наличию информационного элемента (предоперанд ...). Тогда в программу приема вводятся операторы, выбирающие произвольный еще не использованный антецедент импликации K, и x12 идентифицируется с ним.

2. Все переменные, встречающиеся в подтерме x14, уже идентифицированы. Тогда определяется операторное выражение x17 для данного подтерма. В программу приема вводятся операторы, выбирающие еще не использованный антецедент импликации K и проверяющие совпадение его с x17.

3. Функциональная переменная x14, т.е. терм вида $f(t)$, выделена указателем

"перечень(f A)". Тогда она идентифицируется с конъюнкцией всех еще не использованных антецедентов идентифицирующей импликации, удовлетворяющих сформулированному на языке фильтров ГЕНОЛОГа условию A. В этом условии текущий антецедент обозначается переменной f. Компилятор определяет операторное выражение x19 для списка еще не использованных антецедентов, затем создает вспомогательный программный блок x21 для обработки цикла просмотра элементов набора x19. Сначала в программе блока x21 регистрируется единственный оператор, перечисляющий элементы набора x19. Затем в нее добавляется результат компиляции фильтра A. После этого из x21 извлекается цепочка операторов, порождающая операторное выражение "выписка(...)" для искомого списка антецедентов. Наконец, формируется операторное выражение x24 для конъюнкции элементов данного списка, которое и дает идентификацию антецедента x12.

Далее рассматривается подслучай идентификации функциональной переменной x14 под квантором существования. Если x14 была выделена указателем "операнд(...)", означающим, что она идентифицируется с единственным операндом ассоциативно - коммутативной операции (в данном случае конъюнкции), то имеется информационный элемент (второйоперанд ...). Тогда, если никакие другие конъюнктивные члены утверждения под квантором еще не идентифицированы, в программу приема вводится оператор, выбирающий произвольный конъюнктивный член соответствующей идентифицирующей конъюнкции. Другой рассматриваемый подслучай - если x12 есть последний еще не идентифицированный конъюнктивный член. Тогда он идентифицируется по остаточному принципу.

Наконец, переходим к рассмотрению подслучая, когда x14 есть вхождение обычной переменной. Этот подслучай начинается с контрольной точки "прием(326)", достижимой из пункта ("Идентификация кванторов", "Идентификация обычной переменной в антецеденте либо консеквенте") оглавления программ. Если x12 - вхождение консеквента кванторной импликации, то ему сопоставляется вхождение консеквента идентифицирующей импликации. В противном случае компилятор проверяет, что все отличные от x12 атомарные подкванторные утверждения рассматриваемого квантора уже идентифицированы, и сопоставляет x12 конъюнкцию оставшихся неиспользованными атомарных подкванторных утверждений идентифицирующего квантора.

По окончании цикла просмотра корневых вхождений x12 атомарных подкванторных утверждений - откат к переходу через "ветвь 1", где проверяется, что все эти вхождения уже идентифицированы. Если это так, то переход через "ветвь 1". Здесь располагается контрольная точка "прием(327)", начиная с которой предпринимается идентификация связывающей приставки квантора. Прежде всего, проверяется наличие не зарегистированных в списке x8 вхождений переменных этой приставки. Если таковых нет, то приставка уже обработана. В этом подслучае проверяется, идентифицирована ли непосредственная надоперация квантора. Если идентифицирована, то текущая установка на идентификацию x9 удаляется, и откат к началу цикла идентификации. Если не идентифицирована, то - откат к рассмотрению этой надоперации.

Если же имеется еще не идентифицированное вхождение переменной связывающей приставки, то x12 присваивается список всех переменных связывающей приставки квантора.

Сначала рассматривается случай, когда x12 состоит из единственной переменной x. Вообще говоря, это не означает, что идентифицирующий квантор K тоже будет иметь единственную связывающую переменную: при наличии указателя "кванторнаясвертка" допускается группировка части подкванторных утверждений под вспомогательный квантор, который и будет иметь одну переменную, идентифицируемую с x; если переменная x выделена указателем "кортежпеременных", то она может идентифицироваться с произвольным набором связывающих переменных. Однако, при отбрасывании указанных подслучаев, x однозначно идентифицируется с единственной связывающей переменной квантора K. Здесь разбираются два подслучая: x уже была идентифицирована ранее, либо (переход через "ветвь 3") еще не идентифицирована. В обоих случаях в программу приема вставляются операторы, проверяющие, что K имеет единственную связывающую переменную y. В первом случае проверяется, что эта переменная совпадает с той, которая ранее была сопоставлена x, во втором - вводится информационный элемент (переменная x ...), сохраняющий программное выражение для y. Если непосредственная надоперация квантора уже идентифицирована, то далее происходит удаление текущей установки на идентификацию x9. В любом случае после этого - откат к началу цикла идентификации.

Если случай однозначной идентификации единственной переменной x набора x12 не имел места, то переход через "ветвь 2". Здесь рассматривается случай, в котором все переменные связывающей приставки x12 уже идентифицированы; части из них сопоставлены наборы переменных, части - единичные переменные. Определяется программное выражение x15 для объединенного списка переменных, сопоставленных переменным списка x12. Если имеется указатель "кванторнаясвертка", причем идентифицируемый квантор имеет корневое вхождение в заменяемой части теоремы, то проверяется лишь включение списка x15 в связывающую приставку P идентифицирующего квантора и различие его элементов; иначе проверяется, что списки x15 и P одинаковы с точностью до перестановки. Далее, как и выше, проверка возможности удаления установки x9 и откат к началу цикла идентификации.

Идентификация функциональных переменных

Под функциональной переменной понимается терм вида "значение(f t)" либо "значениемн(f t)", где f - переменная, t - переменная либо набор различных переменных, если имеются указатели приема, определяющие идентификацию такого терма с термами достаточно общего вида (например, указатель "отображение(f)"). Идентификация функциональной переменной заключается в том, что для f вводится информационный элемент, играющий роль шаблона при идентификации либо создании любых термов вида "значение(f T)", "значениемн(f T). Тип этого элемента соответствует типу функциональной переменной. Компилятор обрабатывает идентификацию функциональных переменных в различных точках - главным образом в рамках процедуры "учетоперанда", однако некоторые специальные случаи вынесены в ветвь процедуры "идентификатор", начинающуюся с контрольной точки "прием(328)". Выйти на эту контрольную точку можно через пункт ("Установка на идентификацию (операнд ...)", "Не коммутативный и не ассоциативный символ", "Идентификация функциональных переменных", "Исходная точка") оглавления программ.

Оператор "равно(x11 значение)" начинает рассмотрение наиболее часто встречающегося случая функциональных переменных вида "значение(f t)".

После контрольной точки "прием(134)" идет подслучай переменной, выделенной указателем "заменатермов". Здесь используется информационный элемент (замена-

термов $f\ v\ P\ M$), у которого v - вхождение в теорему функциональной переменной; P, M - сначала нули, а затем операторные выражения, соответственно, для идентифицирующего терма, сопоставляемого функциональной переменной, и для набора вхождений в него терма t', построенного по теоремному терму t. Этот элемент позволяет идентифицировать либо строить терм вида "значение($f\ T$)" как результат замены в идентифицирующем терме P всех вхождений списка M на терм, T', построенный по теоремному терму T. После обнаружения данного элемента компилятор проверяет, что все переменные в t уже идентифицированы и определяет операторное выражение x14 для t'. Далее, в зависимости от того, определились ли уже ненулевые P и M, компилятор либо вводит в программу приема операторы, присваивающие их значения новым программным переменным и доопределяет информационный элемент (заменатермов ...), либо сравнивает идентифицирующий терм с термом, определяемым по P и M указанной выше заменой вхождений. В обоих случаях установка x9 исключается, и откат к началу цикла идентификации. Заметим, что после оператора программы приема, определяющего список M, размещается оператор, проверяющий непустоту данного списка, т.е. представимыми в виде "значение($f\ t$)" считаются лишь термы, имеющие хотя бы одно вхождение t'.

При отсутствии указателя "заменатермов", выделяющего рассматриваемую функциональную переменную, переход через "иначе 5", где x12 присваивается символ f.

После контрольной точки "прием(135)" рассматривается идентификация функциональной переменной "значение($f\ t$)", выделенной указателем "сммногочлен($f\ a$)". Этот указатель означает, что идентифицирующий терм будет представляться как многочлен от выражения t', причем способ определения t' задается выражением a. Если a - логический символ "неизвестные", то степени t' группируются из всех содержащих неизвестные текущей задачи множителей одночленов; если a - переменная, то степени t' группируются из всех содержащих идентифицированную с a переменную множителей одночленов; если a имеет вид "терм(B)", то t' явно определяется операторным выражением B. Наконец, в указателе может отсутствовать a, и тогда степени t' группируется из всех неконстантных множителей одночленов. В начале компиляции по указателю "сммногочлен" вводится информационный элемент (сммногочлен $f\ K$). До идентификации функциональной переменной элемент K равен 0, а затем он становится программным выражением для набора коэффициентов многочлена, расположенных по возрастанию степеней. В этот набор вставлены нулевые коэффициенты; элементы набора представлены в формате термов. Компилятор вводит в программу приема обращение к оператору "сммногочлен" либо (при наличии указателя "комплексное") "смМногочлен"; этот оператор по вхождению корня идентифицирующего подтерма и по значению a предпринимает попытку распознать многочлен требуемого вида и выдает на выходе выражение t' вместе с набором коэффициентов.

Следующий случай идентификации рассматривается после контрольной точки "прием(240)". Здесь функциональная переменная "значение(f набор(значение($g\ t$) r))" выделена указателем "функподст($f\ g$)". Это - крайне редко встречающийся случай идентификации, используемый в приемах с заголовком "теорема". Идентифицирующий терм содержит переменную g только в виде "значение(g ...)", причем символ "значение" встречается явно. Компилируемый прием вывода теорем будет заменять все функциональные конструкции "значение(g ...)" на некоторый константный терм C, выбирая в качестве функции g константу. Компилятор вводит в программу приема оператор, проверяющий, что g встречается только как первый операнд термов "значение(...)" и создает информационный элемент (функподст $f\ A\ B$), у которого A

- программное выражение для идентифицирующего терма, сопоставляемого функциональной переменной, B - набор вхождений в него подтермов вида "значение(g ...)". Этот элемент будет использован как шаблон при построении результата перехода от g к константе.

Далее, после контрольной точки "прием(241)", рассматривается случай функциональной переменной "значение(f набор(t_1 t_2))", выделенной указателем "Бинарнаяоперация(f)". Этот указатель тоже используется лишь в приемах вывода теорем. Идентифицирующий терм должен иметь вид $g(A\ B)$ либо $h(g(A\ B))$, где g, h - логические символы, причем t_1, t_2 идентифицируются с операндами A, B при возможной их перестановке. Переменная f считается идентифицированной с логическим символом g (даже при наличии внешней операции h). Компилятор вставляет в программу приема обращение к процедуре "Бинарнаяоперация", распознающей термы указанного вида и выдающей символы g, h, индикатор перестановки операндов A, B и вхождения операндов. Эти значения регистрируются в информационном элементе (Бинарнаяоперация ...) и используются при построении новых термов вида "значение(f набор(T_1 T_2))".

После контрольной точки "прием(159)" рассматривается случай функциональной переменной "значение(f t)", выделенной указателем "симметрично" либо "символ". Здесь t - либо единственная переменная x, либо набор различных переменных $x_1, \ldots, x_n$. Идентифицирующий терм должен иметь вид $g(\ldots)$, где число операндов у g равно числу переменных у t. При этом переменная f идентифицируется с логическим символом g. Указатель "симметрично" отличается от указателя "символ" лишь тем, что допускает произвольные перестановки операндов при идентификации. Почти все случаи использования данных указателей приходятся на вывод теорем и на текстовый анализ. Они нужны для идентификации операции заданной арности, когда заголовок ее заранее неизвестен. Компиляция пока предусмотрена только для числа переменных в наборе t, не превосходящего 4.

Завершает перечень специальных подслучаев идентификации функциональных переменных вида "значение(f t)" рассмотрение указателя "внешфункция(f x)", начинающееся после контрольной точки "прием(253)". Идентифицирующий терм здесь должен иметь вид $g(\ldots\ t'\ldots)$, где t' - единственный его корневой операнд, в который входит переменная x', идентифицированная с x. Компилятор вставляет в программу приема оператор "внешфункция(...)", выполняющий данную идентификацию по заданным корневому вхождению идентифицирующего терма и переменной x'. Роль шаблона для построения новых термов вида "значение(f T)" играет стандартный информационный элемент (функция f x ...), которому передается модифицированный идентифицирующий терм: вместо t' у него размещена переменная x'.

Обработка функциональной переменной "значениемн(f t)" начинается после контрольной точки "прием(269)". Эта переменная выделяется указателем "значениемн". Идентифицирующий терм представляет собой сумму (быть может, вырожденную), которая рассматривается как значение многочлена f в точке t (либо на наборе t). По мере возможности, многочлен вводится над кольцом целых чисел, иначе - над полем вещественных. Напомним, что многочлены представляются в решателе при помощи выражений "многочлен(X R K)", где X - набор переменных многочлена, R - кольцо, над которым рассматривается многочлен, K - функция, сопоставляющая наборам степеней (или степеням) коэффициенты многочлена. Эта функция определена на некотором конечном множестве. Компилятор проверяет, что все переменные из t уже идентифицированы, строит программное выражение для t и вставляет в про-

грамму приема обращение к процедуре "форммногочлен". Последняя и осуществляет необходимую идентификацию, определяя для многочлена выражение указанного выше вида.

Идентификация в направлении к корню терма

При обработке установки на идентификацию (операнд v x) имеются две возможности - либо перейти к идентификации операндов операции по этому вхождению, либо перейти к идентификации той надоперации w, чьим операндом является v. Первый случай был частично рассмотрен выше - для тех подслучаев, когда идентификация операндов происходит однозначно или почти однозначно. Чтобы ускорить отсечение программой приема побочных ситуаций, идентификация с разбором случаев относится компилятором в конец программы, а в начале размещаются однозначно выполняемые шаги. К числу таких шагов, кроме уже перечисленных выше, относится также идентификация в направлении от текущего вхождения к корню идентифицируемого терма: переход от вхождения v к надоперации w делается однозначно и позволяет сразу же проанализировать заголовок по вхождению w.

Начальная точка ветви компилятора, обрабатывающей переход к надоперации w, относится к нулевому уровню срабатывания (x7 = 0). Из фрагмента, содержащего начало цикла сканирования установок на идентификацию, переход к ней выполняется через оператор "ветвь 5". Здесь размещается контрольная точка "прием(47)", выйти на которую можно также из пункта ("Установка на идентификацию (операнд ...)", "Идентификация в направлении к корню терма", "Исходная точка") оглавления программ. Напомним, что x10 - текущее идентифицируемое вхождение v, и оператор "операнд(x11 x10)" присваивает переменной x11 вхождение надоперации w. Далее проверяется, что w относится к идентифицируемой части теоремы. Если это не так, причем v - корневое вхождение идентифицируемой части, а прием имеет указатель "теквхожд(...)", то в программу добавляется оператор, проверяющий, что соответствующее идентифицирующее вхождение - тоже корневое. При отсутствии указателя "теквхожд(...)" данный оператор вводится другими фрагментами компилятора.

Пусть w относится к идентифицируемой части теоремы приема. Проверяется, что оно еще не зарегистрировано в списке x8 идентифицированных вхождений, после чего заносится в данный список. Далее идентификация в направлении к корню терма распадается на ряд перечисляемых ниже подслучаев.

Переход через внешнюю операцию, выделенную указателем "развертка"

После контрольной точки "прием(274)" рассматривается подслучай, когда w имеет вид "отображение(i и(целое(i) меньшеилиравно(1 i) меньшеилиравно(i n))$P(i)$)" либо вид "длялюбого(i если принадлежит(i номера(1 ... n))то $P(i)$)", причем в первом случае непосредственный надтерм w, а во втором - сам терм w выделен указателем "развертка". В этом случае v есть корневое вхождение выражения либо утверждения $P(i)$ (согласно процедуре, выбирающей точку привязки). Переменной x13 присваивается: в первом случае - вхождение одноместной надоперации для w, во втором случае - само вхождение w. Если еще не все свободные переменные терма x13 идентифицированы, то обработка установки x9 откладывается. Иначе - переход через "ветвь 3". Переменной x15 здесь присваивается символ по вхождению x13. Указатель "развертка" означает, что теоремный терм x13 должен идентифицироваться с термом вида $P(1) * \ldots * P(n)$, где ассоциативно-коммутативная операция $*$ определяется по

символу x15 с помощью справочника "развертка", а в случае x15 = "длялюбого" представляет собой логическую связку "и". Эта операция присваивается переменной x16. Чтобы перейти к корню указанного идентифицирующего терма от вхождения v, в программу приема заносится оператор "операнд(начало(x3)конец(x9))", и далее - оператор "символ(начало(x3)x16)", фиксирующий наличие требуемого заголовка данного терма. В x17 запоминается программная переменная, значением которой служит корневое вхождение идентифицирующего терма для идентификации с "разверткой". Однако, предпринятая до этого момента идентификация изнутри $P(i)$ в направлении к корневому вхождению x17 являлась предварительной - она позволила идентифицировать лишь свободные переменные терма x13 путем рассмотрения какого-то одного операнда из списка $P(1), \ldots, P(n)$ и найти точку x17. Чтобы идентифицировать весь список операндов, будет выполняться независимая идентификация по специально создаваемой установке (развертка x13 набор(x17) 0). При этом все результаты предварительной идентификации внутри $P(i)$ сбрасываются - относящиеся к ней вхождения исключаются из списка x8 и из информационных элементов (операнд ...); удаляются информационные элементы, в которых упоминаются связанные переменные терма x13, и т.п. Затем - откат к началу цикла идентификации.

Переход через внешнюю функциональную переменную, выделенную указателем "символ"

После контрольной точки "прием(276)" рассматривается подслучай, в котором текущее вхождение v является корневым вхождением некоторого t_i из функциональной переменной "значение(f набор(t_1 ... t_n))", выделенной указателем "символ". Здесь рассматривается только случай $n > 1$. Тогда w - вхождение символа "набор". Корневое вхождение самой функциональной переменной присваивается x12. Идентифицирующий терм в этом случае должен иметь вид $F(T_1 \ \ldots \ T_n)$. Компилятор вводит в программу приема оператор, определяющий корневое вхождение идентифицирующего терма как внешней операции для T_i; оператор, определяющий символ F, расположенный по данному вхождению, а также оператор, проверяющий, что T_i есть i-й операнд. Вводится информационный элемент (переменная f ...), сопоставляющий теоремной переменной f программное выражение, значением которого служит F. Затем для каждого $j = 1, \ldots, n$, отличного от i, предпринимается обращение к процедуре "учетоперанда", регистрирующей сопоставление идентифицируемому вхождению t_j идентифицирующего вхождения T_j. По окончании этих действий - вводится дополнительная установка на идентификацию (операнд u ...), у которой u - корневое вхождение рассматриваемой функциональной переменной, и откат к началу цикла идентификации.

Переход через внешний квантор

При переходе через внешний квантор иногда бывает нужно учитывать возможность специальных преобразований этого квантора, допускаемых указателями приема.

После метки "прием(53)" рассматривается случай кванторной импликации; v есть вхождение ее антецедента либо консеквента с отброшенным отрицанием. Переменной x12 здесь присваивается корневое вхождение кванторной импликации, а переменной x13 - индикатор антецедента (0) либо консеквента (1), к которому относится v после устранения возможного внешнего отрицания путем контрапозиции. Если импликация не выделена указателем "внешнийквантор", то при идентификации нужно учи-

тывать возможность контрапозиций. Поэтому для перехода от идентифицирующего для v вхождения V к корневому вхождению импликации и проверки того, что индикатор антецедента - консеквента вхождения V равен x13, применяется специальная процедура "внешдлялюбого". Если прием имеет указатель "кванторнаясвертка" и рассматриваемая кванторная импликация - корневая, то вместо процедуры "внешдлялюбого" используется процедура "внешнийквантор", которая учитывает не только контрапозиции, но также возможность перехода от квантора существования к отрицанию квантора общности.

После метки "прием(54)" аналогичным образом рассматривается случай перехода через внешний квантор существования, причем только для приемов с указателем "кванторнаясвертка", где необходимо учитывать возможность перехода от квантора общности к отрицанию квантора существования.

Наконец, после метки "прием(55)" рассматривается еще одна особенность перехода через внешнюю кванторную импликацию - идентификация конъюнкции либо дизъюнкции при таком приеобразовании антецедентов и консеквента идентифицирующей импликации, которые давали бы эту конъюнкцию или дизъюнкцию в явном виде. Допустимыми преобразованиями здесь считаются контрапозиция, группировка части антецедентов под общую конъюнкцию и спуск отрицаний. v является вхождением конъюнктивного либо дизъюнктивного члена идентифицируемой конъюнкции либо дизъюнкции u, возможно, с отброшенным внешним отрицанием. Чтобы найти корневое вхождение идентифицирующего терма - внешней конъюнкции, либо дизъюнкции, либо кванторной импликации, - компилятор вставляет в программу оператор "внешконъюнкция(...)" либо "внешдизъюнкция(...)". Создается дополнительная установка на идентификацию (операнд u ...).

Во всех перечисленных выше случаях перехода через внешний квантор далее происходит откат к началу цикла идентификации, с предварительным отбрасыванием текущей установки x9, если все операнды вхождения v уже идентифицированы.

Учет указателя "заменазнака" при переходе к внешней операции

После контрольной точки "прием(56)" рассматривается случай, когда идентифицируемое вхождение v выделено указателем "заменазнака". Тогда для него ранее был создан информационный элемент (заменазнака v f x g A), означающий, что если соответствующее v идентифицирующее вхождение V является операндом внешней одноместной операции f, то эта внешняя операция переносится на терм, идентифицируемый с переменной x, заменяясь при этом на одноместную операцию g. A - сначала 0, а затем программное выражение для вхождения U, полученного переходом от V к внешней операции f, если таковая имеется. Если A равно 0, то компилятор вставляет в программу приема оператор, присваивающий первой неиспользуемой программной переменной вхождение U. При этом A заменяется на данную программную переменную y; она же в информационном элементе (операнд v ...) становится ссылкой на идентифицирующее вхождение для v.

Заметим, что несколько ранее переменной x12 была присвоена пара программных переменных, первая из которых еще не определена, а вторая - имеет своим значением идентифицирующее вхождение для v. Если A было только что определено указанным выше образом, то второй элемент пары x12 заменяется на A. Это же происходит, если ненулевое значение A уже было зарегистрировано в информационном элементе (заменазнака ...). Далее - продолжение процесса обработки надоперации w, с переходом через "ветвь 1" к контрольной точке "прием(57)".

После контрольной точки "прием(56)" рассматривается еще один особый подслучай; переход к его обработке происходит через "иначе 2". Здесь указатель "заменазнака" выделяет не вхождение v, а вхождение его надоперации w, имеющее вид $f(\ldots)$. Это означает, что при наличии надоперации f вхождения v символ g не будет навешиваться на терм, идентифицируемый с x, а при ее отсутствии - будет. Фактически в данной точке компилятор лишь вставляет в программу приема оператор, присваивающий первой неиспользуемой программной переменной y уже упоминавшееся выше вхождение U, регистрируя эту переменную в информационных элементах (операнд w y) и (заменазнака w f x g y). Кроме того, он вводит новую установку на идентификацию (операнд w y) - для последующей обработки процедуры перенесения "знака" f. В данном подслучае происходит откат к началу цикла идентификации, с предварительным отбрасыванием текущей установки x9, если все операнды вхождения v уже идентифицированы.

Общий случай перехода через внешнюю операцию

Рассмотрение общего случая для перехода через внешнюю операцию начинается с контрольной точки "прием(57)". Как и выше, будем обозначать посредством v текущее идентифицируемое вхождение; V - соответствующее идентифицирующее вхождение; w - вхождение операции, операндом которой служит v. Прежде всего, компилятор вводит набор x13, образованный одним либо двумя операторами, определяющими идентифицирующее вхождение W, соответствующее w. Если операция w может оказаться вырожденной из-за наличия операндов, обращающихся в единицу, то в x13 вводится контроль наличия вырожденной ситуации - если она имеет место, то W совпадает с V, иначе V является операндом у W. Операторы набора x13 выполняют также проверку того, что на вхождении W располагается нужный символ. Чтобы обеспечить ее, компилятор обращается к процедуре "символвхождения", учитывающей различные указатели приема, определяющие допустимые модификации заголовков подтермов.

Если прием имеет указатель "знаксуммы(...)", выделяющий операцию w, то после контрольной точки "прием(58)" выполняется коррекция x13. Здесь рассматривается информационный элемент (знаксуммы s u A), созданный ранее по указателю "знаксуммы(...)". u есть вхождение некоторой операции, у которой при идентификации допускается одновременное изменение знака s всех ее операндов. Под изменением знака понимается отбрасывание одноместной операции s там, где она была, и ввод этой операции там, где ее не было. A - сначала 0, а впоследствии становится программной переменной, играющей роль индикатора изменения знака. Если знак был изменен, то значение индикатора равно 1, иначе - 0. Компилятор проверяет, что u имеет своим операндом v либо $s(v)$. В первом случае это означает, что u совпадает с w, во втором случае - что w является операндом операции u. Затем вводится набор операторов x17, на который заменяется в рассматриваемом подслучае набор x13. В качестве W берется идентифицирующее вхождение для u. Информационный элемент (операнд v ...) корректируется с учетом возможного изменения знака. Далее - откат к переходу через "ветвь 1" для продолжения рассмотрения общего случая.

Если w должно идентифицироваться с корневым вхождением терма, рассматриваемого в контексте применения приема, то к списку операторов x13 добавляется оператор "корень(...)", проверяющий это условие. Затем x13 регистрируется в программе приема и вводится информационный элемент (операнд w ...), определяющий идентификацию вхождения w. К списку установок на идентификацию

добавляется установка (операнд w ...).

Далее компилятор вводит операторы, уточняющие номер операнда V операции W. Если идентифицируемое выражение w - описатель "класс" либо "отображение", то v должно быть вхождением последнего либо предпоследнего его операнда. В этом случае компилятор вводит оператор, проверяющий соответствующее расположение для V,W. Если имеется информационный элемент (дробь A ...), указывающий, что операнды двуместной некоммутативной операции w могут быть переставлены синхронно с операциями, перечисленными в списке A, то предпринимается учет данной перестановки. Если w - первая из операций списка A, причем v - первый операнд для w, то в информационном элементе (дробь ...) регистрируются программные выражения для V,W. Они используются далее как индикатор перестановки: если перестановки не было, то V есть первый операнд для W, иначе - второй. Если до w были рассмотрены другие операции списка A и индикатор перестановки уже имелся, то в программу приема вводятся операторы, проверяющие размещение V относительно W в соответствии с этим индикатором. Наконец, при отсутствии указателя "дробь", выделяющего некоммутативную операцию w, вводится оператор, проверяющий, что номер операнда V для W совпадает с номером операнда v для w.

По окоанчании перечисленных действий - откат к началу цикла идентификации, быть может, с предварительным исключением текущей установки на идентификацию x9.

Идентификация ассоциативного и коммутативного символа

Перечисленные выше случаи исчерпывают приемы обработки установок на идентификацию, срабатывающие при нулевом значении текущего уровня x7. Переходя к рассмотрению приемов более высоких уровней, выделим прежде всего наиболее сложную ветвь процедуры "идентификатор", связанную с идентификацией операндов ассоциативного и коммутативного символа. Эта ветвь начинается с контрольной точки "прием(63)", достижимой, например, из пункта ("Установка на идентификацию (операнд ...)", "Коммутативный и асоциативный символ", "Исходная точка") оглавления программ. Контрольной точке предшествуют проверки коммутативности и ассоциативности символа по текущему идентифицируемому вхождению v. К числу коммутативных символов здесь относится также символ "конкатенация", если его внешняя одноместная операция выделена указателем "список". После контрольной точки "прием(63)" начинается обработка различных подслучаев идентификации. Заметим, что приемы компилятора, обрабатывающие эти подслучаи, вообще говоря, относятся к различным значениям текущего уровня x7. Перейдем к перечислению данных подслучаев.

Идентификация при наличии указателя "смодночлен"

После контрольной точки "прием(263)" рассматривается случай, когда идентифицируемая операция имеет вид ax^n и должна идентифицироваться как одночлен от заданной переменной x. Этот случай (встречающийся крайне редко) определяется указателем "смодночлен", по которому вводится информационный элемент (смодночлен v x). Ассоциативной и коммутативной операцией здесь является операция "умножение". Компилятор находит программные выражения x15 и x16, соответственно, для степени n и для переменной x. Если ранее проведенная идентификация пока не позволяет определить данные выражения, то применение приема компиля-

ции откладывается. В противном случае проверяется, что a есть еще не идентифицированная переменная, и в программу компилируемого приема заносится обращение к процедуре "смодночлен", которая по заданным идентифицирующему вхождению V и переменной X, определяемой выражением x16, находит коэффициент A и степень N. Добавляется также оператор, проверяющий совпадение N и x15. Вводится информационный элемент (терм a ...), определяющий идентификацию коэффициента a. После этого установка на идентификацию x9 удаляется; текущий уровень x7 сбрасывается до 0, и откат к началу цикла идентификации.

Идентификация при наличии указателя "коэффициент"

После контрольной точки "прием(219)" рассматривается случай, когда идентифицируется сумма, у которой некоторое слагаемое выделено указателем "коэффициент". Такое слагаемое имеет вид ax либо ax/b, причем переменная x идентифицируется после определения идентифицирующих выражений A, B для a, b как сумма всех коэффициентов k в слагаемых, которые можно представить в виде Ak либо, соответственно, в виде Ak/B. Перемножение степеней с одинаковыми основаниями в этих представлениях не допускается. По указателю "коэффициент" был введен информационный элемент (коэффициент v x t), где t - либо a, либо a/b. Компилятор присваивает переменной x16 идентифицирующий терм для t, находит остаток еще не идентифицированных слагаемых рассматриваемой суммы, и для определения x вставляет в программу компилируемого приема обращение к процедуре "коэффициент". Если все остальные слагаемые идентифицируемой суммы уже были обработаны ранее, то добавляется оператор, проверяющий, что возникающий после данного обращения остаток слагаемых идентифицирующей суммы пуст. Затем - замена x7 на 0 и откат к началу цикла идентификации.

Идентификация при наличии указателя "набор(первыйтерм)"

Если прием имеет указатель "набор(первыйтерм)", то до обращения к процедуре "идентификатор" был создан информационный элемент (набор u 0), у которого u - вхождение в теорему разгруппируемой приемом ассоциативно-коммутативной операции. Если u совпадает с текущим рассматриваемым вхождением v, то у элемента (набор u 0) последний разряд заменяется на программное выражение, определяющее идентифицирующее для v вхождение V. Этой информации уже достаточно для последующего программирования процесса разгруппировки, и компилятор удаляет установку на идентификацию x9. Текущий уровень x7 заменяется на 0, и откат к началу цикла идентификации.

Идентификация операндов ассоциативно-коммутативной операции, представляющих собой переменные либо функциональные переменные

Перечисляемые в данном подразделе приемы компиляции срабатывают при текущем уровне x7, равном 1. Ветвь этих приемов начинается после контрольной точки "прием(65)". Просматриваются вхождения x11 операндов текущей операции v. Как и выше, будем обозначать посредством V идентифицирующее вхождение, соответствующее вхождению v.

Начиная с контрольной точки "прием(66)" рассматривается случай, когда по вхождению x11 расположена некоторая переменная x. Эта переменная присваивается

x12. Если имеется информационный элемент (знаксуммы s v A) либо информационный элемент (заменазнака u s x r A), у которого x11 расположено внутри u, то проверяется, что программное выражение A, которое будет использоваться в нижеследующих приемах компиляции, уже определилось. Далее рассматриваются следующие подслучаи:

1. Имеется информационный элемент (второйоперанд v x) (контрольная точка "прием(67)"). Это означает, что x должно идентифицироваться с отдельным операндом операции V. Если уже создан элемент (набороперандов v A), у которого программное выражение A определяет список еще не идентифицированных операндов операции V, то компилятор вводит в программу приема оператор "входит(y A)", выбирающий для x идентифицирующее вхождение y. При этом A корректируется: вхождение y из него исключается. Если элемент (набороперандов v A) отсутствует, причем логический символ по вхождению v определен неоднозначно из-за возможной вырожденности рассматриваемой ассоциативно-коммутативной операции, то элемент (набороперандов v ...) вводится, и далее - как выше. В остальных случаях компилятор вводит в программу приема группу операторов, перечисляющих вхождения y операндов операции V, отличные от ранее использованных. Во всех случаях создается информационный элемент (терм x ...) либо (вхождение x ...), идентифицирующий переменную x. Значение x7 заменяется на 0, и откат к началу цикла идентификации.

2. Имеется информационный элемент (переменная x A), указывающий, что переменная x должна идентифицироваться с переменной либо логическим символом, определяемыми программным выражением A (контрольная точка "прием(68)"). Если уже был создан информационный элемент (набороперандов v B), то вводится оператор, проверяющий наличие символа A среди еще не идентифицированных операндов B операции V. При этом элемент (набороперандов v B) корректируется: из B исключается однобуквенный терм A. При отсутствии элемента (набороперандов ...) проверка наличия символа A среди не идентифицированных операндов операции V реализуется с использованием информационных элементов (операнд ...).

3. Имеется информационный элемент (неизвестная x), указывающий, что переменная x должна идентифицироваться с неизвестной текущей задачи (контрольная точка "прием(69)"). Если уже был создан информационный элемент (набороперандов v B), то компилятор вводит операторы, отбирающие в списке B терм, являющийся неизвестной. При компиляции приема пакетного оператора список неизвестных извлекается из комментария (неизвестные ...), вводимого перед обращением к этому оператору. Если информационного элемента (набороперандов v ...) нет, то компилятор вводит операторы, отбирающие еще не идентифицированный операнд операции v, представляющий собой неизвестную. Последующие приемы рассматриваемой ветви компилятора применяются лишь в том случае, если отсутствует информационный элемент (неизвестная y), относящийся к еще не идентифицированному операнду y операции v. Таким образом обеспечивается приоритетное рассмотрение ситуаций, уменьшающих перебор при идентификации операндов ассоциативно-коммутативной операции.

4. Переменная x уже была идентифицирована ранее - в виде вхождения, либо терма, либо набора операндов текущей ассоциативно-коммутативной операции

f (контрольная точка "прием(70)"). Переменной x13 присваивается информационный элемент, определяющий значение переменной x. Если вхождение v имеет вид $f(y\ x)$, где переменная y выделена указателем "группировка", то обработка операнда x отменяется, так как этот операнд будет учтен в процессе группировок, определяющих y. В противном случае - переход через "ветвь 2", где инициализируется нулем переменная x14. Этой переменной далее переприсваивается программное выражение, определяющее набор f - операндов терма, идентифицированного с x.

После переприсвоения - переход через "ветвь 1". Здесь осуществляются коррекции выражения x14. Если имеется информационный элемент (знаксуммы $s\ v\ k$), то в термах набора, являющегося значением выражения x14, предпринимается коррекция знака s согласно индикатору замены знака k, введенному ранее при рассмотрении операндов операции v. Если имеется информационный элемент (заменазнака $A\ s\ x\ r\ B$), у которого вхождение x11 переменной x расположено внутри вхождения A, то x14 корректируется так, чтобы в нем был учтен переносимый на x внешний знак s.

Далее находится информационный элемент (набороперандов $v\ R$), у которого программное выражение R имеет своим значением список всех еще не идентифицированных операндов операции v; если такого элемента не было, то он предварительно создается. Затем компиляция разветвляется.

Если справочник "пересечениесписков" указывает по текущей ассоциативно-коммутативной операции f специальную процедуру P, используемую для усмотрения представления терма t_1 в виде $f(t_2\ X)$ по заданным t_1, t_2, причем нет указателей, отменяющих применение данного справочника, то компилятор вставляет в программу приема обращение к процедуре P, где в качестве t_1 выступает результат применения операции f к термам набора R, а в качестве t_2 -результат применения ее к термам набора x14. Таким образом, из остатка неиспользованных f - операндов идентифицирующего терма исключаются операнды, соответствующие переменной x. Если операция v имеет какие-то другие еще не идентифицированные операнды, то корректируется R. Иначе - вместо обращения к процедуре P вставляется обращение к процедуре "равнозначны", проверяющей равенство термов t_1, t_2.

Если обращение к справочнику "пересечениесписков" не используется, то компилятор вставляет в программу приема оператор, проверяющий включение набора x14 в набор R. Если других еще не идентиицированных операндов у v нет, то добавляется оператор, проверяющий равенство длин этих наборов. Корректируется R.

В обоих случаях далее - откат к началу цикла идентификации.

5. Переменная x уже была идентифицирована ранее - в виде набора операндов ассоциативно-коммутативной операции g, отличной от текущей ассоциативно-коммутативной операции f (контрольная точка "прием(71)"). Случай обрабатывается компилятором аналогично предыдущему; отличие состоит лишь в том, что переменной x14 присваивается не набор f - операндов терма, идентифицированного с x, а сам этот терм.

6. Существует фильтр приема "контекст(вид($x\ A)B$)", определяющий заголовок терма, идентифицируемого с переменной x (контрольная точка "прием(72)").

Проверяется отсутствие указателей приема, которые могли бы разрешить отличие данного заголовка от заголовка x16 терма A. После этого переменная x идентифицируется с произвольным еще не использованным операндом идентифицирующего терма, имеющим заголовок x16. Отдельно рассматриваются два случая - наличие либо отсутствие элемента (набороперандов v R). В каждом из них особо рассматривается подслучай, когда имеется указатель "знаксуммы"; тогда перед определением заголовка операнда предпринимается необходимая коррекция знака.

7. Существует фильтр приема, определяющий, что переменная x должна быть идентифицирована с термом, являющимся десятичной записью числа (контрольная точка "прием(73)"). Этот фильтр имеет один из следующих типов: "десчисло(x)"; "целое(x)"; "натуральное(x)". Рассматриваются два подслучая.

   Первый из них характеризуется следующими требованиями: заголовок идентифицирующего терма совпадает с f; переменная x не выделена указателем "заменазнака" и указателем "единица" для вырожденного значения 0; информационный элемент (набороперандов ...), перечисляющий еще не использованные операнды идентифицирующего терма, не введен.

   Если либо f отлично от умножения, либо переменная x не выделена указателем "единица" для вырожденного значения 1, то компилятор вводит в программу приема операторы, идентифицирующие x с произвольным еще не использованным операндом идентифицирующего терма, представляющим собой десятичную запись числа.

   Если f есть "умножение" и переменная x выделена указателем "единица" для вырожденного значения 1, то используется несколько более сложная конструкция. Компилятор вводит операторы, инициализирующие единичное значение x и фиктивное нулевое значение для вхождения операнда, идентифицирующего для x. После них размещается оператор, выполняющий просмотр еще не использованных сомножителей идентифицирующего терма, представляющих собой десятичные записи чисел, и корректирующий значения программных переменных для x и идентифицирующего вхождения.

   Второй подслучай относится к идентификации целочисленных констант (см. переход через "иначе 1"). В этом случае идентифицируемое выражение имеет вид nx, где n - целочисленная константа. Компилятор вводит операторы, проверяющие, что идентифицирующий для v терм представляет собой десятичную запись целого числа m, делящегося на n, и определяющие значение x как частного от деления m на n.

8. Переменная x является последним еще не идентифицированным операндом и идентифицируется по остаточному принципу (контрольная точка "прием(74)").

   Сначала выделяется специальный подслучай - наличие указателя "группировка", выделяющего переменную x (контрольная точка "прием(26)"). В этом подслучае v имеет вид $f(x\ A)$ и является опрандом некоторой внешней ассоциативно-коммутативной операции g, причем A уже идентифицировано. x группируется из всех операндов надоперации g, имеющих заголовок f, в которых выделяется подмножество операндов, образующее A. Переменной x15 присваивается информационный элемент, определяющий идентификацию A (как вхождения либо как переменной); x16 присваивается вхождение надоперации g. Находится

информационный элемент (набороперандов x16 ...), задающий список еще не использованных операндов идентифицирующего терма для надоперации. Если такого элемента нет, то он водится. Переменной x19 присваивается список операторов компилируемой программы, которые будут определять набор f-членов терма, идентифицированного с A, а переменной x20 - программная переменная для этого набора. Наконец, определяется список x21 операторов компилируемой программы, которые осуществляют извлечение g- членов терма для x, формируют этот терм и находят модифицированный набор для элемента (набороперандов x16 ...). Здесь особо рассматривается случай выделения переменной x указателем "заменазнака".

Если указанный выше специальный случай отсутствует, то после контрольной точки "прием(74)" выполняется переход через "ветвь 2" для рассмотрения общего случая. Здесь прежде всего проверяется, что операция v не имеет еще не идентифицированного отличного от x11 операнда, выделенного указателем "группировка". При наличии указателя "знаксуммы", выделяющего вхождение v, проверяется, что либо вхождение v не является корневым вхождением заменяемого терма, либо x является переменной-модификатором, введенной в теорему приема для идентификации с остатком операндов ассоциативно-коммутативной операции. Если последние условия не выполнены, то идентификация переменной x по остаточному принципу отменяется - она должна быть идентифицирована с отдельным операндом.

Далее компиляция разветвляется в зависимости от того, имеется ли информационный элемент (набороперандов A). Если такой элемент есть, то A есть набор f - членов идентифицирующего терма t для x, с точностью до коррекции знака, выполняемой при наличии указателя "заменазнака". В случае, когда указатели приема не допускают возможности формрования t по пустому набору A, в программу приема добавляется проверка непустоты этого набора. Если элемента (набороперандов A) нет, то x формируется из всех операндов идентифицирующего для v терма, не указанных в информационных элементах (операнд ...). При этом предпринимается учет указателей "единица" и "заменазнака".

9. Переменная x идентифицируется путем группировки всех операндов операции V, удовлетворяющих условию P, заданному в указателе "перечень(x $P(x)$)" (контрольная точка "прием(75)"). Проверяется, что все отличные от x свободные переменные фильтра $P(x)$ уже идентифицированы. Если элемента (набороперандов v A) еще не было, то он вводится. Далее создается вспомогательный программный блок x17. В него отбираются копии всех информационных элементов текущего программного блока, заголовки которых имеются в списке x16. Единственный фрагмент программы блока x17 состоит из оператора вида "принадлежит(y A)", перечисляющего еще не использованные операнды идентифицирующего вхождения V, причем добавлен информационный элемент (терм x y), указывающий, что x будет идентифицироваться с y. Находится результат x18 компиляции фильтра $P(x)$ относительно программного блока x17. При добавлении конъюнктивных членов оператора x18 к программе блока x17 получается фрагмент, перечисляющий все термы y набора A, удовлетворяющие условию $P(y)$. По этому фрагменту строится вспомогательный оператор x19, присваивающий первой неиспользованной переменной z текущего программного блока x3 выписку всех допустимых y. x19 регистрируется в компилируемой программе; для идентификации x вводится программное выражение x20, зна-

чение которого равно результату объединения под операцией f всех элементов набора z. Попутно учитывается, если он есть, указатель "заменазнака".

Далее рассматриваем случаи, относящиеся к идентификации функциональной переменной, расположенной по вхождению x11. Началом этого рассмотрения служит контрольная точка "прием(329)" (пункт оглавления программ "Идентификация функциональной переменной - исходная точка"). Ассоциативно-коммутативную операцию по вхождению v обозначаем f. Функциональную переменную далее обозначаем $g(t)$, причем переменной x12 присвоена переменная g. Рассматриваются следующие подслучаи:

1. $g(t)$ является элементом уже идентифицированной матрицы g (контрольная точка "прием(246)"). Эта ситуация распознается по наличию информационного элемента (элементыматрицы g A m n), операторные выражения A, m, n которого позволяют найти операторное выражение x13 для терма T, идентифицированного с $g(t)$. Затем выполняются действия, аналогичные случаю уже идентифицированной обычной переменной. Находится информационный элемент (набороперандов v B). Чтобы установить возможность "выделения" терма T в терме, полученном применением операции f к остаточным операндам набора B, используется, если это возможно, справочник "пересечениесписков". Иначе просто устанавливается включение в B списка f-членов терма T.

2. Функциональная переменная $g(t)$ выделена указателем "операнд" и должна идентифицироваться с отдельным операндом идентифицирующей операции V (контрольная точка "прием(84)"). Ситуация распознается по наличию информационного элемента (второйоперанд v g). Если имеется информационный элемент (набороперандов v A), то $g(t)$ идентифицируется с произвольным элементом набора A. Если информационного элемента (набороперандов v A) нет, причем допускается случай отличия заголовка идентифицирующей операции V от f, то этот элемент вводится, и далее - как выше. Иначе $g(t)$ идентифицируется с вхождением произвольного еще не использованного операнда операции V. Заметим, что для регистрации результата идентификации вхождения функциональной переменной во всех подслучаях здесь используется процедура "учетоперанда".

3. Ранее была идентифицирована функциональная переменная вида $g(x)$ (контрольная точка "прием(76)"). Тогда имеется информационный элемент (функция g x A B) с ненулевым B. В данном пункте компилятора предполагается, что у идентифицируемого терма $g(t)$ выражение t является переменной. B содержит информацию о терме G, определяющем значение функции $g(x)$. Если B - пара (вхождение s), то s есть программное выражение, определяющее корневое вхождение терма G; если B - пара (терм s), то s есть программное выражение, значением которого служит сам терм G. Переменной x15 сначала присваивается программное выражение для G. Если t отлично от x, то x15 корректируется таким образом, чтобы получилось программное выражение для терма, сопоставляемого $g(t)$ - путем подстановки t вместо x в G. Дальнейшая обработка данного случая - стандартная (см., например, случай функциональной переменной - элемента идентифицированной матрицы).

4. Функциональная переменная $g(t)$ идентифицируется путем группировки всех операндов операции V, удовлетворяющих условию P, заданному в указателе "перечень(g $P(g)$)" (контрольная точка "прием(147)"). Идентифицирующий терм определяется здесь так же, как в случае обычной переменной, однако регистрация его происходит с использованием оператора "учетоперанда".

5. Функциональная переменная является последним еще не идентифицированным операндом и идентифицируется по остаточному принципу (контрольная точка "прием(229)"). Идентифицирующий терм определяется по информационному элементу (набороперандов v) и регистрируется с использованием оператора "учетоперанда".

Кроме перечисленных выше случаев идентификации операнда, представляющего собой переменную (обычную либо функциональную), в данной ветви компилятора рассматривается случай идентификации операнда x11, выделенного указателем "подтерм" (контрольная точка "прием(333)"). На момент идентификации все переменные теоремного терма t, расположенного по вхождению x11, должны быть определены. Это позволяет сначала построить для t соответствующий терм T контекста срабатывания приема, а затем проверить его совпадение с идентифицирующим термом. Наличие указателя "подтерм" распознается по информационному элементу (стандтерм ...).

Идентификация операндов ассоциативно-коммутативной операции в общем случае

В этом подразделе собраны приемы идентификации операндов ассоциативно-коммутативной операции, выделяемых по какому-либо характерному признаку (например, по заголовку). Некоторые из них относятся к переменным, но имеют меньший приоритет, чем приемы идентификации переменных, рассмотренные выше. Это выражается в уровне срабатывания - для приемов данного подраздела он равен 2 либо 5. Рассматриваемая ветвь компилятора начинается с контрольной точки "прием(77)". Прежде всего, выбирается текущий обрабатываемый операнд x11 идентифицируемой операции v. Сначала разбирается случай, когда этот операнд отличен от переменной, т.е. по вхождению x11 расположен некоторый логический символ s. Сразу заметим, что случай операнда-переменной относится только к указателю "отрицание", когда при идентификации переменная x и терм вида $s(x)$ (где $s(s(x)) = x$) выступают равноправным образом. Таким образом, даже в случае переменной здесь возникает некоторый "скрытый" заголовок s.

После проверки того, что по вхождению x11 расположен логический символ, идет группа проверок целесообразности обработки операнда x11 на данном уровне x7. Такая обработка на меньшем уровне x7 = 2 допускается лишь если число операндов операции v, имеющих заголовок s, не более трех, причем число уже идентифицированных операндов этой операции менее двух. Эти эвристические ограничения направлены на уменьшение перебора, выполняемого компилируемым приемом. Далее, требуется, чтобы в первую очередь в качестве x11 выбирался тот из еще не идентифицированных операндов операции v, у которого заголовок идентифицирующего терма определяется однозначно. Следующий приоритет отбора x11 - рассмотрение константного терма либо такого терма, идентификация для которого выполняется с помощью справочника "вид". Этот справочник используется в решателе пока лишь для идентификации выражений, представимых в виде натуральной степени,

т.е. для усмотрения полного квадрата, куба, и т.п. При рассмотрении алгебраических выражений обнаружение таких множителей является достаточно характерным признаком, ускоряющим отсечение непригодных для срабатывания приема ситуаций. Поэтому оно и выносится компилятором ближе к началу приема. Наконец, заблокировано использование данной ветви компиляции в тех случах, когда указатели приема явно определяют идентификацию x11 с первым операндом идентифицирующего терма. Далее рассматриваются следующие подслучаи:

1. Операнд x11 выделен указателем "развертка", причем все свободные переменные этого операнда уже идентифицированы (контрольная точка "прием(166)"). С помощью процедуры "прогрвыражение" находится программное выражение, значением которого служит набор термов, определяемый теоремным выражением x11. В программу приема вставляется оператор, проверяющий, что этот набор включается в набор еще не использованных операндов идентифицирующего терма. Если x11 - последний не идентифицированный операнд операции v, то добавляется проверка равенства числа элементов в этих наборах.

2. Операнд x11 представляет собой функциональную переменную $g(t)$, идентификация которой должна происходить либо с первым операндом идентифицирующей операции V (указатель "первыйтерм"), либо с термом, группируемым из всех операндов V, содержащих переменные терма t. Последний случай реализуется по умолчанию, если ни один из оставшихся не идентифицированными операндов операции v (кроме x11) не зависит от переменных, входящих в t. Данная ветвь компилятора начинается с контрольной точки "прием(78)". Переменной x14 присваивается либо список программных выражений для переменных терма t (тогда x13 равно 0), либо одноэлементный список, состоящий из программного выражения для набора таких переменных (тогда x13 равно 1). Данный прием компиляции применяется при уровне x7, равном 5.

   Оставшиеся приемы рассматриваемой ветви компилятора учитывают при отборе обрабатываемого операнда x11 приоритет, согласно которому предпочтение отдается константному терму либо терму, хотя бы один из корневых операндов которого отличен от переменной. Кроме того, обработка x11 блокируется в ряде специальных случаев (выделенный указателем "группировка" смежный операнд надоперации операции v, либо наличие еще не идентифицированной переменной, которая, согласно указателям приема, должна быть идентифицирована раньше x11). С учетом этих ограничений, далее рассматриваются следующие подслучаи:

3. Операнд x11 выделен указателем "подстановка", т.е. фактически может отсутствовать, и тогда определенная в нем переменная x будет идентифицироваться с заданным логическим символом s (контрольная точка "прием(79)"). x присваивается программной переменной x13. Затем формируется список x15 всех переменных y операнда x11, которые встречаются в идентифицируемой части теоремы приема только внутри подтермов, выделенных указателями "подстановка". Момент фактической идентификации такой переменной y зависит от того, какие из содержащих ее подтермов будут вырожденными, причем в программу приема далее будет введен оператор, проверяющий, что y в конце концов оказалась идентифицирована. Проверяется, что на текущий момент

все переменные операнда x11, отличные от x и не входящие в список x15, уже идентифицированы. Иначе - переход к следующему операнду x11. Далее компиляция разветвляется в зависимости от того, введен ли уже информационный элемент (набороперандов v ...) или нет. Обе ветви аналогичны; ограничимся рассмотрением случая, когда данный элемент отсутствует. Сначала вводится оператор, инициализирующий первую неиспользуемую программную переменную z, которой будет присвоен идентифицирующий терм для x, однобуквенным термом s. Здесь же инициализируется нулем следующая программная переменная (обозначим ее u); этой переменной будет присвоено вхождение идентифицирующего операнда для x11, если таковой найдется. Переменной x16 присваивается набор той же длины, что и x15. Для тех переменных y списка x15, которые к текущему моменту идентифицированы, в x16 размещается соответствующее программное выражение; для прочих переменных y списка x15 - вводится новая программная переменная, которая инициализируется нулем, а впоследствии ей будет присвоен идентифицирующий терм, соответствующий y. Далее создается вспомогательный программный блок x20. В него заносится единственный фрагмент программы, обеспечивающий просмотр еще не использованных операндов x14 идентифицирующего вхождения V (с учетом необходимого для идентификации x11 заголовка). Предпринимается обращение к процедуре "идентификатор", обеспечивающее создание в блоке x20 программы идентификации x11 с x14. Если в фильтрах приема имеются условия, относящиеся только к переменной x, то они компилируются и тоже заносятся в программу блока x20. Для блокировки их повторной обработки вводятся информационные элементы (комментарии ...). Переменной x22 присваивается список всех новых программных переменных, добавленных в блоке x20. В рамках блока x20 определяются программные выражения для переменной x и для всех переменных списка x15. После этого создается оператор x25 для занесения в программу основного программного блока x3, который обеспечивает цикл просмотра операндов x14, попыток их идентификации с x11, а также коррекции значений переменных z, u и новых программных переменных для ранее не идентифицированных переменных списка x15. Этот оператор имеет вид кванторной импликации, связывающей приставкой которой служит список x22, антецедентом - программа блока x20, а консеквентом - операторы, выполняющие коррекции значений переменных. Наконец, для тех переменных y списка x15, у которых x11 было последним содержащим y не идентифицированным подтермом, выделенным указателем "подстановка", вводится оператор, проверяющий отличие от нуля введенной для их идентифицирующего терма программной переменной.

4. Операнд x11 выделен указателем "нормзнака" (контрольная точка "прием(97)"). Тогда этот операнд имеет вид $g(x)$, где x - переменная, которая в рассматриваемом подслучае предполагается уже идентифицированной. x11 будет идентифицироваться с набором g - отрицаний f - членов идентифицирующего терма для x. Программной переменной x12 присваивается переменная x, а x13 - набор f - членов ее идентифицирующего терма. Вводится программное выражение x14 для набора g - отрицаний элементов набора x13. Далее используется информационный элемент (набороперандов v A), определяющий список A еще не использованных операндов идентифицирующего вхождения V. В программу приема вводится проверка включения набора x14 в A (если x11 - последний не идентифицированный операнд у v, то также проверка равенства длин этих

наборов). Корректируются информационные элементы (набороперандов v A) и (нормзнака x11 ...).

5. Операнд x11 содержит единственную переменную x, причем справочник "повторение" указывает на возможность его представления в виде $f(x \ldots x)$, где f - рассматриваемая коммутативно-ассоциативная операция по вхождению v (контрольная точка "прием(80)"). Этот же справочник определяет число n повторений переменной x в таком представлении. Данный прием идентификации пока используется только для обработки константных натуральных степеней, которые можно представить в виде произведений выражения, идентифицированного с x. Перед применением приема идентификации проверяется ряд ограничений (в частности, отсутствие указателей, блокирующих применение справочника "пересечениесписков"). Программной переменной x16 присваивается набор f - членов идентифицирующего терма для x. Находится информационный элемент (набороперандов v A), дающий список A еще не использованных операндов идентифицирующего вхождения V. При помощи справочника "пересечениесписков" находится процедура P, которая будет применяться для выделения поднабора x16 в наборе A, и далее в программу приема вставляется n - кратное исключение поднабора x16 из A.

   После выполнения ряда проверок, отсекающих рассмотрение в данной ветви компилятора специальных случаев, обрабатываемых в других его ветвях, список приемов компиляции продолжается:

6. Операнд x11 выделен указателем "группировки" (контрольная точка "прием(81)"). Это означает, что он имеет вид $g(xy)$, где x - обычная либо функциональная переменная, уже идентифицированная с термом $f(t_1 \ldots t_n)$; $n \geq 1$, а переменная y должна идентифицироваться с таким термом r, что все термы $g(t_1\ r), \ldots, g(t_n\ r)$ содержатся среди еще не использованных операндов идентифицирующего вхождения V. Здесь f, g - коммутативно-ассоциативные операции, причем f расположена по вхождению V. Прежде всего, программной переменной x14 присваивается набор $(t_1, \ldots, t_n)$. Затем - переход через "ветвь 3", где программной переменной x16 присваивается переменная y. Находится информационный элемент (набороперандов v A), определяющий список A еще не использованных операндов идентифицирующего вхождения V. С помощью справочника "пересечениесписков", если применение его в данной ситуации не заблокировано указателями приема, находится заголовок P той процедуры, которая будет использоваться для усмотрения представимости операнда в виде $g(t_i\ r)$. Символ P присваивается переменной x18 (если специальной процедуры нет, то x18 равно 0). Далее переменной x19 присваивается набор g - членов идентифицирующего терма r переменной y. Если эта переменная еще не идентифицирована, то для определения r используется t_1 и какой-либо операнд списка A, представимый в виде $g(t_1\ r)$. Данный операнд присваивается переменной x20 (изначально обнуленной). Это нужно для того, чтобы впоследствии исключить из рассмотрения уже обработанный терм t_1. Наконец, в программу приема вводится кванторная импликация, осуществляющая просмотр еще не обработанных термов t_i и проверку для каждого из них наличия операнда списка A, представимого как $g(t_i\ r)$. Здесь отдельно рассматриваются два подслучая - наличие специальной процедуры P (ненулевое x18) и ее отсутствие.

7. Операнд x11 - константный, причем не указано, что он должен идентифицироваться с явным вхождением логического символа. Если в этом случае еще не веден информационный элемент (набороперандов v ...) для остатка операндов идентифицирующего терма, то он вводится (контрольная точка "прием(82)").

   Далее приемы компиляции рассматриваемой ветви распадаются на две группы - для подслучая, когда информационный элемент (набороперандов v ...) уже имеется и для подслучая, когда его еще нет. Начнем с рассмотрения первого подслучая.

8. Операнд x11 - константный (контрольная точка "прием(86)"). Если этот операнд - не последний из еще не идентифицированных операндов вхождения v, причем допустимо использование справочника "пересечениесписков" для получения специальной процедуры P, выделяющей x11 среди еще не использованных операндов идентифицирующего вхождения V, то в программу приема вставляется обращение к процедуре P. Корректируется список не использованных операндов V. Если все операнды, кроме x11, уже идентифицированы, то проверяется, что не использован единственный операнд V, который равен x11. В случае указателя "знаксуммы" предпринимается необходимая коррекция знака.

9. Операнд x11 относится к группе вхождений, выделенных указателем "общаястепень", причем все остальные операнды вхождения v уже идентифицированы (контрольная точка "прием(87)"). Выделенные данным указателем вхождения $w_1, \ldots, w_n$ имеют вид двуместных операций, один операнд которых общий (некоторая переменная x), а прочие операнды суть различные и более нигде не встречающиеся в идентифицируемой части теоремы переменные $y_1, \ldots, y_n$ (некоторые из этих переменных могут быть функциональными). Выделение общей части x идентифицирующих термов $t_1, \ldots, t_n$ вхождений $w_1, \ldots, w_n$ предпринимается вспомогательной процедурой P, ссылка на которую содержится в указателе. Информационный элемент (общаястепень P x $(y_1, \ldots, y_n)$ $(w_1, \ldots, w_n)$ $(A_1 \ldots A_n)$) накапливает программные выражения A_i для термов t_i. В указанной выше ситуации компилятор определяет A_i для терма t_i, соответствующего x11, формируя этот терм из остатка неиспользованных операндов идентифицирующего вхождения V.

10. Операнд x11 выделен указателем "группировка" (контрольная точка "прием(25)"). Тогда этот операнд имеет вид $g(x\ t)$, причем переменная x идентифицируется как $f(X_1 \ldots X_n)$, где $X_1, \ldots, X_n$ получаются путем выделения всех операндов $B_1, \ldots, B_n$ идентифицирующего терма для V, представимых в виде $g(X_1\ T), \ldots, g(X_n\ T)$. Здесь T - идентифицирующий терм для t; f - операция по вхождению v; $n \geq 1$. Компилятор распознает данную ситуацию по наличию информационного элемента (логарифмы x11 ...). Переменной x16 присваивается символ операции g; проверяется, что эта операция ассоциативная и коммутативная. Переменной x18 присваивается переменная x. Проверяется, что все переменные терма t уже идентифицированы. Если программное выражение A для набора не использованных операндов идентифицирующего терма V отлично от переменной, то для этого набора вводится вспомогательная переменная. Далее значением переменной x14 является выражение A, возможно, модифицированное указанным образом. Переменной x20 присваивается программное

выражение для терма T. В программу приема вводится оператор, присваивающий первой неиспользуемой программной переменной набор R g - членов терма T; эта программная переменная сохраняется в x21. Далее создается список операторов x22, которые обеспечивают просмотр не использованных операндов идентифицирующего терма V, представимых в виде $g(X_i\ T)$, и группировку из них значения $f(X_1 \ldots X_n)$. Сначала переменной x22 присваивается 0, а затем ей переприсваивается необходимое значение, причем рассматриваются два случая: переменная x либо выделена указателем "заменазнака" (тогда предпринимается учет знаков операндов V и коррекция знаков у X_i), либо не выделена. После регистрации в программе операторов списка x22 проверяется, остались ли не идентифицированные операнды вхождения v; если не остались, то добавляется оператор, проверяющий пустоту списка не использованных операндов вхождения V.

11. Общий случай идентификации операнда x11 (контрольная точка "прием(99)"). В этом случае x11 идентифицируется с произвольным не использованным операндом A идентифицирующего вхождения V, и результат идентификации регистрируется процедурой "учетоперанда". Эта же процедура обеспечивает анализ заголовка операнда A. В особых случаях для идентификации x11 используется не один операнд вхождения V, а вся оставшаяся не идентифицированной группа операндов, которые соединяются расположенной по вхождению v операцией f. Например, это происходит, если x11 - последний не идентифицированный операнд, и справочник "вид" предусматривает специальную процедуру его идентификации. Данный справочник пока применяется лишь при компиляции распознавания в одночлене заданной натуральной степени другого одночлена (полного квадрата, куба и т.п.).

    На этом ветвь компилятора, отосящаяся к подслучаю, в котором информационный элемент (набороперандов v) уже введен, заканчивается. Ветвь, относящаяся к альтернативному подслучаю, начинается с контрольной точки "прием(90)". Здесь расположены следующие приемы компиляции:

12. v - вхождение конъюнкции либо дизъюнкции, для которого имеется информационный элемент (длялюбого v), задающий необходимость учета возможности выделения v в наборе антецедентов кванторной импликации (контрольная точка "прием(91)"). Здесь компилятор вставляет в программу приема обращение к процедуре "конъюнктоперанд" (в случае конъюнкции) либо "дизъюнктоперанд" (в случае дизъюнкции). Эти процедуры перечисляют как операнды конъюнкции (соответственно, дизъюнкции), так и антецеденты кванторной импликации (учтена возможность контрапозиции). Отбираются только операнды с заданным заголовком.

13. Общий случай идентификации операнда v (контрольная точка "прием(100)"). В накопителе x13 формируется оператор, перечисляющий вхождения операндов идентифицирующего вхождения V; эти вхождения становятся значениями переменной x12. Если имелся указатель, определяющий идентификацию x11 с первым операндом V, то вместо этого в x13 заносится оператор присвоения x12 вхождения данного первого операнда. Если операция V может оказаться вырожденной, то x13 корректируется: учитывается вырожденный случай, в котором переменной x12 присваивается само V. Далее накопитель x13 регистрируется в программе приема и добавляется группа операторов, проверяющих

отличие выделенного вхождения x12 от ранее использованных. Для регистрации идентификации операнда x11 используется процедура "учетоперанда". Если не идентифицированный операнд x11 вхождения v был последним, то при необходимости вводится проверка равенства чисел операндов вхождений v, V.

Идентификация операнда коммутативно-ассоциативной операции, выделенного указателем "развертка"

Этот прием компиляции располагается начиная с контрольной точки "прием(211)". Уровень его срабатывания x7 достаточно большой - он равен 6, т.е. применению данного приема предшествуют исчерпание описанных выше приемов. Ситуация распознается компилятором по наличию информационного элемента (развертка A), в списке A которого встречается терм t, размещенный по вхождению v. Терм t представляет собой одноместную операцию F над конечным семейством элементов, которую нужно идентифицировать с термом вида $f(x_1 \dots x_n)$, причем $x_1, \dots x_n$ идентифицируется с данным конечным семейством. Здесь f - коммутативно-ассоциативная операция по вхождению v, обобщением которой на конечные семейства служит F (например, f может быть операцией $+$, а F - $\sum$). Кроме того, t может иметь вид кванторной импликации $\forall_i(i \in 1, \dots, n \to P(i))$, идентифицируемой с конъюнкцией $P(1)$ & ... & $P(n)$. Компилятор определяет по F символ f с помощью справочника "развертка", присваивая его переменной x13. В случае кванторной импликации этой переменной присваивается символ "и". Фактическая идентификация x11 будет выполняться другими приемами компилятора; здесь же лишь формируется необходимая для их срабатывания установка на идентификацию (развертка x11 R 2), где R - информационный элемент (набороперандов v B), определяющий список еще не использованных операндов, из которого будут извлекаться элементы конечного семейства. Если текущая установка на идентификацию x2 оказывается исчерпанной (x11 - последний не идентифицированный операнд v, а надоперация v уже идентифицирована), то она удаляется. Затем - откат к началу цикла идентификации.

Особые случаи идентификации операнда ассоциативно-коммутативной операции

После контрольной точки "прием(102)" собраны остальные приемы компилятора, обеспечивающие идентификацию операнда x11 ассоциативно-коммутативной операции. Наименьший уровень x7 срабатывания таких приемов равен 4. Все они относятся только к идентификации переменных - простых либо функциональных, причем первоочередное применение их (см. рассмотренную выше ветвь идентификации переменных, относящуюся к "малому" уровню срабатывания 1) нецелесообразно. В частности, здесь находится прием идентификации переменной, встречающейся в нескольких различных одноименных операциях, как пересечения операндов этих операций (обобщенный с учетом справочника "пересечениеписков").

Начнем с рассмотрения случая, в котором по вхождению x11 расположена обычная переменная x12. Здесь выделяются следующие подслучаи:

1. Переменная x12 идентифицируется с числовой константой (контрольная точка "прием(103)"). Этот прием компиляции дополняет аналогичный прием, применявшийся на уровне 1 (там его применение сопровождалось рядом ограничений, которые здесь снимаются). Если операция по вхождению v есть умножение, то

при отсутствии неиспользованного сомножителя идентифицирующего вхождения V, представляющего собой десятичную константу, x12 идентифицируется с единицей. Иначе x12 идентифицируется с произвольным не использованным операндом вхождения V, являющимся десятичной константой. При компиляции отдельно рассматриваются подслучаи наличия элемента (набороперандов v) и его отсутствия. Предпринимается учет указателя "заменазнака", выделяющего x12.

Для дальнейших приемов компиляции составляется список x13 всех вхождений переменной x12 в идентифицируемую часть теоремы. Если имеется указатель "пересечение", специально выделяющий те подтермы, из рассмотрения которых должна происходить идентификация x12, то x13 заменяется на набор вхождений x12 в эти подтермы. После контрольной точки "прием(105)" начинается рассмотрение случаев, в которых список x13 имеет более одного элемента.

2. Переменная x12 имеет несколько идентифицируемых вхождений; хотя бы одно из них является операндом коммутативно-ассоциативной операции u, идентифицируемой с указателем "знаксуммы"; элемент (набороперандов u ...) не введен, причем обращение к процедуре "идентификатор" произошло из компиляции фильтра "контекст"(контрольная точка "прием(106)"). Это - редкий особый случай: встречающееся в фильтре выражение t предметного уровня выделено указателем "знаксуммы", означающим, что возможна идентификация t как с изменением знаков всех операндов, согласно указателю, так и без изменения знаков. При этом для остальной части фильтра (и приема в целом) несущественно, имело ли место изменение знаков или нет. Соответственно, компилятор вставляет в программу фильтра дизъюнктивный оператор, сначала присваивающий некоторой программной переменной y набор корневых операндов t, а затем набор этих же операндов с измененными знаками. Для корневого вхождения w подтерма t вводится информационный элемент (набороперандов w y); элемент (знаксуммы ...) удаляется, и откат к началу цикла идентификации.

   Далее проверяется, что если переменная x12 выделена указателем "заменазнака", то информационные элементы (заменазнака ...), соответствующие точкам перенесения знака на вхождения списка x13, уже содержат программные выражения для этих точек.

3. Все вхождения списка x13 суть вхождения операндов подтермов, имеющих один и тот же коммутативно - ассоциативный заголовок f либо преобразуемых к такому виду путем тождественного преобразования, состоящего в изменении знака всех корневых операндов и вынесении этого знака наружу (контрольная точка "прием(107)"). Этот случай распадается на два подслучая:

   (a) Все вхождения списка x13 относятся к различным вхождениям своих надопераций f (контрольная точка "прием(109)"). Прежде всего, здесь проверяется отсутствие такой не идентифицированной переменной y, которая являлась бы операндом всех надопераций вхождений списка x13, а также еще каких-то операций в идентифицируемой части теоремы. Если такая переменная y имеется, то приоритетной является ее идентификация, а обработка переменной x12 обрывается. Таким образом, в первую очередь идентифицируются те переменные, которые определяются как "общая часть" возможно большего числа операций f.

Далее - переход через "ветвь 3", где анализируется случай выделения рассматриваемых в списке x13 вхождений переменной x12 указателями "заменазнака". Чтобы идентификация переменной x12 как общей части соответствующих f - операций была корректной, необходимо соответствие переносимых на x12 внешних одноместных операций s этих f-операций, имеющихся у идентифицируемых термов, таким операциям у идентифицирующих термов (с точностью до повсеместной замены знака s). Проверка такого соответствия обеспечивается вводимыми компилятором фильтрами x20: все элементы списка x13, начиная со второго, сравниваются с первым элементом.

После создания группы фильтров x20 - откат к переходу через "ветвь 1", где инициализируется нулем переменная x14 и вводится пустой накопитель x15. В процессе просмотра элементов x16 списка x13 этот накопитель заполняется наборами операндов надопераций вхождений x16. Значение x14 формируется в цикле как "общая часть" всех таких наборов. При заполнении списка x15 учитываются два случая: надтерм вхождения x16 имеет вид $f(...x16...)$, либо вид $f(...s(x16)...)$, где знак s может быть изменен одновременно у всех операндов и вынесен за операцию f. Во втором случае знаки элементов набора, заносимого в список x15, заменяются на противоположные. Собственно коррекция значения x14 на очередном шаге цикла выполняется, начиная с перехода через "ветвь 2" (в том же фрагменте, где x14 было инициализировано). Для первого элемента списка x13 переменной x14 присваивается только что зарегистрированный в накопителе x15 набор. Для всех последующих - реализуется переход через "иначе 1".

Здесь переменной x18 присваивается либо 0, либо название P той процедуры для нахождения "общей части" двух выражений, которая определяется справочником "пересечениесписков". Переменной x19 присваивается текущий (последний занесенный в список x15) набор f- операндов. Если имеется указатель "выписка(...)", выделяющий специальные группы операндов, участвующие в идентификации переменной x12, то x19 корректируется - в нем сохраняются только допустимые согласно этому указателю операнды. Особо рассматривается случай ненулевого x18 и длины набора x13, равной 2 - здесь процедура P сразу определяет не только "общую часть" двух выражений, но и их остатки, что позволяет за один шаг провести идентификацию x12 и сохранить данные остатки в информационных элементах (набороперандов ...). Если такой случай имеет место, то после его обработки происходит откат к началу цикла идентификации, иначе - по окончании цикла просмотра элементов x16 списка x13 выполняется переход через "иначе 1". Здесь реализуется цикл определения остатков наборов операндов надопераций f вхождений списка x13 после исключения из них "общей части", идентифицируемой с x12. Эти остатки регистрируются в информационных элементах (набороперандов ...). По окончании цикла регистрируется результат идентификации переменной x12 (при необходимости учитывается указатель "заменазнака").

(b) Существуют различные вхождения списка x13, имеющие одно и то же вхождение надоперации f (контрольная точка "прием(110)"). Тогда выбирается какая-то пара различных вхождений v_1, v_2 списка x13, представляющих собой непосредственные операнды одного и того же вхождения w

операции f, и переменная x12 идентифицируется с термом, встречающимся среди неиспользованных операндов идентифицирующего для w вхождения W хотя бы дважды. Отдельно рассматриваются подслучаи наличия информационного элемента (набороперандов w ...) и его отсутствия.

4. Все вхождения списка x13 суть вхождения операндов коммутативно - ассоциативных операций, однако не все эти операции имеют один и тот же заголовок (контрольная точка "прием(108)").

   Проверяется, что для всех вхождений w коммутативно - ассоциативных операций, имеющих операнд из списка x13, уже введены установки на идентификацию (операнд w ...) - тогда определены и программные выражения для их идентифицирующих вхождений. Чтобы идентифицировать переменную x12, используется первый элемент списка x13, являющийся операндом некоторой операции $f(\ldots)$, а также какой-то элемент x16 этого списка, являющийся операндом операции $g(\ldots)$ с другим заголовком. Вхождения данных операций в теорему приема присваиваются переменным x14 и x17 соответственно. Идентифицирующий терм для x12 может либо быть отдельным операндом операции $f(\ldots)$, либо быть образован группой таких операндов. Но в последнем случае он будет иметь известный заголовок f, и его можно найти среди операндов операции $g(\ldots)$, имеющих этот заголовок. Переменная x20 представляет собой накопитель, в который заносится пара операторов: первый просматривает неиспользованные операнды идентифицирующего терма для операции f, второй - неиспользованные операнды идентифицирующего терма операции g, имеющие заголовок f. Отдельно здесь рассматриваются случаи наличия информационного элемента (набороперандов ...) для неиспользованных операндов операции и отсутствия такого элемента. В последнем случае учитываются возможности наличия указателей "заменазнака" и "знаксуммы". Собственно идентификация переменной x12 выполняется оператором x21 - дизъюнкцией операторов пары x20. После того, как x12 идентифицирована по паре элементов набора x13, начинается обработка всего этого набора. Прежде всего, переменной x23 присваивается список всех символов f_i коммутативно - ассоциативных операций, имеющих операнд из x13. Затем вводится и в цикле просмотра списка x13 заполняется накопитель x24 наборов f_i - членов терма t, идентифированного с x12. В этом же цикле происходит проверка корректности идентификации переменной x12 - наборы f_i - членов терма t должны представлять собой часть наборов неиспользованных операндов идентифицирующих термов для операций $f_i(\ldots)$. Здесь "часть" понимается, в зависимости от указателей приема и символа операции f_i, либо в смысле процедуры, определяемой справочником "пересечениесписков", либо как обычное теоретико-множественное включение.

5. Длина списка x13 равна 1, т.е. переменная x12 имеет единственное вхождение в идентифицируемой части (контрольная точка "прием(111)"). Уровень срабатывания приема компиляции, обрабатывающего данный случай, равен 6. Он имеет "остаточный" характер - применяется по исчерпании прочих средств идентификации и идентифицирует переменную x12 с произвольным еще не использованным операндом вхождения V, соответствующего v. Особо выделен подслучай, когда кроме x12 не идентифицирован лишь единственный операнд вхождения v, представляющий собой функциональную переменную, зависящую от переменных внешних связывающих приставок, в то время как x12 не входит в эти

приставки. Тогда x12 идентифицируется путем группировки всех неиспользованных операндов вхождения V, не пересекающихся с соответствующими связывающими приставками.

Среди приемов, обрабатывающих особые случаи идентификации операнда коммутативно - ассоциативной операции, упомянем также прием идентификации функциональной переменной (контрольная точка "прием(112)"), срабатывающий на седьмом уровне. Если эта переменная оказывается единственным не идентифицированным операндом рассматриваемой коммутативно - ассоциативной операции, то она идентифицируется по остаточному принципу. Если остаются не идентифицированы ровно два операнда, представляющие собой различные функциональные переменные, то первый из них идентифицируется с произвольным неиспользованным операндом идентифицирующего вхождения, а второй - с остатком этих операндов.

Идентификация коммутативного, но не ассоциативного символа

Этот раздел процедуры "идентификатор" начинается с контрольной точки "прием(48)" и легко может быть найден через оглавление программ. Его приемы срабатывают при значении текущего уровня x7, равном 2. Логический символ f по текущему идентифицируемому вхождению v допускает произвольные перестановки своих операндов, но не ассоциативен. При этом отсутствует информационный элемент (коммутативно v), означающий, что перестановки операндов v при идентификации запрещены. В основном, здесь обрабатываются символы симметричных двуместных отношений. Рассматриваются следующие случаи:

1. f - логическая связка "эквивалентно", причем вхождение v выделено указателем "Отрицание" (контрольная точка "прием(227)"). Это означает, что при идентификации эквивалентности допускается одновременный переход к отрицаниям обеих ее частей. Данный случай встречается достаточно редко - в приемах вывода теорем. Компилятор вставляет оператор "Операнды(...)", присваивающий двум первым неиспользуемым программным переменным a, b вхождения операндов v - сначала в прямом, а затем в обратном порядке. Таким образом учитывается коммутативность. Чтобы учесть указатель "Отрицание", после этого поочередно двум следующим программным переменным c, d присваиваются сначала вхождения термов a, b, а затем корневые вхождения отрицаний этих термов. Полученные таким образом идентифицирующие вхождения c, d для первого и второго операндов вхождения v регистрируются с помощью обращений к процедуре "учетоперанда".

2. f - двуместная операция, имеющая своим операндом переменную x, выделенную указателем "единица" (контрольная точка "прием(49)"). Переменной x12 присваивается список программных выражений для вхождений уже идентифицированных операндов вхождения v. Затем просматриваются еще не идентифицированные вхождения x13 операндов v. Если x13 - вхождение операнда x, то вводится оператор "альтернатива(...)", присваивающий некоторой программной переменной z произвольный операнд идентифицирующего для v вхождения V, если по вхождению V находится символ операции f, а в противном случае присваивающий этой переменной единицу (однобуквенный терм, согласно указателю "единица"). Если же x13 - вхождение отличного от x операнда, то вводится оператор "альтернатива(...)", отличающийся от первого лишь тем,

что во втором подслучае переменной z присваивается подтерм по вхождению V. Далее идентифицирующий для вхождения x13 терм z регистрируется с помощью процедуры "учетоперанда".

3. Операнды двуместной операции f - переменные, относительно которых теорема приема симметрична (контрольная точка "прием(50)"). Симметричность теоремы приема относительно данных переменных y, z проверяется путем перестановки их в теореме и проверки того, что новая и старая версия теоремы после применения к ним процедуры "стандупорядочение" совпадают. Проверяется также, что переменные y, z еще не идентифицированы. Ввиду симметричности, нет необходимости рассматривать при идентификации два случая: без перестановки и с перестановкой операндов идентифицирующего вхождения. Здесь переменная y идентифицируется с первым операндом, а z - со вторым.

   Для оставшихся в рассматриваемой ветви компилятора приемов создается список x11 операндов идентифицируемого вхождения v, в котором сначала располагаются операнды, отличные от переменных, а затем - переменные. После этого не идентифицированные элементы x12 списка x11 просматриваются по порядку, и к ним применяются перечисляемые ниже приемы.

4. v есть вхождение антецедента, выделенного указателем "равно" (контрольная точка "прием(165)"). Проверяется, что x12 - последний не идентифицированный операнд вхождения v. Находится информационный элемент (операнд ...), определяющий программное выражение для уже идентифицированного операнда вхождения v. Переменной x16 присваивается программное выражение для списка посылок, соответствующего контексту выполняемых преобразований. Затем в программу компилируемого приема заносится обращение к оператору "транзитоперанд", перечисляющему возможные вхождения для идентификации x12. Регистрация результата идентификации x12 выполняется процедурой "учетоперанда".

5. Общий случай идентификации коммутативного, но не ассоциативного символа (контрольная точка "прием(51)"). Здесь x12 идентифицируется с произвольным еще не использованным операндом идентифицирующего вхождения. Регистрация выполняется с помощью процедуры "учетоперанда".

Идентификация символа, обладающего специальной симметрией

Рассмотрение данного случая начинается с контрольной точки "прием(62)". По вхождению v (переменная x10) расположен логический символ f, для которого справочник "схемаоперандов" определяет древовидную схему x11 допустимых перестановок операндов. Каждая вершина схемы x11 соответствует некоторому множеству операндов операции f, определяет разбиение его на подмножества и указывает, допустимы ли произвольные перестановки этих подмножеств при идентификации. Она кодируется набором $(A\ B_1\ \ldots\ B_n)$, у которого $B_1, \ldots, B_n$ суть либо вершины схемы, соответствующие подмножествам операндов, либо, в случае одноэлементного подмножества, номера операндов. Для ссылок на операнды используются символьные номера, начиная с 1. A - указатель допустимости перестановки подмножеств: 1 - допустима, 0 - не допустима. Заметим, что коммутативные операции не учитываются в справочнике "схемаоперандов"; он нужен только для случаев специальной симметрии.

Например, такая симметрия встречается у выражения "угол(ABC)", где допустима перестановка A и C, в то время как B имеет фиксированную вторую позицию.

После оператора "повторение" начинается цикл обработки операндов вхождения v. Прежде всего, создается информационный элемент (схемаоперандов v A_1 A_2) (см. переход через "иначе 2"). У этого элемента A_1 представляет собой результат замены в древовидной схеме x11 всех номеров операндов операции v вхождениями этих операндов в теорему приема. A_2 - накопитель, в котором по мере идентификации регистрируются тройки (1 C_1 C_2), такие, что C_1 - либо набор из древовидной схемы A_1, либо вхождение операнда операции v; C_2 - программное выражение, значением которого служит объект, с которым идентифицируется C_1. Первый элемент 1 введен из технических соображений - чтобы можно было использовать оператор "биключ". Если C_1 - набор, определяющий древовидное подразбиение некоторого множества операндов операции v, то значением C_2 служит соответствующая древовидная конструкция, образованная операндами идентифицирующего вхождения. В ней лишь отброшены первые элементы - индикаторы допустимости перестановок. Если же C_1 - вхождение операнда, то значением C_2 служит вхождение соответствующего идентифицирующего операнда. При допустимости перестановок ветвей корневого набора схемы A_1 накопитель A_2 инициализируется единственным элементом, соответствующим корневому набору этой схемы. Иначе A_2 образуется из всех наборов и вхождений, на которые ссылается корневой набор.

Далее начинается цикл обработки неидентифицированных операндов x13 вхождения v. Древовидная структура A_1 (переменная x14) просматривается по принципу "сначала вглубь" до обнаружения вхождения x13. При просмотре используется стэк x15, образованный тройками (D_1 D_2 D_3), где D_1 - набор дерева A_1; D_2 - номер выделенного разряда в D_1, который соответствует предыдущему элементу стэка (нумерация начинается с 0); D_3 - вхождение этого разряда. Новые тройки заносятся в начало стэка. После нахождения ветви, ведущей в дереве A_1 к x13, начинается перемещение по ней от корня дерева. При этом в A_2 заносятся элементы, соответствующие проходимым вершинам. Здесь учитываются указатели допустимости перестановок: либо берется ветвь поддерева с заданным номером, либо - произвольная ветвь, еще не использованная в A_2. По достижении концевой вершины определяется программное выражение для вхождения x13, которое регистрируется с помощью процедуры "учетоперанда".

Идентификация символа, допускающего циклические перестановки операндов либо изменение порядка операндов на противоположный

Такие символы часто встречаются в геометрических обозначениях (например, "квадрат($ABCD$)"). Соответствующая ветвь компилятора легко находится через оглавление программ.

Сначала рассматривается особый случай - вхождение v выделено указателем "сдвиг" (контрольная точка "прием(143)"). Переменной x11 присваивается заблаговременно созданный по указателю информационный элемент (сдвиг v A t 0). Здесь A - список символьных номеров операндов вхождения v, внутри которого происходят циклические перестановки; t - терм, при формировании которого требуется синхронная перестановка корневых операндов. Если по вхождению v располагается символ "набор", то вставляется оператор, проверяющий совпадение количеств корневых операндов в идентифицируемом и идентифицирующем наборах. Затем вводится обращение к процедуре "сдвиг", перечисляющей наборы w операндов идентифицирующего

для v вхождения V, получающиеся при всевозможных перестановках указанного типа. Кроме набора операндов, эта процедура выдает величину циклического сдвига, причем программная переменная x для нее регистрируется в информационном элементе (сдвиг ...) вместо последнего нуля. Затем неидентифицированные операнды вхождения v идентифицируются с расположенными на соответствующих позициях элементами набора w.

Рассмотрение общего случая циклических перестановок начинается с контрольной точки "прием(161)". Здесь находится информационный элемент (циклупорядочение v A), присваиваемый переменной x11. Символьный номер A определяет тип допустимых перестановок операндов вхождения v. Если он равен 0, то допускаются произвольные циклические перестановки; если он равен 1, то - лишь одна циклическая перестановка (после нее на первом месте оказывается второй операнд); тип 2 означает, что допускается не циклическая перестановка, а лишь изменение порядка операндов на противоположный; тип 3 - что допускаются вообще произвольные (не обязательно циклические) перестановки операндов; тип 4 - допускаются произвольные циклические перестановки, быть может, сопровождаемые изменением порядка операндов на противоположный.

Сначала рассматривается случай $A = 2$, т.е. возможность изменения порядка операндов на противоположный. Компиляция предусмотрена лишь для числа операндов вхождения v, равного 3 либо 4. Создается дизъюнктивный оператор x15, который поочередно присваивает первым неиспользованным программным переменным операнды идентифицирующего для v вхождения V - сначала в прямом, а затем в обратном порядке. Этот оператор заносится в компилируемую программу, после чего результаты идентификации операндов регистрируются с помощью процедуры "учетоперанда".

Если A не равно 2, то переход через "иначе 2". Здесь проверяется, фиксировано ли число операндов операции по вхождению v. Если нет, то в программу приема заносится проверка совпадения чисел операндов идентифицируемого и идентифицирующего вхождений v, V. Далее - переход через "ветвь 1".

Проверяется, идентифицирован ли хотя бы один операнд вхождения v. Если нет, то выполняется определяющая циклический сдвиг идентификация первого операнда x12 вхождения v. При $A = 4$ в программу заносится оператор, поочередно присваивающий указателю u изменения порядка операндов значения 0 и 1. Этот указатель регистрируется в информационном элементе (подобны v u). При A отличном от 1 в качестве идентифицирующего для x12 берется произвольный операнд вхождения V; если A равно 1, то - только первый либо второй операнды.

Если уже имеется идентифицированный операнд вхождения v, то находится произвольный не идентифицированный операнд x12. Если A не равно 3, то проверяется, что операнд x13, циклически предшествующий операнду x12, уже идентифицирован с некоторым вхождением s. Тогда x12 идентифицируется путем циклического сдвига s; в случае $A = 4$ учитывается указатель изменения порядка. Если A равно 3, то x12 идентифицируется с произвольным еще не использованным операндом вхождения V. Здесь учитывается возможность выделения одного из операндов вхождения v указателем "пустоеслово". Такой операнд идентифицируется последним, по остаточному принципу (возможно, с единицей, соответствующей пустому остатку операндов).

Идентификация связывающей приставки, если в ней выделена переменная, идентифицируемая по остаточному принципу

Начало данной ветви компилятора - контрольная точка "прием(164)". Символ f по текущему вхождению v - один из связывающих символов "класс", "отображение", "существует", "длялюбого". Уровень x7 срабатывания приемов компилятора равен 2. Переменной x11 присваивается связывающая приставка вхождения v; в ней находится переменная x13, выделенная указателем "кортежпеременных". Такая переменная x13 будет идентифицироваться с некоторой группой переменных связывающей приставки идентифицирующего вхождения V. Реализуется цикл просмотра не идентифицированных вхождений x14 в связывающую приставку переменных x15, которые должны быть идентифицированы с единственной переменной. Они идентифицируются с произвольными не использованными элементами X идентифицирующей связывающей приставки, причем без учета порядка вхождения. Если переменная x15 уже была идентифицирована с некоторой переменной Y, то при этом проверяется равенство X и Y.

После цикла просмотра вхождений x14 - переход через "иначе 2". Здесь выполняется идентификация переменной x13. Сначала рассматривается случай, когда эта переменная выделена указателем "списокпеременных", определяющим ее идентификацию со списком всех переменных связывающей приставки вхождения V, имеющих свободные вхождения в заданном терме t теоремы приема. Переменной x19 присваивается программное выражение для этого списка, и идентификация x13 регистрируется в информационном элементе (наборпеременных x13 x19). Если же x13 не выделена указателем "списокпеременных", то она идентифицируется со списком всех не использованных переменных связывающей приставки вхождения V. В случае, когда x13 была ранее идентифицирована из других соображений с некоторым списком переменных, вставляется оператор, проверяющий совпадение обоих списков (без учета порядка их элементов).

Идентификация антецедента - функциональной переменной

Если функциональная переменная $f(t)$ является антецедентом кванторной импликации, связывающая приставка X которой содержит переменные y, не входящие в t, то применяется прием компиляции, идентифицирующий $f(t)$ с конъюнкцией всех еще не использованных антецедентов, не имеющих вхождений переменных, идентифицированных с какой-либо из указанных переменных y. Этот прием расположен после контрольной точки "прием(234)"; он применяется при текущем уровне x7, равном 3.

Идентификация ассоциативного, но не коммутативного символа

Данный случай обрабатывается начиная с контрольной точки "прием(168)". Вхождения, выделенные указателем приема, определяющим возможные циклические перестановки операндов, здесь не рассматриваются. Выделяются два подслучая:

1. Идентифицируемая операция имеет вид $f(t_1\ t_2)$ и представляет собой заменяемый терм теоремы приема, имеющей вид тождества (контрольная точка "прием(244)"). Тогда в идентифицирующей операции $f(A_1 \dots A_n)$ выбираются произвольные два идущих подряд операнда A_i, A_{i+1}; t_1 идентифицируется с первым из них, а t_2 - со вторым. Остатки операндов, идущие до и после пары A_i, A_j, регистрируются в информационном элементе (заменагруппы

...), который будет использован при синтезе преобразующей части приема: они образут не изменяемый контекст преобразования.

2. Общий случай. Последовательно просматриваются операнды t_i идентифицируемого терма $f(t_1 \dots t_n)$. Операнд t_1 идентифицируется с первым операндом A_1 идентифицирующей операции $f(A_1 \dots A_m)$. Каждый последующий операнд t_i, кроме t_n, идентифицируется как следующий за операндом A_{i-1}, сохраняемым в информационном элементе (ассоциативно ...). Оба эти подслучая разбираются после контрольной точки "прием(169)". Наконец, последний операнд t_n (контрольная точка "прием(170)") идентифицируется с термом $f(A_n \dots A_m)$.

Идентификация набора без учета порядка элементов

Если вхождение терма "набор($t_1 \dots t_n$)" выделено указателем "список", то операнды t_i идентифицируются с операндами A_j идентифицирующего терма "набор($A_1 \dots A_n$)" в произвольном порядке - каждый раз выбирается какой-либо еще не использованный операнд (контрольная точка "прием(174)").

На этом список приемов компилятора, обрабатывающих установку на идентификацию "операнд", завершается. Прежде, чем переходить к обработке установок других типов, анализируется наличие указателя "начало(...)", задающего первоочередное рассмотрение антецедентов. Если текущая установка на идентификацию x9 относится к антецеденту, не выделенному указателем "начало", в то время как имеется другая установка, относящаяся к выделенному антецеденту, то рассмотрение установки x9 блокируется (контрольная точка "прием(113)").

19.10.5 Установка на идентификацию "корень"

Если установка на идентификацию x9 имеет вид (корень v), то v есть вхождение антецедента теоремы приема (иногда - конъюнктивного члена заменяемой части теоремы), который должен быть идентифицирован с каким-либо утверждением из контекста точки привязки приема. Это может быть посылка или условие текущей задачи; посылка пакетного оператора; конъюнктивный член внешней по отношению к точке привязки конъюнкции, и т.п. Рассмотрение установки "корень" начинается с контрольной точки "прием(114)". Оно происходит при значениях 3 либо 8 текущего уровня x7. Если вхождение v выделено указателем приема "конец", то уровень 3 отменяется. Как сказано выше, v, вообще говоря, не обязательно является вхождением антецедента теоремы приема. Однако, для краткости в подзаголовках нижеследующих пунктов будем называть его антецедентом.

Переменной x10 присваивается вхождение v. Если вхождение v выделено указателем приема "подчинено" (информационный элемент (смпосылка ...)), определяющим список переменных, которые должны быть идентифицированы до обработки v, то проверяется, что все эти переменные уже найдены. Если прием имеет заголовок "параметризация" либо "заменаусловия(...)", то он может выполнять замену сразу нескольких условий задачи: в одном расположена точка привязки, а другие выделены установкой на идентификацию "корень". В этом случае обработка других установок "корень", относящихся к антецедентам теоремы, откладывается до завершения обработки первых. Далее переменной x11 присваивается символ f, расположенный по вхождению v, и переход к рассмотрению перечисляемых ниже случаев.

Прием с заголовком "свертка"

В этом случае теорема приема представляет собой кванторную импликацию, консеквент которой сам является кванторной импликацией вида $\forall_{x_1 \ldots x_n}(A_1 \ldots A_m \to A_0)$. Прием используется в задачах на доказательство: A_0 идентифицируется с условием задачи, а $A_1, \ldots, A_m$ - с частью посылок этой задачи. Приемы такого типа используются в решателе для реализации различных схем математической индукции. Точка привязки приема не обязательно размещается внутри A_0 - например, в приеме для доказательства индукцией по натуральному параметру она выбирается в антецеденте "натуральное(n)", и попытка применения приема начинается с усмотрения посылки "натуральное(...)".

Программа приема компиляции начинается с контрольной точки "прием(203)". Проверяется, что v есть вхождение консеквента либо антецедента внешней импликации; переменной x12 присваивается вхождение этой импликации. Если v - вхождение консеквента, то проверяется, расположена ли по данному вхождению функциональная переменная. Если нет, то вхождению v сопоставляется консеквент K идентифицирующей импликации, и для последующего их сравнения создается установка на идентификацию (операнд v ...). В случае функциональной переменной происходит непосредственная регистрация ее идентифицирующего терма K.

Если v - вхождение антецедента, то составляется список x13 программных выражений для уже использованных при идентификации посылок задачи. Затем проверяется наличие указателя "подстановка", выделяющего вхождение v. Если его нет, то вхождению v сопоставляется произвольная еще не использованная посылка задачи, и для последующего их сравнения вводится установка на идентификацию (операнд v ...).

Если v - выделенное указателем "подстановка" вхождение антецедента A_i, то это означает, что посылка задачи для данного антецедента может отсутствовать, причем тогда некоторой переменной m, входящей в A_i, присваивается заданная константа C. Например, такая ситуация возникает в простейшем приеме доказательства индукцией по натуральному параметру x, где предусмотрено наличие посылки, определяющей нижнюю границу m допустимых значений параметра x. При отсутствии данной посылки параметру m будет присвоена единица. Компилятор сначала вводит в программу приема операторы, присваивающие первым неиспользуемым программным переменным y, z, соответственно, константу C и ноль. Затем создается кванторная импликация x21, выполняющая просмотр неиспользованных посылок задачи, идентификацию их с подтермом v и коррекцию значений y, z в случае обнаружения подходящей посылки P, соответственно, на идентифицирующее значение для m и на терм P. Для получения такой кванторной импликации используется вспомогательный программный блок x17, в котором предпринимается идентификация v с текущей посылкой. После занесения в компилируемую программу оператора x21 вводится информационный элемент (терм m y), регистрирующий идентификацию переменной m.

Идентификация антецедента с переменным заголовком

Если антецедент имеет вид "значение(f набор($t_1 \ldots t_n$))" либо, при $n = 1$, вид "значение(f t_1)", причем переменная f выделена указателем "символ", то он идентифицируется с утверждением вида $F(T_1 \ldots T_n)$, где F - логический символ, сопоставляемый при идентификации переменной f. Прием компиляции для данного случая начинается с контрольной точки "прием(344)". Переменная f становится здесь значением

программной переменной x12. Если f уже идентифицирована и F определилось, то идентифицирующее утверждение P - произвольное утверждение из контекста применения приема, имеющее заголовок F. Иначе берется произвольное утверждение P из контекста применения приема, и по его заголовку F идентифицируется переменная f. В обоих случаях создается новая установка на идентификацию (операнд v ...), обеспечивающая дальнейшее сравнение v и P.

Идентификация антецедента, выделенного указателем "развертка"

Если антецедент имеет вид кванторной импликации $\forall_i(i \in 1 \ldots n \rightarrow A(i))$ и выделен указателем "развертка", то он идентифицируется с конъюнкцией утверждений контекста применения приема, представимых в виде $A(i)$. Ветвь процедур компиляции, обрабатывающих данный случай, размещается начиная с контрольной точки "прием(207)". Предварительно предпринимается обращение к оператору "учет-нормализаторов", вводящему на основе указателей нормализации информационные элементы (быстрпреобр ...).Эти элементы нужны для синтеза программных выражений, дающих обрабатываемые нормализаторами новые термы. Здесь предвосхищается возможный синтез таких термов при идентификации $A(i)$.

После того, как установлено, что рассматриваемый антецедент имеет указанный выше вид, переменной x17 оказывается присвоено выражение n. Далее рассматриваются два случая:

1. Все входящие в n переменные уже идентифицированы, так что для n удается построить программное выражение x18 (контрольная точка "прием(256)"). Переменной x19 присваивается терм $A(i)$, и находится список x20 всех входящих в этот терм не идентифицированных функциональных переменных вида $f(\ldots\ i\ \ldots)$. Обычная переменная f для такой функциональной переменной будет идентифицироваться с выражением вида "набор($t_1 \ldots t_n$)". Переменной x21 присваивается набор из нулей и единиц - указателей многоместных функциональных переменных списка x20 (1 - многоместная, 0 - одноместная). Проверяется, что все входящие в терм $A(i)$ переменные, отличные от i и не встречающиеся в наборе x20, уже идентифицированы. Если прием имеет указатель "внутрпосылка(...)", выделяющий еще один кванторный антецедент вида $\forall_i(i \in 1 \ldots n \rightarrow B(i))$, используемый для задания дополнительного условия $B(i)$ на отбираемые для идентификации с $A(i)$ утверждения, то одновременно проверяется, что идентифицированы все входящие в $B(i)$ переменные, отличные от i и не встречающиеся в наборе x20.

   Далее вводится пустой накопитель x22 для программных переменных, значениями которых будут становиться наборы термов $t_1, \ldots, t_n$, идентифицирующие переменные из x20. После контрольной точки "прием(209)" он заполняется программными переменными y (в количестве, равном длине набора x20), для инициализации которых в программу приема заносятся операторы "равно(y пустоеслово)".

   Если компилируемый прием имеет заголовок "заменатермов" либо "заменаусловия", т.е. обеспечивает замену группы посылок или условий, то рассматриваемая кванторная импликация может быть не антецедентом теоремы приема, а конъюнктивным членом его заменяемой части. Тогда она будет определять подмножество заменяемых утверждений. В этом случае инициализируется пустым набором новая программная переменная - накопитель отбираемых для

замены утверждений. Этот накопитель X регистрируется в информационном элементе (блокзамещения X).

Далее - переход через "ветвь 1", где находится результат x25 замены всех вхождений в $A(i)$ функциональных переменных $f(\ldots i \ldots)$ набора x20 на термы, получаемые вычеркиванием варьируемой переменной i. Если число аргументов такой функциональной переменной равнялось 1, то она превращается в обычную переменную. Это делается для того, чтобы получить из $A(i)$ терм, который фактически будет идентифицироваться с текущим просматриваемым утверждением контекста срабатывания приема.

После контрольной точки "прием(210)" вводится вспомогательный программный блок x26, в котором и будет обрабатываться "шаг цикла" - идентификация терма x25 с текущим утверждением контекста. В программу этого блока сразу заносится единственный фрагмент, состоящий из: оператора, перечисляющего утверждения U контекста срабатывания приема; оператора, проверяющего совпадение заголовка утверждения с заголовком терма $A(i)$, и оператора, присваивающего новой программной переменной корневое вхождение утверждения U. Перед обращением к процедуре "идентификатор" для блока x26 предпринимается следующая коррекция его информационных элементов:

(a) Для всех многоместных функциональных переменных f списка x20 добавляется информационный элемент (усмматрица f). Такие функциональные переменные, после фиксации i, зависят от прочих своих аргументов, и при обработке блока x26 будут идентифицироваться по ним в режиме "развертка". Добавленный элемент определит регистрацию результата промежуточной идентификации f в блоке x26 не как терма вида "набор($T_1 \ldots T_k$)", а как набора термов $T_1, \ldots, T_k$ (строки либо столбца матрицы).

(b) Если информационный элемент (развертка ...) программного блока x3 указывает на термы T, содержащие функциональные переменные списка x20, то в x26 регистрируется еще один такой элемент, в котором предпринято исключение аргумента i из указанных функциональных переменных. Этим достигается переадресация указателей режима "развертка" на модифицированный по сравнению с $A(i)$ терм x25. Информационные элементы (единица ...), (заменазнака ...), (заголовок ...) блока x26, указывающие на вхождения внутри $A(i)$, корректируются так, чтобы они указывали на соответствующие вхождения внутри терма x25.

(c) Пусть имеется информационный элемент (внутрпосылка P Q), ссылающийся на вхождение P выделенного указателем "идентификатор" антецедента теоремы приема - кванторной импликации вида $\forall_i(i \in 1 \ldots n \rightarrow B(i))$. Такая импликация вводит дополнительное условие $B(i)$ на утверждения, выбираемые из контекста срабатывания приема для идентификации их с x25. Чтобы согласовать обозначеня, находится результат B' исключения аргумента i в функциональных переменных списка x20, имеющихся в терме $B(i)$, и к списку x27 установок на идентификацию блока x26 добавляется установка "идентификатор", относящаяся к корневому вхождению терма B'. При обработке блока x26 это обеспечит компиляцию проверки условия B'. Относящиеся к подтермам $B(i)$ элементы (быстрпреобр ...) также корректируются для переадресации их терму B'.

Наконец, после контрольной точки "прием(275)" происходит обращение к процедуре "идентификатор", обрабатывающей программный блок x26. Она создает в нем операторы, идентифицирующие x25 с текущим просматриваемым утверждением контекста. Если имелись фильтры приема, зависящие от варьируемой переменной i, то они компилируются и добавляются в программу блока x26. Таким образом обеспечивается еще одна возможность ограничения списка просматриваемых утверждений контекста. Далее к программе блока x26 добавляются операторы, пополняющие на текущем шаге цикла накопители, перечисленные в списке x22, и вводится кванторный оператор x29, полученный навешиванием на программу блока x26 квантора существования по всем новым программным переменным, не использованным пока в блоке x3. Так как в рассматриваемом случае для числа n отбираемых утверждений имеется программное выражение x18, то с его помощью создается кванторный оператор x31, реализующий последовательное перечисление значений i от 1 до x18 и обращение для текущего i к оператору x29. В этом цикле и запоняются накопители списка x22, позволяющие в итоге идентифицировать переменные списка x20. Предусматривается случай возникновения чрезмерно громоздкого оператора x31 - тогда этот оператор и оператор x29 не используются, а цикл организуется путем размещения программы блока x26 после операторов, перечисляющих значения i. Выход из цикла осуществляется при помощи оператора "ветвь", вставляемого перед перечислением значений i.

2. Длина n набора $(A_1, \ldots, A_n)$ перед идентификацией неизвестна. Случай аналогичен предыдущему, и хотя реализующая его программа выделена в независимую ветвь, обе ветви почти идентичны. Даже номера программных переменных, используемых для рассматриваемых при компиляции объектов - x20, x21, x22, x23, x24, x26 и т.п, - сохраняются без изменений. Переменная n идентифицируется как длина набора, найденного для первой переменной списка x20.

Идентификация антецедента, представляющего собой функциональную переменную

В этом случае по вхождению v расположено выражение $f(y)$, где f - переменная, выделенная указателем "отображение". Проверяется, что для нее уже введен информационный элемент (функция f X 0 r), где r - пара (вхождение A) либо (терм A), определяющая программное выражение A для терма, дающего значения функции f. X - набор теоремных переменных, рассматриваемых в качестве аргументов f. Проверяется также, что y - обычная переменная. Далее рассматриваются следующие подслучаи:

1. Имеется информационный элемент (результподст y B). Этот подслучай возникает при компиляции приемов вывода теорем, использующих указатель "унификация". y - теоремная переменная, обозначающая подставляемые при унификации термы; B - программное выражение для набора этих термов. Набор переменных X в данной ситуации одноэлементный; его единственная переменная x обозначает список переменных S, вместо которых подставляются термы унифицирующей подстановки. Список S определяется по информационному элементу (наборпеременных x S). С помощью перечисленных информационных элементов строится оператор, находящий результат R применения ранее

найденной унифицирующей подстановки к терму, определяемому программным выражением A. Затем вставляется проверка включения конъюнктивных членов утверждения R в контекст применения приема.

2. Переменная y уже идентифицирована. Тогда находится программное выражение для утверждения $f(y)$, и добавляется оператор, проверяющий, что оно встречается в контексте применения приема.

3. Переменная y еще не идентифицирована, причем X состоит из единственной переменной x, для которой уже имеется информационный элемент (наборпеременных x S). Тогда для идентификации утверждения $f(y)$ в контексте K применения приема используется оператор "подборзначений". Он определяет подстановку в определяемый программным выражением A терм вместо переменных списка S некоторых термов $R_1, \ldots, R_n$, которая дает конъюнктивные члены утверждения из K. При этом y идентифицируется с термом "набор(R_1 $\ldots R_n$)".

Общий случай идентификации антецедента

Началом данной ветви компилятора служит контрольная точка "прием(116)". Как и выше, обозначаем v идентифицируемое вхождение антецедента; f - заголовок антецедента. Прежде всего, инициализируется нулем переменная x12, которой будет присвоен оператор, перечисляющий утверждения из контекста применения приема для идентификации их с v. Вид оператора зависит от заголовка приема. Перечисление утверждений достигается либо с помощью процедуры "обл", либо с помощью оператора "ключ"; в последнем случае отбираются только утверждения с заголовком f. После переприсвоения переменной x12 данного оператора - переход через "ветвь 2". Далее выполняются следующие коррекции оператора x12:

1. После контрольной точки "прием(117)" происходит учет указателей "альтернатива", "вариант" и "комплексное", относящихся к вхождению v. Он заключается в том, что оператор "ключ(A f ...)" в x12 заменяется либо на дизъюнкцию операторов, соответствующих альтернативным версиям символа f, либо на оператор "альтернатива(...)", выбирающий нужный заголовок согласно ранее идентифицированным вхождениям из указателя "альтернатива".

2. После контрольной точки "прием(118)" происходит анализ фильтров приема замены. Если в них явно указывается, что идентифицируемый антецедент содержит неизвестные, а текущая задача имеет тип "описать" либо "исследовать", то x12 заменяется на оператор, перечисляющий в первом случае условия, а во втором случае - посылки, отличные от утверждения, содержащего точку инициализации приема.

3. После контрольной точки "прием(140)" происходит проверка наличия указателей "списокпосылок" либо "спискокусловий", явно определяющих область идентификации антецедента (в первом случае - список посылок текущей задачи, во втором - список ее условий). Эти указатели порождают соответствующие им информационные элементы (примечпосылки ...), (замечусловие ...). x12 заменяется на оператор, перечисляющий посылки либо условия текущей задачи.

Далее инициализируется пустой накопитель x13, в который при необходимости будет занесен оператор, учитывающий заголовок текущего выделенного оператором x12 утверждения. Если x12 выполняет перечисление с помощью оператора "ключ", то учет заголовка уже произошел, и x13 остается пустым. Если же перечисление выполняется с помощью операторов "посылка", "условие", "обл", то вводится оператор x14, проверяющий наличие заголовка f. Далее x14 корректируется с учетом указателей "альтернатива", "вариант", "комплексное" - как и выше для перечисления с помощью оператора "ключ", а затем заносится в x13.

Наконец, переходя через "ветвь 2", попадаем на точку регистрации накопителей x12, x13 в программе компилируемого приема. Далее в программу вставляются операторы, блокирующие рассмотрение не подходящих для идентификации утверждений - например, утверждений из комментария (исключение ...) к пакетному оператору, либо уже использованных для идентификации других антецедентов. Наконец, вводится установка на идентификацию (операнд ...), обеспечивающая сравнение отобранного утверждения с термом по вхождению v.

19.10.6 Установка на идентификацию "извлекается"

Установка на идентификацию (извлекается v) ссылается на вхождение v антецедента теоремы, для проверки истинности которого необходимо обращение к проверочному пакетному оператору. Рассмотрение этой установки начинается с контрольной точки "прием(119)".

Обработка установки происходит при достаточно больших значениях текущего уровня x7 - 7 либо 8. Это объясняется тем, что обращение к проверочному оператору является сравнительно трудоемкой процедурой, и ее целесообразно выполнять лишь после установления непосредственно наблюдаемых признаков контекста срабатывания приема. После проверки того, что все необходимые для компиляции переменные антецедента v уже идентифицированы - переход через "ветвь 2".

Здесь переменной x10 присваивается вхождение v, переменной x11 - символ f по вхождению v. Далее, в зависимости от f, начинается рассмотрение альтернативных подслучаев.

Обработка кванторной импликации

Если f - квантор общности, то переход через "иначе 1" к контрольной точке "прием(141)". В этом подслучае антецедент v имеет вид $\forall_x(A_1(x)\ \&\ \ldots\ \&A_n(x) \to A_0(x))$, а проверочный оператор обрабатывает последовательность утверждений, возникающих из $A_0(x)$ при перечислении значений переменных связывающей приставки x, определяемых утверждениями $A_1(x), \ldots, A_n(x)$. Последние должны быть программно реализуемыми. Переменной x12 присваивается связывающая приставка x; переменной x13 - терм $A_0(x)$. Проверяется, что все параметры последнего терма, не входящие в x, уже идентифицированы. Затем создается вспомогательный программный блок x15 для компиляции группы антецедентов $A_1(x), \ldots, A_n(x)$. Последовательность операторов, получающаяся после такой компиляции, присваиватся переменной x16. Далее накопитель программ блока x15 расчищается, и этот блок используется для компиляции $A_0(x)$ с помощью проверочного оператора. На конъюнкцию операторов результата компиляции x18 навешивается квантор существования; полученный оператор x20 становится консеквентом кванторной импликации x22, антецеденты которой суть определяющие перечисление операторы x16. Эта кванторная импликация

и реализует проверку антецедента v.

Обработка дизъюнкции

Здесь v - вхождение антецедента вида "или($A_1 \dots A_n$)", где $A_1, \dots, A_n$ - конъюнкции утверждений, для обработки которых во вспомогательных программных блоках могут быть использованы установки (извлекается ...). В частности, на эту роль годятся утверждения, допускающие обработку проверочными операторами. Случай рассматривается после контрольной точки "прием(142)". Последовательно рассматриваются утверждения A_i. Каждое из них обрабатывается во вспомогательном программном блоке x16, и эти программные блоки, содержащие фрагменты программы для проверки истинности A_i, регистрируются в накопителе x13. По окончании цикла заполнения x13 - переход через "иначе 1".

Если хотя бы один из программынх блоков списка x13 имеет информационный элемент "выводимо", то необходим накопитель, регистрирующий все использованные при проверках посылки. Тогда вводится оператор "равно(начало(x3)пустоеслово)", инициализирующий данный накопитель; номера переменных в операторах из x13, больших или равных переменной "начало(x3)", увеличиваются на единицу, и каждый фрагмент программы из x13 дополняется оператором, заносящим в накопитель все использованные в этом фрагменте посылки.

Далее создается набор x14 результатов навешивания кванторов существования по новым переменным на конъюнкции операторов фрагментов программ, представленных в списке x13. Наконец, формируется дизъюнкция x15 операторов списка x14, которая и заносится в компилируемую программу.

Создание обращения к проверочному оператору

Начиная с контрольной точки "прием(132)", рассматривается случай, когда истинность антецедента v проверяется путем непосредственного обращения к проверочному оператору. Предварительно расположен регулятор уровня x7 применения данного приема компиляции: если антецедент v выделен указателем "конец", то этот уровень равен 8, иначе - равен 7.

Чтобы определить заголовок проверочного оператора, который будет обеспечивать обработку антецедента v, предпринимается просмотр расположенных внутри v вхождений x12 логических символов x13. Для них выполняются обращения к справочнику "легковидеть" либо (если имеется указатель на применение усиленного проверочного оператора) справочнику "проверка", который и определяет необходимый проверочный оператор. При этом переменной x14 переприсваивается тройка (заголовок проверочного оператора p - набор расположенных внутри v вхождений, идентифицируемых с входными переменными этого оператора - символьное число уровней срабатывания оператора). Особо рассматривается случай антецедента v, имеющего вид "не(равно(t_1 t_2))", для которого не найден проверочный оператор. Тогда используется проверочный оператор "различимы", способный усматривать различие объектов в некоторых простейших случаях. После попытки определения p для текущих x12, x13 - переход через "ветвь 4".

Здесь, если x14 отлично от 0, т.е. оператор p определен, начинается подготовка ввода в компилируемую программу обращения к нему. Прежде всего, проверяется наличие фильтров "проверка(A n)", у которых n - номер обрабатываемого антецедента v. Условия A таких фильтров должны проверяться непосредственно перед обработ-

кой антецедента v. Находится реализующий проверку истинности A оператор x22, который заносится в компилируемую программу. Чтобы в дальнейшем этот фильтр уже не обрабатывать, он регистрируется в информационном элементе (комментарий ...). Далее - переход через "ветвь 1".

После контрольной точки "прием(15)" выполняется учет указателей приема "спуск" и "спуск($i_1 \dots i_n$)". Эти указатели используются в приемах проверочных операторов для обрыва проверки при неудачной попытке применения приема. В первом случае "отказ" выдается, если после обработки всех антецедентов, не выделенных указателем "блокпроверок", происходит откат. Во втором случае выдача отказа при откате требует также предварительной проверки антецедентов, номера которых суть $i_1, \dots, i_n$. В обоих случаях выдача отказа реализуется за счет вставки оператора "ветвь", переводящего к фрагменту F, состоящему из операторов "блок(...)", "учетвбуфере(...)" и "стоп". Первый из них проверяет, не был ли текущий прием заблокирован специальным комментарием (приемы ...); второй - регистрирует отказ в буфере результатов обращений к данному проверочному оператору. В случае указателя "спуск" решение о вставке оператора "ветвь" принимается, если остались только установки на идентификацию "извлекается". Во втором случае для этого используется информационный элемент (спуск A), у которого список A изначально состоит из всех вхождений антецедентов с номерами $i_1, \dots, i_n$. Обработанные антецеденты исключаются из A, и в тот момент, как список оказывается пустым, происходит вставка оператора "ветвь". Фактически вставляется не оператор "ветвь(
N)", а фиктивный оператор "2(1)". Здесь двойка будет заменена при редактировании откомпилированной программы на "ветвь"; единица же является номером ссылки на фрагмент F, который пока регистрируется не в фрагментах программы блока x3, а в комментарии (блокпрограммы ...). Вставка его в программу будет происходить одновременно с указанной заменой фиктивного оператора перехода на реальный.

Далее переменной x15 присваивается обрабатываемый антецедент (терм); переменной x16 - список всех его свободных переменных. Переменной x17 присваивается список программных выражений для входных термов обращения к проверочному оператору.

Чтобы проверочный оператор получал в качестве входных данных только термы, те элементы списка x17, которые представляют собой однобуквенные термы, состоящие из логического символа s, заменяются на выражения "набор(s)". Без этого преобразования на вход проверочного оператора попал бы сам логический символ s.

Если антецедент представляет собой двуместное отношение, выделенное указателем "дробь", то программные выражения списка x17 модифицируются так, чтобы при необходимости входные данные проверочного оператора переставлялись (контрольная точка "прием(222)"). Затем - переход через "ветвь 1".

Переменной x18 присваивается программное выражение для списка посылок, относительно которых будет происходить проверка антецедента v. Если указатели приема вводили дополнительные посылки P для данной проверки, то P извлекается из информационного элемента (занесениепосылки P), находится программное выражение x20 для P, и конъюнктивные члены P присоединяются к контексту проверки x18. Далее - переход через "ветвь 1".

Инициализируется пустым словом накопитель x19 программных выражений для комментариев создаваемого обращения к проверочному оператору. Определяются посылки, использованные приемом и не выделенные указателем "повтор"; эти посылки регистрируются в комментарии (исключение ...), заносимом в накопитель x19.

Такой же комментарий (исключение ...) вводится и для посылок, перечисленных в информационном элементе (транзитпереход ...).

Если антецедент v выделен указателем "внешобрыв", то компилируется прием проверочного оператора P, причем установление ложности антецедента v влечет ложность проверяемого оператором P утверждения. Чтобы это учитывалось проверочным оператором, обрабатывающим антецедент v, ему передается комментарий (Обрыв K n), где K - ссылка на текущий стэковый кадр, в котором реализуется прием; n - символьный номер переменной проверочного оператора P, являющейся его списком комментариев. В компилируемую программу заносится оператор, определяющий ссылку K, а в x19 - программное выражение для комментария (Обрыв K n).

Если список x19 программных выражений для комментариев непуст, то x19 переприсваивается программное выражение для списка этих комментариев. Рассматриваются указатели приема "комментарий(...)"; "комментарии(...)", определяющие дополнительные комментарии, вводимые при обращении к проверочному оператору. По ним вводятся программные выражения для списков новых комментариев, и x19 корректируется для включения этих списков. Затем - переход через "ветвь 1".

Если компилируется прием пакетного оператора, то переменной x20 присваивается программное выражение для списка комментариев обращения к этому оператору, иначе x20 равно 0. В первом случае рассматриваются указатели приема "исключение(...)", "удалениепримечания(...)", и x20 корректируется так, чтобы были удалены комментарии, на которые ссылаются данные указатели. Затем - снова переход через "ветвь 1".

В случае проверочного оператор либо синтезатора программное выражение x19 корректируется так, чтобы в список комментариев вошли все элементы, определяемые программным выражением x20. Наконец, создается оператор x21, осуществляющий обращение к проверочному оператору для обработки антецедента v. Его единственная выходная переменная - "начало(x3)" - накопитель использованных при проверке утверждений.

Перед регистрацией оператора x21 в программе приема выполняется ряд предварительных его преобразований. Прежде всего, проверяется наличие информационного элемента (сброс ...), указывающего, что при неудачной попытке проверки истинности v должен быть выдан отказ на тот пакетный оператор (проверочный либо синтезатор), к которому относится компилируемый прием. Тогда x21 заменяется на фиктивный оператор "1(x21 n)", который при редактировании программы будет развернут в два оператора: "x21 иначе(...)". При этом n - ссылка на сохраняемый в информационном элементе (блокпрограммы n F) фрагмент программы F, состоящий из единственного оператора "стоп". К нему будет выполняться переход по "иначе" после попытки проверки v.

Если имеется указатель приема "замещение", требующий замены заголовка антецедента v при выполнении специального условия U, то находится оператор x26 для проверки U, создается версия x28 антецедента v, в которой выполнена замена заголовка, и определяется альтернативная версия x33 проверочного оператора x21 для обработки x28. Затем x21 заменяется на условный оператор "альтернатива(x26 x33 x21)", и переход через "ветвь 1".

Переменной x22 присваивается результат модификации оператора x21 процедурой "учетпассива".

Если антецедент v выделен указателем приема "См", то при неустановлении его истинности организуется слежение за последующим возникновением этого антеце-

дента в текущей задаче. Для этого используется комментарий (внимание ...), формируемый процедурой "См". Компилятор определяет программное выражение x26 для антецедента v, заменяет x22 на фиктивный оператор "1(x22 n)" и сохраняет в информационном элементе (блокпрограммы n F) ссылку на фрагмент программы F, обеспечивающий обращение к процедуре "См" при неудачной попытке проверки истинности v.

Наконец, после контрольной точки "прием(21)" происходит регистрация оператора x22 в компилируемой программе.

Если антецедент v выделен указателем "внешобрыв", то после оператора x22 размещается фиктивный оператор "1(x22 1)", который будет заменен при редактировании программы на переход "иначе(...)" к фрагменту x25, сохраняемому пока в информационном элементе (блокпрограммы 1 x25). Фрагмент x25 учитывает информацию о ложности антецедента v, представленную комментарием "внешобрыв"; передает ее при необходимости внешнему проверочному оператору, регистрирует в буфере неудачу проверки, и выдает отказ.

Если антецедент v выделен указателем "Обрыв(...)", определяющим лимит числа шагов работы интерпретатора, отведенных для выполнения проверки, то перед оператором x22 вставляется оператор "лимит(...)" для соответствующего числа шагов, а после x22 - оператор "Обрыв", отменяющий установленный лимит по завершении проверки.

Далее происходит регистрация в информационном элементе (выводимо ...) выходной переменной использованного для обработки v проверочного оператора. Если антецедент v выделен указателем "откат", то после x22 вставляется фиктивный оператор "2(n)", преобразуемый впоследствии в переход "ветвь ..." к фрагменту, состоящему из единственного оператора "обрыв".

19.10.7 Установка на идентификацию "значения"

Установка на идентификацию (значения M) ссылается на список M вхождений антецедентов теоремы, которые обрабатываются с помощью одного обращения к пакетному синтезатору. До обработки должны быть идентифицированы входные (в смысле применяемого синтезатора) переменные антецедентов списка, а термы для выходных переменных будут найдены синтезатором. Выбор подходящего синтезатора осуществляется компилятором самостоятельно. Рассмотрение этой установки начинается с контрольной точки "прием(180)".

Рассмотрение установок (значения M) происходит при текущем уровне x7, равном 2, либо 6, либо большем 8. Переменной x10 присваивается список M. Если хотя бы один из антецедентов этого списка выделен указателем "конец" либо имеется еще не обработанная установка (контекст ...), то допускается лишь уровень x7, больший 8. Переменной x11 присваивается конъюнкция K антецедентов списка M. Чтобы определить синтезатор, который будет обрабатывать эти антецеденты, просматриваются отличные от логических связок и кванторов логические символы x14, входящие в x11. Для каждого такого символа справочник "значения" предлагает список отнесенных к нему названий синтезаторов. Названия x17 из этого списка просматриваются, и справочник "синтезатор" дает набор x18, определяющий формат обращения к синтезатору x17. Этот набор имеет вид (T вход($x_1 \dots x_n$) выход($y_1 \dots y_m$) $A_1 \dots A_k$), где T - шаблон обрабатываемой синтезатором конъюнкции утверждений; $x_1, \dots, x_n$ - список входных переменных шаблона; $y_1, \dots, y_m$ - список выходных переменных шаблона; $A_1, \dots, A_k$ - информационные термы, уточняющие общие особенности син-

тезатора. Переменной x19 присваивается шаблон T; x20 и x21 - списки входных и выходных переменных шаблона. Находится объединенный список x22 этих переменных, и проверяется, что найденная выше конъюнкция K представима как результат подстановки в шаблон T некоторого набора термов x23 вместо переменных списка x22. Заметим, что здесь не анализируется возможность перестановок операндов коммутативных операций, так что порядок антецедентов списка M должен строго соответствовать порядку конъюнктивных членов шаблона T.

Далее переменной x24 присваивается список теоремных термов, соответствующих входным переменным синтезатора x17. Переменной x25 присваивается список всех свободных переменных тех входных термов, которые пока не идентифицированы с помощью информационного элемента (выражение ...). Последний способ идентификации означает, что терм идентифицируется "в целом", возможно, без идентификации каких-либо входящих в него переменных.

Если некоторая переменная x26 списка x25 еще не идентифицирована, но существует указатель "переменные(...)", определяющий ее идентификацию с набором новых переменных, имеющим заданную длину, то в компилируемую программу заносятся операторы, определяющие требуемый список переменных для x26. Затем - переход через "ветвь 2".

Здесь проверяется, что каждая переменная из x25 либо идентифицирована, либо выделена указателем "новаяпеременная", означающим, что для идентификации с ней должна быть выбрана новая переменная. Переменной x26 присваивается список всех теоремных переменных, соответствующих выходным переменным шаблона T.

Если имеются подтермы входных термов списка x24, которые должны обрабатываться нормализаторами, то проверяется, что для них уже введены информационные элементы (быстрпреобр ...), необходимые для компиляции обращений к нормализаторам. Если это не так, то при уровне x7, меньшем 6, обработка данной установки на идентификацию откладывается, а на 6-м уровне предпринимается обращение к процедуре "учетнормализаторов", вводящей информационные элементы (быстрпреобр ...). Далее - откат через "ветвь 1".

Если имеются фильтры приема вида "проверка(n A)", означающие, что проверка условия A должна происходить непосредственно перед обработкой n-го антецедента теоремы, причем n - номер первого из антецедентов списка M, то выполняется компиляция этих фильтров.

После контрольной точки "прием(184)" расположен цикл заполнения накопителя x27 программными выражениями для входных термов списка x24. Эти программные выражения создаются процедурой "метаперевод". После контрольной точки "прием(185)" предпринимается обработка указателей приема "комментарий", "комментарии", относящихся к антецедентам списка M. Она аналогична обработке, имевшей место для проверочных операторов; накопитель x28 заполняется программными выражениями для наборов комментариев компилируемого обращения к синтезатору. После контрольной точки "прием(186)" происходит заполнение пары накопителей x29 программными выражениями для комментариев, исключаемых при обращении к синтезатору из списка комментариев того пакетного оператора, к которому относится компилируемый прием. Первый элемент пары x29 заполняется программными выражениями для отдельных комментариев; второй - для групп комментариев. Далее - переход через "ветвь 1".

Здесь процесс компиляции разветвляется: отдельно рассматриваются случаи приема сканирования задачи и приема пакетного оператора.

Обращение к синтезатору из приема сканирования задачи

Этот подслучай рассматривается начиная с контрольной точки "прием(187)". Переменной x30 присваивается список неиспользуемых программных переменных, имеющий ту же длину, что список x26 выходных теоремных переменных антецедентов M. Эти переменные будут использоваться как выходные переменные обращения к синтезатору. Переменной x31 присваивается набор информационных элементов (терм ...), определяющих связь между теоремными переменными списка x26 и программными переменными списка x30. Переменной x32 присваивается список посылок, используемых синтезатором. При наличии указателей приема "занесениепосылки", относящихся к антецедентам группы M, список x32 пополняется.

После контрольной точки "прием(188)" переменной x33 присваивается оператор, выполняющий обращение к синтезатору. Он регистрируется в компилируемой программе; при наличии относящегося к M указателя "Обрыв(...)" вводятся операторы, учитывающие ограничение на трудоемкость обращения. В заключение элементы списка x31 и элемент (выводимо ...), ссылающийся на использованные синтезатором посылки, регистрируются в программном блоке.

Обращение к синтезатору из пакетного оператора

Этот подслучай рассматривается начиная с контрольной точки "прием(189)". Прежде всего, находится программная переменная x31 для списка комментариев пакетного оператора, к которому относится компилируемый прием. Затем x31 корректируется с учетом списка исключаемых из обращения к синтезатору комментариев, перечисленных в накопителе x29. Далее к x31 добавляются комментарий (исключение ...), перечисляющий использованные для непосредственной идентификации антецедентов посылки, а также все комментарии накопителя x28.

После контрольной точки "прием(267)" выполняется учет указателей "спуск", обеспечивающих обрыв применения пакетного оператора при неудачной попытке обращения к синтезатору. Это делается так же, как для проверочных операторов.

Дальнейшие действия аналогичны случаю приема сканирования задачи: создается оператор x35, выполняющий обращение к синтезатору (контрольная точка "прием(191)"), учитывается указатель "Обрыв", выполняется регистрация в программном блоке оператора x35 и сопровождающих обращение информационных элементов. Добавляется лишь учет указателя "сброс", аналогичного указателю "спуск".

19.10.8 Установка на идентификацию "программа"

Установка на идентификацию (программа v) ссылается на программно реализуемый антецедент v. Рассмотрение ее начинается с контрольной точки "прием(125)". Уровень x7, на котором обрабатывается установка, должен быть больше 0, но меньше 7. Дополнительные условия на этот уровень накладываются идущими вначале операторами. Если x7 = 1, то либо антецедент имеет вид $x = t$, где все переменные выражения t идентифицированы, а x еще не идентифицирована, либо вообще все переменные антецедента уже идентифицированы. Если антецедент v выделен указателем "конец", то x7 = 6. В зависимости от x7, накладываются ограничения на типы еще не обработанных установок на идентификацию. Во всех этих случаях, установки (операнд ...) уже должны быть обработаны. Если антецедент содержит символ "значение", то x7 больше 2.

Переменной x10 присваивается вхождение антецедента v; переменной x11 - сам этот антецедент. Далее рассматриваются подслучаи, определяемые видом антецедента.

Антецедент имеет вид дизъюнкции

Подслучай рассматривается начиная с контрольной точки "прием(278)". Если антецедент имеет такую свободную переменную x, которая уже идентифицирована, однако информационный элемент (транслвыражения x ...), дающий программное выражение значения x в формате, ориентированном на вычисления, отсутствует, то предпринимается попытка создать этот элемент. Для этого анализируются фильтры приема. Если какой-либо из них указывает, что x должна идентифицироваться с числовой константой ("десчисло", "целое", "натуральное"), то для получения программного выражения, дающего значение x в формате десятичного числа ЛОСа, используется обращение к процедуре "прогрвыражение". После рассмотрения указанных переменных x - переход через "ветвь 3".

Здесь определяется список x13 еще не идентифицированных переменных антецедента v - компилируемый оператор должен будет присвоить им некоторые значения. Последовательно просматриваются дизъюнктивные члены x16 утверждения v. Для компиляции его создается вспомогательный программный блок x17, обрабатываемый процедурой "идентификатор". Единственной установкой на идентификацию здесь является (программа w), где w - корневое вхождение утверждения x16. Переменной x19 присваивается конъюнкция операторов, выполняющих обработку дизъюнктивного члена. Она регистрируется в накопителе x12. При этом x14 играет роль накопителя наборов информационных элементов (транслвыражения ...), дающих программные переменные для значений переменных списка x13. Так как итоговый оператор для v будет представлять собой дизъюнкцию операторов списка x12, то нужно, чтобы в различных дизъюнктивных членах значения одной и той же переменной списка x13 давались одной и той же программной переменной - иначе потребуется дополнительная синхронизация выходных переменных. Такая синхронизация пока не реализована, а вместо нее проверяется совпадение значений x14 для всех дизъюнктивных членов. По завершении просмотра дизъюнктивных членов в программу заносится дизъюнкция операторов списка x12; корректируется первая неиспользуемая программная переменная в x3, и предпринимается регистрация информационных элементов из x14.

Антецедент имеет вид конъюнкции

Подслучай рассматривается начиная с контрольной точки "прием(283)". Разумеется, сам по себе конъюнктивный антецедент в теоремах приемов не возникает. Однако, конъюнктивное утверждение, компилируемое с установкой на идентификацию "программа", может появиться, например, как дизъюнктивный член другого компилируемого утверждения (см. разобранный выше случай дизъюнктивного антецедента).

Переменной x12 присваивается список всех еще не идентифицированных переменных рассматриваемого антецедента v. Вводится вспомогательный программный блок x13, в котором будет происходить компиляция v. Создается список x14 установок на идентификацию, состоящий из пар (программа w) для всевозможных конъюнктивных членов w утверждения v. После обращения к процедуре "идентификатор" из программного блока x13 извлекаются возникшие там операторы, и конъюнкция их

регистрируется в программе блока x3. Из блока x13 в блок x3 переносятся также информационные элементы (транслвыражения ...), определяющие идентификацию переменных списка x12.

Антецедент имеет своим заголовком отрицание

Подслучай рассматривается начиная с контрольной точки "прием(284)". Переменной x17 присваивается утверждение под отрицанием. Проверяется, что все свободные переменные этого утверждения уже идентифицированы. Переменной x19 присваивается вспомогательный программный блок, а переменной x20 - одноэлементный список установок на идентификацию, состоящий из пары (программа w); w - корневое вхождение утверждения x17. После обращения к процедуре "идентификатор" находится конъюнкция x21 операторов, возникших в программном блоке x19. Далее на нее навешивается квантор существования по всем ее новым переменным, и отрицание полученного оператора регистрируется в программном блоке x3.

Антецедент имеет своим заголовком квантор существования

Подслучай рассматривается начиная с контрольной точки "прием(289)". Переменной x12 присваивается подкванторное утверждение A. Проверяется, что все свободные переменные антецедента v уже идентифицированы. Вводится вспомогательный программный блок x14 и создается список x15 установок на идентификацию, состоящий из единственного элемента (программа w), где w - корневое вхождение подкванторного утверждения. После обращения к процедуре "идентификатор" находится конъюнкция x16 операторов программного блока x14. На нее навешивается квантор существования по всем новым переменным, и результат регистрируется в программе блока x3.

Антецедент представляет собой кванторную импликацию

Подслучай рассматривается начиная с контрольной точки "прием(330)". Антецедент имеет вид $\forall_i(i \in \{m, \ldots, n\} \,\&\, P(i) \to Q(i))$ и определяет цикл вычислений для последовательных значений целочисленной переменной i, удовлетворяющих условию P. Программной переменной x13 присваивается изменяемая в цикле переменная i; находятся операторные выражения x18 и x19 для начала и конца цикла. Эти выражения определяют значения в формате "целое со знаком". Переменной x20 присваивается список конъюнктивных членов утверждения $P(i)$, накладывающего дополнительные условия на изменяемый параметр i.

Если компилируется прием пакета продукций, то $Q(i)$ может определять преобразования ранее созданных структур данных (например, числовых массивов). Такая ситуация возникает после модификации теоремы приема процедурой "схемавычисл". Необходимые для доопределения преобразований пометки регистрируются этой процедурой в описании приема. В частности, они связывают переменные, используемые в $Q(i)$ для обозначения одной и той же структуры данных до и после преобразования. Переменной x21 присваивается список данных пометок. Затем переменной x22 присваивается список конъюнктивных членов утверждения $Q(i)$. Если в них встречается некоторая функциональная переменная $f(i)$, причем f более нигде в теореме приема не появляется, то эта переменная заменяется на обычную переменную f.

Далее формируется список x25 троек $(x_j,\ a_j,\ b_j)$, соответствующих всевозможным вхождениям в $Q(i)$ и $P(i)$ выражений вида $a + bi$. Для каждого такого выражения выбирается новая переменная x_j; a_j, b_j - значения a, b в этом выражении. Все вхождения в $P(i), Q(i)$ указанных выражений заменяются на соответствующие переменные x_j, которые будут независимо друг от друга и от i инициализироваться и изменяться в цикле со своими шагами b_j.

Затем в программу вводится оператор, инициализирующий переменную i. Значение ее будет сохраняться в программной переменной x26. Тройки списка x25 преобразуются: каждое a_j и b_j заменяется на соответствующее ему программное выражение. Вводится накопитель x28 программных переменных для текущих значений теоремных переменных x_j. Цикл организуется с помощью оператора "повторение", к которому будут происходить откаты при каждом изменении i. Для компиляции консеквента $Q(i)$ вводится вспомогательный программный блок x30. При этом отдельно рассматриваются случаи вырожденного (тождественно истинного) и невырожденного $P(i)$. После регистрации операторов, реализующих $Q(i)$, размещаются операторы, проверяющие завершение цикла и корректирующие i, x_j. По завершении цикла происходит переход через "иначе" либо "ветвь" к фрагментам, продолжающим реализацию приема.

Операторы типа присвоения

Здесь рассматриваются следующие подслучаи:

1. Обычное присвоение. Рассмотрение подслучая начинается с контрольной точки "прием(285)". Антецедент v имеет вид "равно(x t)" (быть может, при перестановке операндов); x - еще не идентифицированная переменная; t - терм, все переменные которого идентифицированы. При помощи операторного выражения "типзнач" определяется указатель "вычисл(f)" формата f, в котором должно быть вычислено значение выражения t. Такие форматы перечислены в справочной информации логического символа "тип". Затем процедура "прогрвыражение" дает программное выражение для вычисления значения t в требуемом формате. В компилируемую программу заносится оператор присвоения для программной переменной, соответствующей переменной x, и регистрируется информационный элемент (транслвыражения x ...), определяющий идентификацию x.

2. Групповое присвоение. Рассмотрение пдслучая начинается с контрольной точки "прием(287)". Антецедент v имеет вид "равно(набор($x_1 \ldots x_n$)t)"; $x_1, \ldots, x_n$ - еще не идентифицированные попарно различные переменные; t - терм, все переменные которого идентифицированы. При помощи процедуры "прогрвыражение" находится программное выражение x18 для вычисления значения t. Этой процедуре передается неопределенный указатель "вычисл(объект)" требуемого формата данных, который может быть уточнен при определении x18. Далее происходит инициализация символом "объект" указателя x19 формата данных для элементов набора t. Если формат значения t был уточнен и имеет после уточнения вид "набор(A)", то указатель A переприсваивается переменной x19. Наконец, реализуется цикл просмотра теоремных переменных x_i, ввода операторов, присваивающих соответствующим программным переменным i - й элемент значения t, и регистрации информационных элементов (транслвыражения x_i ...).

3. Идентификация одноэлементного списка. Рассмотрение начинается с контрольной точки "прием(288)". Антецедент v имеет вид "равно(перечень(набор(x))t)". Как и в случае обычного присвоения, переменная x еще не идентифицирована, а все переменные терма t идентифицированы. Находится программное выражение x20, вычисляющее значение t. По скорректированному указателю "вычисл(объект)" формата данных для представления одноэлементного множества - значения t определяется указатель x21 формата данных, в котором представлен элемент этого множества. Далее в компилируемую программу заносятся: оператор, проверяющий одноэлементность значения t, и оператор, присваивающий программной переменной для x элемент этого значения. Предполагается, что конечные множества при вычислениях на ГЕНОЛОГе задаются конечными наборами.

4. Перечисление делителей целого числа. Рассмотрение начинается с контрольной точки "прием(290)". Антецедент v имеет вид "делит(x t)", где x - еще не идентифицированная переменная; t - терм, все переменные которого идентифицированы. Определяется программное выражение x13 для вычисления значения t в формате десятичных чисел ЛОСа. Если в фильтрах приема указано, что x имеет своим значением простое число, то в компилируемую программу вводится оператор, перечисляющий все простые делители t, иначе - все целочисленные делители (включая отрицательные).

5. Идентификация матрицы. Рассмотрение начинается с контрольной точки "прием(320)". Антецедент v имеет вид "равно(Матрица(A m n)t)", где m - еще не идентифицированная переменная. Он равнозначен присвоению переменным A, m, n: массива, перечисляющего элементы матрицы t, числа строк и числа столбцов этой матрицы. Если m либо n - выражение, все переменные которого уже идентифицированы, то вместо присвоения реализуется сравнение. Сначала находится программное выражение x18 для представления матрицы t. Это представление может иметь один из двух форматов - "матрица(R)", либо "вектор(R)" (число столбцов матрицы равно 1). Здесь R - формат представления элемента матрицы либо вектора. Матрица задается тройкой (M_1 M_2 M_3, где M_1 - массив элементов; M_2, M_3 - числа строк и столбцов, представленные в формате "целое со знаком". Вектор задается аналогичной парой (M_1 M_2). В зависимости от того, как определяется формат значения t после применения процедуры "прогрвыражение", далее отдельно рассматриваются матричный и векторный подслучаи.

6. Определение наибольшего либо наименьшего элемента набора. Рассмотрение начинается с контрольной точки "прием(321)". Антецедент v имеет вид "точкамакс(отображение(i принадлежит(i номера(m n)) $t(i)$)x y)" либо "точкамин(отображение(i принадлежит(i номера(m n)) $t(i)$)x y)". Происходит присвоение переменным x, y номера i и значения $t(i)$, соответственно, для наибольшего либо наименьшего элемента набора. Если таких i несколько, создаваемый компилятором оператор выбирает первое из них.

   Теоремная переменная i присваивается программной переменной x13; выражение $t(i)$ - переменной x16. Проверяется, что все входящие в $t(i)$ отличные от i переменные уже идентифицированы. По косвенным признакам определяется формат x17 для вычисления значения $t(i)$ - целое число либо число с плавающей запятой. Переменным x18, x19 присваиваются программные выражения

для m, n. Далее в компилируемую программу заносятся операторы, инициализирующие отбираемые значения x, y - для номера i и соответствующего значения $t(i)$. Фактически эти значения будут определяться с помощью кванторной импликации, выполняющей перечисление всех целочисленных значений от m до n, сравнение текущего значения $t(i)$ с y и необходимую коррекцию x, y. Для получения программного выражения, вычисляющего в цикле значение $t(i)$, вводится вспомогательный программный блок x21. При организации цикла процедура "циклпарам" выделяет всевозможные изменяемые выражения $a+bi$, вводя для них новые переменные со своими шагами изменения. Это происходит аналогично рассмотренному выше случаю компиляции кванторной импликации.

7. Создание набора псевдокоманд для вычисления значений числовой функции. Рассмотрение начинается с контрольной точки "прием(292)". Антецедент v имеет вид "равно(f отображение($x_1 \ldots x_n$ и(число(x_1) ... число(x_n)) t))", где выражение t без связанных переменных определяет значения функции f. Идентификация f заключается в определении набора псевдокоманд x19, позволяющего вычислять значение t с помощью оператора "Выч(программа ...)". Она осуществляется процедурой "вычпрог". При этом переменной x20 присваивается набор констант, используемых в x19. Для регистрации идентификации служит информационный элемент (усмфункция f ...).

8. Рекурсивное определение функции на конечном отрезке целых чисел. Такие определения возникают при решении задач на синтез программ для решения дифференциальных уравнений либо для решения обычных уравнений методом итераций. Компилятор ГЕНОЛОГа получает в этих случаях не теорему приема, а систему условий задачи на описание, в рамках которой создавалась схема вычислений. Для ссылок на эти условия используются установки на идентификацию (программа ...). Различаются два случая - отрезок целых чисел фиксирован (как при решении дифференциального уравнения) либо верхняя граница его является плавающей (как при решении обычного уравнения методом итераций).

   В первом случае функция f определяется утверждениями "равно(f отображение(x существует(i и(принадлежит(i номера(m n)) равно(x $A+Bi$)) значение(g дробь($x-A$, B)))))", "равно($g(m)$ a)", "длялюбого($jx_1 \ldots x_k$ если принадлежит(j номера(m $n-1$))то и($A_1 \ldots A_n$))". Здесь g - вспомогательная функция целочисленного аргумента, возникающая из f при масштабировании аргумента; второе утверждение инициализирует цикл вычислений значений g на отрезке; последнее - определяет шаг цикла, выражая значение $g(j+1)$ через $g(j)$. Эта конструкция обобщается для нескольких функций, определяемых в одном цикле (используется при решении систем дифференциальных уравнений). Данный случай рассматривается компилятором начиная с контрольной точки "прием(293)" (для нескольких функций - начиная с "прием(300)").

   Во втором случае функция f определяется утверждениями "равно($f(m)$ a)", "длялюбого(i если принадлежит(i номера(m n))то $A(i)$)". Первое из них определяет значение f в начале отрезка; второе - выражает значение $f(i+1)$ через $f(i)$. Кроме этих утверждений, компилятору передается также условие на значение n - при выполнении его цикл обрывается. Случай рассматривается начиная с контрольной точки "прием(312)".

9. Изменение элемента массива. Этот случай может возникнуть в продукции вычислительного пакета. Теорема такой продукции предварительно модифицируется процедурой "схемавычисл", вводящей для изменной версии массива x вспомогательное обозначение - некоторую новую переменную y. Информация о соответствии между x и y сохраняется в терме "схемавычисл(... замена(x y)...)", регистрируемом при компиляции в качестве дополнительного указателя приема. Антецедент v имеет вид "равно(значение(y n)t)". Заметим, что многомерные массивы конвертируются процедурой "схемавычисл" в одномерные, и тогда n становится выражением для вычисления нужной позиции одномерного массива, соответствующей рассматриваемой точке многомерного.

   Рассмотрение подслучая начинается с контрольной точки "прием(342)". Переменные x,y присваиваются, соответственно, программным переменным x18 и x16. Проверяется, что все переменные выражений n, t уже идентифицированы. Чтобы не изменить тот элемент массива, который нужен для последующих вычислений, проверяется, что выражение "значение(x n)" не встречается в антецедентах, указанных в оставшихся установках на идентификацию. Находится информационный элемент (транслвыражения x ...), ссылающийся на изменяемый массив. По нему определяется формат элемента массива, и создается программное выражение x24 для вычисления значения t в этом формате. Предварительно создается программное выражение x23 для получения значения n в формате "целое со знаком". Наконец, в компилируемую программу заносится оператор "Выч(запись ...)", реализующий изменение массива x.

10. Изменение значения переменной. Этот случай, как и предыдущий, возникает при компиляции продукций вычислительного пакета. Данный прием компиляции выделен контрольной точкой "прием(343)" и представляет собой упрощенную версию предыдущего приема.

11. Сравнение двух значений (контрольная точка "прием(347)"). Антецедент имеет вид "равно(t_1 t_2)", причем все его переменные уже идентифицированы. Определяются программные выражения для значений t_1, t_2, и в компилируемую программу заносится их равенство.

Общий случай

Компиляция программно реализуемого антецедента в общем случае осуществляется с использованием справочника "вычисл". Этот справочник связывает вычислимые утверждения либо выражения конкретных типов с вычисляющими их операторами ЛОСа. Он реализуется на ГЕНОЛОГе. Теорема приема такого справочника имеет вид "вычисл(f F $A_1 \ldots A_m$ T)", где f - логический символ, на котором будет происходить обращение к справочнику; F - содержащее f вычисляемое утверждение либо выражение; T - терм, определяющий используемый для вычисления оператор либо операторное выражение. Каждое A_i - терм вида "вход(x_i t_i)", либо "выход(x_i t_i)", либо "выход(t)". В первых двух случаях указывается, рассматривается ли переменная x_i как входная или как выходная, и фиксируется формат t_i ее значений. В третьем случае вычисляется значение выражения F, и t - формат, в котором при использовании операторного выражения T будет представлен результат. Здесь T может иметь также вид "и($C_1 \ldots C_n$ D)", где $C_1, \ldots, C_n$ - операторы, выполнение которых предшествует вычислению операторного выражения D. Форматы, используемые при

вычислениях на ГЕНОЛОГе, перечислены в справочной информации для логического символа "тип". Обращение к справочнику "вычисл" имеет единственную входную переменную x1 - одноэлементный набор, состоящий из пустого накопителя. При обращении этот накопитель заполняется всевозможными наборами ($F\ A_1 \dots A_m\ T$), соответствующими указанным выше теоремам для символа f. Использование справочника позволяет быстро вводить в компилятор новые возможности для вычислений на ГЕНОЛОГе - достаточно создать единственный прием с теоремой указанного выше вида и реализовать на ЛОСе используемые в T процедуры.

Перед обработкой общего случая компилятор присваивает программной переменной x12 список свободных переменных антецедента v, а переменной x13 - список тех из них, которые уже идентифицированы. Вводятся пустые накопители x14, x15. После контрольной точки "прием(286)" реализуется цикл заполнения этих накопителей. Здесь находятся всевозможные вхождения в v более чем двуместных коммутативно-ассоциативных операций $g(t_1 \dots t_n)$, у которых полностью идентифицированы переменные не менее двух, но не всех операндов. Для подоперации $g(\dots)$, образованной операндами со всеми идентифицированными переменными, вводится обозначающая ее новая переменная y. Эта переменная регистрируется в x14, обозначаемая ею подоперация - в x15, а сам антецедент x11 модифицируется - выделенная подоперация в нем заменяется на y. Такая разгруппировка нужна для удобства последующей идентификации антецедента v с шаблонами F - чтобы уже "известные" операнды ассоциативно-коммутативной операции рассматривались как один операнд.

Далее - переход через "ветвь 1". Переменной x16 присваивается список всех входящих в v логических символов. Выполняется просмотр элементов списка x16. На текущем символе x17 предпринимается обращение к справочнику "вычисл", заполняющему накопитель x18 наборами, ссылающимися на операторы ЛОСа, которые могли бы быть использованы для обработки v. Просматриваются наборы x20 = (F $A_1 \dots A_m\ T$) из x18. Анализируются числовые типы данных, идентифицированных в v. При наличии явного несоответствия их входным типам данных, указанным в наборе x20, этот набор отбрасывается. Далее - переход через "ветвь 2", где проверяется совпадение заголовков шаблона F и антецедента v. Находится результат x23 переобозначения переменных x21 шаблона F на переменные x22, не встречающиеся в v, и с помощью процедуры "унификация" определяется подстановка вместо переменных списка x22, преобразующая x23 в антецедент v. Переменной x24 присваивается набор подставляемых термов.

Находится список x25 соответствующих входным переменным x_i оператора T троек ($t_i\ y_i\ r_i$), где t_i - теоремное выражение, значение которого должно подаваться на вход оператора; y_i - переменная, в которую x_i перешла при переобозначении переменных в F; r_i - тип значения t_i. Проверяется, что все переменные термов t_i уже идентифицированы. Аналогичный список x26 троек ($t_i\ y_i\ r_i$) составляется для выходных переменных оператора T. Проверяется, что выражения t_i списка x26 суть попарно различные еще не идентифицированные переменные. Составляется список x27 программных выражений для вычисления значений входных термов t_i. Находится результат x28 такого же переобозначения переменных в операторе T, которое было сделано для F. Определяется список x29 всех свободных переменных терма x28. Вводится накопитель x30 программных выражений для переменных списка x29, который заполняется в цикле синхронного просмотра обоих списков. При этом для выходных переменных выбираются новые программные переменные, и это соответствие регистрируется в информационных элементах (транслвыражения ...). Наконец, определяется результат x31 подстановки в x28 выражений x30 вместо переменных x29, и

конъюнктивные члены оператора x31 заносятся в компилируемую программу - они и выполнят обработку антецедента v.

19.10.9 Установка на идентификацию "идентификатор"

Если антецедент v выделен установкой (идентификатор v), то он определяет некоторые вспомогательные термы (например, с помощью обращений к задачам и пакетным нормализаторам), которые либо сравниваются между собой (как записи), либо идентифицируются с заданными шаблонами. В последнем случае продолжение идентификации обеспечивается с помощью установок (операнд ...). Чаще всего антецедент v имеет вид равенства, одна из частей которого определяет вспомогательный терм, а другая задает шаблон для идентификации с ним.

Обработка установки "идентификатор" начинается с контрольной точки "прием(126)". Если антецедент выделен указателем "начало", причем все переменные, на которые ссылается данный указатель, уже идентифицированы, то компиляция происходит при текущем уровне x7, равном 0. Если антецедент выделен указателем "конец", то она откладывается до уровня 7. В обычной ситуации компиляция выполняется при x7, равном 5. Предварительно проверяется отсутствие информационного элемента (внтурпосылка v ...), означающего, что v есть кванторная импликация, сопровождающая некоторую другую кванторную импликацию и компилируется при рассмотрении последней.

Перед обработкой установки выполняется обращение к процедуре "учетнормализаторов", вводящей те необходимые для компиляции нормализаторов информационные элементы (быстрпреобр ...), которые возможно создать на текущем этапе компиляции. После оператора "равно(x7 5)" выполняется проверка наличия более приоритетных еще не обработанных установок на идентификацию x12. Предварительно определяется список x11 еще не идентифицированных переменных антецедента v. Более приоритетными считаются установки типа (операнд ...), (извлекается ...) и (корень ...) (последние - только если все их переменные уже идентифицированы), при условии, что они не выделены указателями "конец", не относятся к антецедентам, содержащим переменные списка x11 и не должны обрабатываться после идентификации переменных из x11, не содержат подтермов, которые перед идентификацией должны специально формироваться. При этом сам антецедент v не должен быть выделен указателем "начало". Если найдена более приоритетная установка, то обработка данной установки откладывается, иначе - откат к переходу через "ветвь 2".

Здесь переменной x10 присваивается корневое вхождение антецедента v. Далее начинается рассмотрение подслучаев для различных заголовков этого антецедента.

Эквивалентность либо равенство

Подслучай рассматривается начиная с контрольной точки "прием(148)". Прежде всего, выбирается вхождение x11 той части равенства, которая должна будет определить вспомогательный терм для последующей его идентификации (либо сравнения) с другой частью равенства либо эквивалентности. Подтерм по вхождению x11 присваивается переменной x12. Находится список x13 всех свободных переменных терма x12, из которого исключаются такие переменные x16, которые встречаются в x12 только внутри какого-либо уже идентифицированного "в целом" подтерма x15. Идентификация таких переменных для определения соответствующего x11 вспомо-

гательного терма не обязательна. После перехода через "ветвь 2" проверяется, что отсутствует указатель приема "видпеременной", определяющий обработку антецедента v лишь после идентификации еще не идентифицированной его переменной x. Проверяется, что каждая переменная списка x13 либо уже идентифицирована, либо для нее должна быть выбрана новая переменная, либо набор новых переменных, либо матрица новых переменных. Проверяется, что переменные вспомогательных посылок, используемых нормализаторами, обрабатывающими подтермы терма x12, либо идентифицированы, либо выбираются как новые переменные. Если имеется указатель приема, определяющий для не идентифицированной переменной списка x13 выбор новой переменной, либо списка новых переменных, либо матрицы новых переменных, то в программу приема заносятся операторы, выполняющие такой выбор, а в информационных элементах программного блока регистрируется идентификация переменной. Далее - переход через "ветвь 1". Здесь компиляция разветвляется. Рассматриваются следующие подслучаи:

1. x12 - переменная x , которая должна идентифицироваться с многочленом, причем антецедент v имеет вид "равно(x отображение(y число(y) $ay^n + p(y)$))" (контрольная точка "прием(349)"). Это достаточно редкий случай, в котором антецедент дает идентификацию старшего коэффициента многочлена, его степени и суммы младших членов. Информационный элемент (смногочлен x t) определяет программное выражение t для набора коэффициентов многочлена x, расположенных по возрастанию степеней. Если на момент обработки установки выражение t было не определено (равно 0), то предпринимается попытка здесь же определить его, используя вхождение, идентифицированное с x и применяя для нахождения набора коэффициентов процедуру "смногочлен" либо, для комплексного случая, "смМногочлен". Затем регистрируются операторные выражения, определяющие по известному набору коэффициентов многочлена его старший коэффициент a, степень n и сумму младших коэффициентов $p(y)$.

2. x12 - выражение "отображение(x $P(x)$ $t(x)$)", причем противоположный операнд равенства v представляет собой переменную y, для которой имеется указатель приема, определяющий ее идентификацию как функциональной переменной. Это - тоже крайне редкий случай. Его обработка начинается после контрольной точки "прием(350)". Если переменная x еще не идентифицирована, то вводится оператор, выбирающий для нее новую переменную. Затем - откат к переходу через "ветвь 2". Здесь определяются программные выражения x16 и x17, соответственно, для термов $t(x)$ и $P(x)$, и предпринимается идентификация обычной переменной y как функциональной. Для этого вводится информационный элемент (функция y x ...), который далее можно будет использовать для формирования выражений вида $y(a)$.

3. Общий случай (контрольная точка "прием(351)". С помощью процедуры "метаперевод" находится программное выражение x14, определяющее терм x12. Предварительно проверяется отсутствие таких обрабатываемых нормализаторами подтермов терма x12, для которых еще не введен информационный элемент (быстрпреобр ...). Если обращение к процедуре "учетнормализаторов" не создает этих информационных элементов, то обработка установки откладывается.

   Если имеется фильтр приема, заключенный в служебную конструкцию "проверка(...)", указывающую, что его следует проверять непосредственно перед

обработкой антецедента v, то этот фильтр компилируется и вставляется в программу. Затем - переход через "ветвь 2".

Если противоположный операнд w равенства либо эквивалентности v выделен указателем "развертка", то происходит преобразование текущей установки на идентификацию. Вместо нее вводится установка (развертка ...), определяющая идентификацию w с термом - значением программного выражения x14. В противном случае - переход через "ветвь 2".

Здесь рассматривается подслучай, в котором антецедент v выделен указателем приема "старшийчлен(...)". Тогда он имеет вид "равно($t \quad ay^n + b$)". Идентификация его состоит в рассмотрении выражения для t (т.е. значения x14) как многочлена от переменной, идентифицированной с y. При этом определяются выражения для коэффициента a при старшей степени, суммы b младших членов и степени многочлена n. Данный случай аналогичен рассмотренному выше случаю выделения антецедента указателем "сммногочлен(...)". Отличие состоит лишь в том, что здесь правая часть не имеет функциональной записи "отображение(...)". Если указателя "старшийчлен(...)" не было, то переход через "ветвь 1".

Находится противоположный член x16 равенства либо эквивалентности. Если все его переменные идентифицированы, то определяется программное выражение x18 для этого члена. В компилируемую программу вставляется оператор, сравнивающий значения выражений x18 и x14.

Далее, если ни один из перечисленных подслучаев не имел места, выполняется откат к переходу через "ветвь 1". Здесь, после контрольной точки "прием(352)", рассмотрение общего случая продолжается. Если программное выражение x14 отлично от переменной, то для него вводится вспомогательная переменная. Если противоположный член x15 равенства либо эквивалентности представляет собой функциональную переменную, то она здесь же идентифицируется. Иначе (быть может, после коррекции информационного элемента (свертка ...)) - переход через "ветвь 1".

Здесь происходит ввод в компилируемую программу оператора, присваивающего новой программной переменной x16 корневое вхождение в терм, определяемый выражением x14. Это инициирует цикл идентификации по x14 противоположной части равенства либо эквивалентности. Впрочем, перед завершением рассмотрения "общего случая" возникает еще одно ответвление программы компилятора - если антецедент v выделен указателем "дизъюнктчлен" (контрольная точка "прием(115)". Это достаточно сложный подслучай, и мы вынесем его рассмотрение в следующий пункт. Для продолжения рассмотрения общего случая переходим через "ветвь 1".

Переменной x15 присваивается вхождение противоположного (идентифицируемого по x14) члена равенства либо эквивалентности. Вводится информационный элемент (операнд x15 ...), указывающий идентифицирующее корневое вхождение для x15, а также элемент (подтерм x15), указывающий, что для идентификации x15 используется вспомогательный терм. Затем анализируется заголовок x17 терма x15. Если этот заголовок представляет собой логический символ, причем отсутствует указание на то, что идентифицирующий терм T может иметь другой заголовок, то в компилируемую программу заносится оператор, контролирующий заголовок T. Предпринимается попытка выпол-

нить идентификацию подтерма x15 с применением справочника "вид" (этот справочник уже рассматривался ранее в связи с установкой (операнд ...)). Если x17 - переменная, то в зависимости от того, была ли она идентифицирована, либо проверяется совпадение ранее найденного для нее терма с новым термом x14, либо регистрируется ее идентификация с x14. Наконец, если одношаговая идентификация подтерма x15 не состоялась, происходит откат к переходу через "ветвь 1", где для продолжения обработки x15 создается установка на идентификацию (операнд x15 x16).

4. Антецедент v выделен указателем "дизъюнктчлен". Рассмотрение этого подслучая начинается после контрольной точки "прием(115)". Предварительно, как и в общем случае, было создано программное выражение x14 для идентифицирующей части x12 рассматриваемых равенства или эквивалентности. Данный указатель выделяет антецедент вида "равно(A B)", у которого идентифицирующая часть A, после обработки нормализаторами, может приобрести вид дизъюнкции. Тогда B будет идентифицироваться не с самой дизъюнкцией, а последовательно со всеми ее дизъюнктивными членами. Прием выполняет эквивалентную замену, и заменяющий терм должен формироваться как дизъюнкция отдельных утверждений, соответствующим подслучаям идентификации B. Эта схема действий бывает удобна для создания приемов, решающих вспомогательные уравнения и формирующих на основе отдельных серий их решений ответ основного уравнения.

   Переменной x15 присваивается корневое вхождение терма, определяемого идентифицирующей частью A. Затем находится информационный элемент (дизъюнктчлен C_1 C_2 C_3 C_4). У него C_1 - набор вхождений всех антецедентов, выделенных указателем "дизъюнктчлен"; C_2 - сначала 0, а затем программная переменная для накопителя дизъюнктивных членов заменяющего терма приема; C_3 - сначала 0, а затем метка отката при обрыве идентификации; C_4 - сначала 0, а затем набор меток отката для продолжения перечислений идентифицируемых дизъюнктивных членов. Длина этого набора равна длине набора C_1.

   Если C_2 оказывается равно 0, то выбирается новая метка перехода x17, и в программу вставляется оператор "переходпометке(x17)". К этой точке будут происходить откаты, если идентификация какого-либо антецедента списка C_1 окажется неудачной - чтобы предотвратить выдачу в качестве результата частичной дизъюнкции. Затем в программу вставляется оператор, инициализирующий пустым словом накопитель дизъюнктивных членов результата. Переменная, являющаяся этим накопителем, регистрируется как C_2, а метка x17 - как C_3. Здесь же происходит инициализация C_4 набором из нулей. Если компилируется прием сканирования задачи, то после инициализации накопителя результата размещается оператор, инициализирующий набор информационных элементов оператора "заменавхождения", которым будет пользоваться прием. Далее - откат к переходу через "ветвь 2".

   Здесь выбирается новая ссылка x18 на дополнительный фрагмент программы, который будет вставлен впоследствии с помощью информационного элемента (блокпрограммы ...), и в компилируемую программу заносится фиктивный оператор "1(P x18)", где P реализует перечисление дизъюнктивных членов вхождения x15. При редактировании программы он будет преобразован в "P

иначе x18". По окончании рассмотрения всех подслучаев прием будет переходить к фрагменту по ссылке x18. Затем выбирается новая метка перехода x19, и вводится оператор "меткаперехода(x19)". К нему будут выполняться переходы для рассмотрения очередного дизъюнктивного члена. Далее размещается фиктивный оператор "2(N)", где N - символьный номер, следующий за x18. При редактировании программы он будет преобразован в "ветвь N" и обеспечит переход к фрагменту программы, организующему откат к метке x17 при неудачной идентификации.

Если прием имел еще и указатель "дизъюнктблок(T)", определяющий присоединение к результату дополнительного дизъюнктивного члена T (например, каких-то особых решений уравнения), то создается набор x20 операторов, пополняющих накопитель C_2 утверждением, определяемым по T.

Наконец, в информационных элементах (блокпрограммы ...) регистрируются фрагменты, к которым имеются переходы по ссылкам x18 и N. Если рассматриваемый антецедент - первый в цепочке вложенных перечислений, определяемых антецедентами из C_1, то фрагмент x18 полагается равным x20, иначе он содержит откат к продолжению предыдущего перечисления.

Пополнение накопителя C_2 по завершении рассмотрения очередного подслучая выполняет оператор, вставляемый компилятором на этапе синтеза преобразующей части приема (например, в случае приемов сканирования задачи - см. процедуру "преобразователь").

Антецедент с заголовком "альтернатива"

Подслучай рассматривается после контрольной точки "прием(149)". Антецедент v здесь имеет вид "альтернатива(A B_1 B_2)", причем все переменные утверждения A уже идентифицированы и оно играет роль переключателя, выбирающего для реализации один из операторов B_1, B_2. Те конъюнктивные члены утверждения B_i, которые представляют собой равенства, будут обрабатываться с установкой "идентификатор", прочие члены - с установкой "извлекается" (т.е. с помощью проверочных операторов).

Для получения операторов, проверяющих истинность A, создается вспомогательный программный блок x12. Его первая неопределенная программная переменная берется на 1 большей, чем первая неопределенная программная переменная блока x3, которая будет играть роль накопителя P использованных приемом посылок. После компиляции условия A процедурой "идентификатор" происходит занесение в конец программы блока x12 операторов, пополняющих накопитель P. Процедура "сборкафильтра" формирует оператор x13 для проверки A, помещая под квантором существования по всем новым переменным конъюнкцию операторов блока x12. Далее предпринимается компиляция утверждений B_1, B_2. Для каждого из них создается свой программный блок x16, и после применения к нему процедуры "идентификатор" происходит добавление операторов, пополняющих P. Пара вспомогательных программных блоков с результатами обработки B_1, B_2 сохраняется в накопителе x14, и переход через "иначе 2". Переменной x16 присваивается список всех еще не идентифицированных переменных, входящих в B_1, B_2. Далее выполняются преобразования программ S_1, S_2, реализующих B_1 и B_2, направленные на синхронизацию обозначений программных переменных, идентифицирующих переменные x19 из x16. Если некоторая такая переменная x19 вычисляется в S_1, S_2 с помощью различных про-

граммных переменных, то вводится новая программная переменная, которой в каждой из программ S_1, S_2 переприсваиваются значения, найденные для x19. Именно она далее и рассматривается как идентифицирующая для x19. Список переменных, идентифицирующих для переменных списка x16, регистрируется в накопителе x17. По окончании обработки переменных списка x16 - переход через "иначе 1". Здесь предпринимается отбрасывание из программ S_1, S_2 операторов, не используемых при вычислении значений x17. Затем - переход через "ветвь 1"и формирование оператора x19 вида "альтернатива(...)", реализующего антецедент v. Перед x19 вставляется оператор, инициализирующий пустым словом накопитель P.

Антецедент с заголовком "или"

Подслучай рассматривается после контрольной точки "прием(150)". Антецедент v имеет вид "или($A_1 \dots A_n$)". Те конъюнктивные члены утверждений A_i, которые представляют собой равенства, будут обрабатываться с установкой "идентификатор", прочие члены - с установкой "извлекается" (т.е. с помощью проверочных операторов).

Компиляция совершенно аналогична ранее разобранному случаю заголовка "альтернатива". Последовательно просматриваются вхождения x12 дизъюнктивных членов A_i. Для текущего такого члена создается вспомогательный программный блок x13. x11 играет роль накопителя блоков с уже созданными фрагментами программы для обработки A_i. После заполнения x11 - переход через "иначе 2". Создается список x13 еще не идентифицированных в основном блоке x3 переменных антецедента v. Эти переменные x16 просматриваются с целью синхронизации программных переменных в различных блоках списка x11, определяющих их значение. x14 используется как накопитель этих программных переменных. После некоторой расчистки программ для обработки дизъюнктивных членов создается оператор x16 - дизъюнкция конъюнкций данных программ. Этот оператор регистрируется в компилируемой программе после оператора, инициализирующего накопитель использованных посылок.

Перечисление целых чисел

Здесь рассматриваются два подслучая. В первом из них (контрольная точка "прием(152)") антецедент v имеет вид "целое(x)", и компилятор вводит оператор "Целое(...)" для перечисления всех целых значений в порядке возрастания их модулей. Переменная x идентифицируется с термами, представляющими эти числа.

Второй подслучай (контрольная точка "прием(172)") относится к антецеденту вида "принадлежит(x номера(m n))", перечисляющему все целочисленные значения от m до n. Перечисление реализуется оператором "Номера".

Антецедент с заголовком "длялюбого"

Рассмотрение данного случая начинается с контрольной точки "прием(245)". Антецедент v имеет вид кванторной импликации "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)". x10 - корневое вхождение этого антецедента; x11 - его связывающая приставка. Предусмотрена компиляция в следующих подслучаях:

1. Проверка совпадения пар термов, перечисление которых задается кванторной импликацией (контрольная точка "прием(249)"). Консеквент A_0 представляет

собой равенство, все переменные которого, отличные от $x_1, \ldots, x_n$, уже идентифицированы. Утверждения $A_1, \ldots, A_m$, рассматриваемые как программно реализуемые антецеденты, компилируются во вспомогательном программном блоке x14 в программу P, перечисляющую некоторые значения переменных $x_1, \ldots, x_n$. Относительно этого же блока определяются программные выражения T_1, T_2 для обеих частей равенства. Наконец, создается кванторный оператор x20 вида "длялюбого($y_1 \ldots y_k$ если P то равно(T_1 T_2))", проверяющий, что при всех допустимых значениях переменных $x_1, \ldots, x_n$ равенство соединяет одинаковые (после определяемой приемом нормализации) термы.

2. Векторная идентификация (контрольная точка "прием(250)"). Кванторная импликация имеет единственный антецедент A_1 вида "принадлежит(x_1 номера(1 N))"; длина связывающей приставки равна 1. A_0 - равенство, все не идентифицированные переменные которого, отличные от x_1, суть обозначения f для его же функциональных переменных $f(x_1)$. Эти функциональные переменные идентифицируются с векторами (наборами термов) в цикле изменения параметра x_1. Прежде всего, составляется список x19 указанных переменных f и для каждой из них вводится программная переменная x20 - накопитель набора термов, инициализируемая пустым словом. Создается информационный элемент (вектор ...), связывающий переменную с ее накопителем. Формируется оператор x22 вида "Номера(...)", который будет выполнять перечисление значений переменной x_1. Для компиляции консеквента создается вспомогательный программный блок x23, которому сообщается идентификация переменной x_1 с выходной переменной пока нигде не зарегистрированного оператора x22. Консеквент A_0 обрабатывается в этом блоке при помощи установки на идентификацию (идентификатор ...). Затем просматриваются переменные f списка x19, и для каждой из них к программе блока x23 добавляется оператор, пополняющий накопитель переменной f очередным термом. Этот терм является идентифицирующим для функциональной переменной "значение(f x_1)". Наконец, вводится оператор вида "длялюбого(y если x22 то существует(z_1 $\ldots z_q$ P))", где P - конъюнкция операторов программы блока x23. Переменные y, z_j - новые программные переменные, встречающиеся, соответственно, в операторе x22 и в P.

3. Матричная идентификация (контрольная точка "прием(248)"). Здесь кванторная приставка состоит из двух переменных, которые будем обозначать далее через i, j. Число антецедентов также равно 2 - "принадлежит(i номера(1 N))" и "принадлежит(j номера(1 M))". Программные выражения для вычисления N, M в формате десятичных чисел присваиваются переменным x18 и x22. Консеквент A_0 имеет вид "равно(значение(B набор(i j))R)". При компиляции предполагается, что матрица B уже идентифицирована и данные о ней содержатся в информационном элементе (элементыматрицы B ...). Находится список x29 всех еще не идентифицированных переменных f выражения R, отличных от i, j, и проверяется, что они относятся к функциональным переменным "значение(f набор(i j))"того же выражения. Эти функциональные переменные должны быть идентифицированы с матрицами тех же размеров, что и матрица B, и зарегистрированы в информационных элементах (элементыматрицы f ...).

   Чтобы идентифицировать в цикле перечисления значений i, j выражение R с текущим элементом матрицы B, удобно заменить все вхождения функциональ-

ных переменных "значение(f набор(i j))" на обычные переменные f. Результат такого преобразования R присваивается переменной x31. Затем вводятся операторы программы, инициализирующие пустыми словами накопители строк матриц, соответствующих переменным f. Для регистрации идентификации этих матриц создаются информационные элементы (элементыматрицы f ...).

Для организации внешнего цикла (варьирование переменной i) вводится вспомогательный набор операторов x33. Кроме оператора "Номера(...)", в него заносятся также операторы, инициализирующие пустыми словами накопители элементов текущей строки для каждой переменной f из x29. Переменной x32 присваивается список данных накопителей.

Аналогично, для организации внутреннего цикла (по переменной j) вводится вспомогательный набор операторов x34. Создается вспомогательный программный блок x35 для компиляции шага цикла. Из него исключаются информационные элементы (элементыматрицы f ...), соответствующие функциональным переменным набора x29, так как в рамках шага цикла эти переменные будут идентифицироваться не как матрицы, а как обычные термы. В x35 регистрируются программные выражения для текущих значений i, j, взятые из наборов x33, x34. Затем переменной x36 присваивается программное выражение для корневого вхождения текущего элемента идентифицирующей матрицы B. Если это выражение не является переменной, то для него вводится новая переменная. К накопителю операторов внутреннего цикла добавляются те операторы блока x35, которые могли возникнуть в нем при компиляции вычисления x36.

Переменной x37 присваивается список тех новых программных переменных, которые появились в программном блоке x35 по сравнению с основным программным блоком x3. Это список будет нужен как связывающая приставка кванторной импликации, реализующей внутренний цикл.

Далее накопитель программы блока x35 сбрасывается (его элементы уже учтены в x34). Ссылки из информационных элементов блока x35 на вхождения в подтерм R преобразуются в ссылки на соответствующие вхождения в терм x31, так как именно он будет использоваться при идентификации на шаге цикла. Затем происходит обращение к процедуре "идентификатор", выполняющей данную идентификацию. При этом используется единственная установка на идентификацию (операнд ...), связывающая программную переменную x36 - текущий элемент матрицы B - с корневым вхождением в терм x31.

После идентификации x31 к концу программной заготовки блока x35 добавляются операторы, регистрирующие текущие значения переменных списка x29 в накопителях строки для их матриц. Переменной x38 присваивается конъюнкция операторов программного блока x35, на которую навешивается квантор существования по всем новым переменным этого блока, кроме уже учтенных в списке x37. Наконец, создается оператор - кванторная импликация, реализующая внутренний цикл по j. Антецедентами ее служат операторы списка x34, а консеквентом - оператор x38. Данная кванторная импликация заносится в одноэлементный список x39, который затем пополняется операторами, регистрирующими текущие версии накопителей строки матриц f списка x29 в информационных элементах (элементыматрицы f ...). Наконец, вводится итоговый оператор x41 для реализации антецедента v. Антецедентами его служат операторы списка x33 (они обеспечивают цикл по i), а консеквентом -

конъюнкция операторов набора x39.

19.10.10 Установка на идентификацию "контекст"

Установка на идентификацию (контекст A) возникает в тех случаях, когда прием имеет указатель "контекст(...)", определяющий идентификацию каких-то переменных вне рассмотрения теоремы. Этот указатель организован так же, как одноименный фильтр "контекст(...)". Он определяет значения своих внутренних переменных (не определенных по другим частям описания приема), удовлетворяющие заданным в нем условиям. В отличие от фильтра "контекст(...)", внутренние переменные которого связаны квантором существования, указатель "контекст(...)" передает значения своих внутренних переменных в основной цикл идентификации, где они могут быть использованы наравне с переменными, идентифицированными по теореме приема.

Терм A установки (контекст A) имеет вид "идентификатор($B_1 \dots B_n$)"; соответствующий указатель - вид "контекст(и($B_1 \dots B_n$))". Обработка установки начинается с контрольной точки "прием(127)". Терм A присваивается здесь переменной x10; теорема приема - переменной x11. Уровни x7 обработки установки "контекст" - либо 0 (случай пакетных синтезаторов, которые часто получают входную информацию через список комментариев, извлекая ее оттуда с помощью указателя "контекст"), либо 2, либо 7.

Сначала рассматривается особый случай - B_1 имеет вид "вид(корень $f(X)$)" для некоторой еще не идентифицированной функциональной переменной f, а компилируемый прием относится к пакетному нормализатору. Тогда для идентификации f по корневому вхождению преобразуемого нормализатором терма сразу вводится информационный элемент (функция f ...).

Если указанный случай не имеет места, то для обработки установки (контекст ...) используется справочник "блокпроверок", который обрабатывает указатель "контекст(...)" так, как если бы он был фильтром. Отличие состоит лишь в том, что на заключительном этапе обработки этот справочник присоединяет к компилируемой программе все операторы, реализующие фильтр, не навешивая на их конъюнкцию квантор существования. Кроме того, в основной программный блок переносятся из блока компилияции фильтра все информационные элементы, определяющие идентификацию внутренних переменных. Заметим, что процедура справочника "блокпроверок" на символе "контекст" представляет собой один из важных крупных блоков компилятора и будет рассмотрена ниже в специальном разделе.

19.10.11 Проверка истинности антецедента с помощью вспомогательной задачи либо общей проверочной процедуры

В этом разделе объединены установки на идентификацию (легковидеть v), (усматривается v), (доказать v), (очевидно v), (следствие v). Здесь v - вхождение антецедента, для проверки истинности которого нужно использовать либо вспомогательную задачу, либо вспомогательную проверочную процедуру общего назначения (процедуры "легковидеть", "очевидно").

Обработка рассматриваемой установки (P v) начинается с контрольной точки "прием(133)". Так как использование вспомогательной задачи - достаточно трудоемкая процедура, то уровень x7 компиляции этой установки берется равным 8, либо,

при выделении антецедента v указателем "начало(...)", равным 6. Это обеспечивает размещение обращений к проверкам в конце программы приема. Если P = "доказать", то проверяется отсутствие менее трудоемких вариантов установок - "усматривается" и "легковидеть". Проверяется отсутствие установки "свертка", при которой может быть продолжена непосредственная идентификация. Проверяется также, что все свободные переменные антецедента v, не выделенные указателем "новаяпеременная", уже идентифицированы.

Переменной x10 присваивается программное выражение для определяемого антецедентом v проверяемого утверждения. Если компилируется прием сканирования задачи и P = "доказать", то имеет место особый случай сведения задачи на доказательство к задаче на доказательство одного лишь утверждения v. Компиляция здесь будет несколько отличаться от прочих случаев, в которых лишь обеспечивается проверка истинности v перед реализацией основных действий приема. Соответственно, в данном случае переменной x11 присваивается 1, иначе ей присваивается программное выражение для списка утверждений, образующих контекст проверки.

Если x11 отлично от 1, то анализируются информационные элементы (удалениепосылок ...), определяющие условия на утверждения списка x11, отбрасываемые из контекста проверки. Выполняется соответствующая коррекция программного выражения x11. Затем - откат к переходу через "ветвь 4", где аналогичным образом x11 корректируется с учетом информационных элементов (занесениепосылки ...), добавляющих некоторые утверждения к контексту проверки.

Если P = "очевидно", то в компилируемую программу заносится оператор "очевидно(x10 x11 ...)", реализующий проверку истинности x10 относительно x11 либо при помощи подходящего проверочного оператора, либо при помощи некоторых простейших общих соображений. В противном случае - переход через "иначе 1".

Инициализируется пустым словом накопитель x12 программных выражений,определяющих наборы комментариев, передаваемых вспомогательным задаче либо оператору при проверке v. Для его заполнения просматриваются указатели приема "комментарий(i ...)", относящиеся к антецеденту v. По ним обычным образом вводятся программные выражения для комментариев. Фактически каждое такое выражение определяет не комментарий, а либо пустой список, либо одноэлементный список, состоящий из комментария. Это делается для того, чтобы учесть условия, при которых комментарий должен быть введен - при их нарушении выдается пустой список. После просмотра указателей "комментарий(...)" - переход через "иначе 1".

В случае P = "легковидеть" к x12 добавляется набор, состоящий из комментария "извлекается", которым для ускорения проверки будет сопровождаться вспомогательная задача. В случае приема синтезатора аналогичным образом добавляется комментарий "значение". Далее по списку x12 не более чем одноэлементных фрагментов списка комментариев определяется программное выражение x13 для самого списка комментариев. Затем компиляция разветвляется. Рассматриваются следующие подслучаи:

1. Компилируется прием сканирования задачи, причем P = "доказать". Тогда - переход через "иначе 1". Здесь в компилируемую программу вводится оператор, присваивающий первой неиспользуемой программной переменной "начало(x3)" ту задачу на доказательство Z', к которой сводится текущая задача на доказательство Z. Задача Z' получается из Z при помощи оператора "спуск", заменяющего ее условие на проверяемый антецедент (он задается программным выражением x10). Вводится также оператор, пополняющий комментарии

задачи Z' с учетом x13. Наконец, вводятся оператор "подуровень(...)", устанавливающий уровень обращения к Z' равным максимальному уровню задачи Z, и оператор "равно(ответзадачи(...)истина)", осуществляющий обращение к решению Z' и проверку того, что это решение было получено.

Если прием имел указатель "Обрыв(...)", устанавливающий ограничение на трудоемкость решения задачи Z', то перед обращением к ее решению вставляется оператор "лимит(...)", контролирующий это ограничение, а после обращения - оператор "Обрыв", отменяющий контроль. Затем - переход через "ветвь 1", где очередной неиспользуемой программной переменной передается формируемый на основе комментариев (выводимо ...) список посылок, использованных при решении задачи Z'. Эта переменная регистрируется в информационном элементе (выводимо ...).

2. P = "следствие". Тогда переменной x14 присваивается программная переменная, значением которой в контексте срабатывания приема является текущая задача. Если компилируется прием сканирования задачи, то текущая задача определяется переменной "x1"; в случае пакетного оператора она находится с помощью оператора "текущаязадача(...)". Переменной x15 присваивается обращение к процедуре "следствие", которая будет выполнять проверку истинности антецедента. Переменной x16 присваивается оператор, устанавливающий уровень обращения к вспомогательной задаче на доказательство, решаемой внутри указанной процедуры. Далее x16 и x15 регистрируются в компилируемой программе; если имелся указатель приема "Обрыв(...)", относящийся к v, то перед x15 вставляется оператор "лимит(...)", а после x15 - оператор "Обрыв".

3. В оставшихся случаях компилятор добавляет к программе операторы "уровеньобращения(N)", "следствие(x10 x11 ...)". Здесь N определяется в зависимости от заголовка установки P.

19.10.12 Установка на идентификацию "новаргумент"

Установка на идентификацию (новаргумент A_1 A_2) возникает по указателям приема "новаргумент(f x N)" либо "функаргумент(f x a N)". Эти указатели определяют такую идентификацию выражения $f(t)$, при которой все вхождения переменной x в результат обработки нормализатором N идентифицирующего терма, сопоставляемого данному выражению, расположены только внутри вхождений подтермов, соответствующих t. Перед обработкой установки все переменные, входящие в t, должны уже быть идентифицированы. Допускается обобщение на многомерный случай (x,t становятся выражениями вида "набор(...)"). В случае указателя "функаргумент" добавляется набор a соответствующих отдельным элементам t_i набора t индикаторов связанных переменных. Каждый такой индикатор - 0 либо указание на переменную выражения t_i, при идентификации которой учитывается возможность переобозначения ее как связанной переменной.

Элементы A_1, A_2 установки "новаргумент" суть: программное выражение для терма $f(t)$, определяемого указателем приема, а также вхождение этого терма в теорему приема. Обработка установки начинается с контрольной точки "прием(137)". Уровень компиляции x7 при этом равен 6. Переменной x10 присваивается вхождение терма $f(t)$ в теорему приема, а переменной x11 - переменная f. Находится указатель приема x12, породивший рассматриваемую установку. Переменной x13 присваивается

список переменных выражения x из указателя x12. Предполагается, что это выражение - либо отдельная переменная, либо терм "набор(...)" с операндами, представляющими собой попарно различные переменные. Далее компиляция разветвляется.

Случай указателя "новаргумент"

В этом случае указатель x12 имеет вид "новаргумент(f x N)" и относится к идентификации выражения $f(t)$. Переменной x14 присваивается терм t; определяется программное выражение x15 для t. Находится программное выражение x16 для списка переменных x13. Если N - название невырожденного нормализатора (т.е. отлично от символа "фикс"), то корректируется программное выражение A_1 из установки - оно начинает определять результат применения к терму $f(t)$ нормализатора N. Заметим, что при обращении к N ему передается комментарий (новаргумент P X), где P - терм t (если t - набор термов, то берется первый терм этого набора); X - список переменных x13. После коррекции A_1 - откат к переходу через "ветвь 3".

Если t, x - не векторные, то в программу заносится оператор "новаргумент(...)", проверяющий условие вхождения переменной x в идентифицирующий терм A_1 только внутри t и присваивающий новой программной переменной результат замены в идентифицирующем терме всех вхождений t на x. Этот результат будет использоваться в качестве шаблона для формирования новых термов вида $f(...)$. Он регистрируется в информационном элементе (функция ...), что и завершает идентификацию в данном подслучае.

Для векторных t, x действия аналогичны, но вместо оператора "новаргумент" используется оператор "Новаргумент".

Случай указателя "функаргумент"

В этом случае x12 имеет вид "функаргумент(f x a N)". Переменной x15 присваивается набор программных выражений для компонент вектора t; переменной x16 - программное выражение для набора переменных x. Как и в предыдущем случае, при невырожденном N корректируется программное выражение A_1 из установки. Затем - откат к переходу через "ветвь 1". Здесь переменной x17 присваивается набор программных выражений для переменных списка a, и переход через "иначе 1". Дальнейшие действия аналогичны разобранным в подслучае указателя "новаргумент", но вместо оператора "новаргумент" применяется оператор "функаргумент".

19.10.13 Установка на идентификацию "развертка"

Рассматриваемая установка имеет вид (развертка v A_1 A_2). Здесь v - вхождение в теорему приема терма, идентифицируемого по указателю "развертка(...)". Такой терм имеет вид операции над конечным семейством, которое должно быть определено по конкретному набору идентифицирующих термов. Дополнительно разрешается также идентификация кванторной импликации как конъюнкции и описателя "отображение" как набора. A_1 - программное выражение для вхождения (при $A_2 = 0$) либо терма (при $A_2 = 1$) для идентификации с v. Возможен также специальный случай, когда A_1 - информационный элемент (набороперандов B_1 B_2), дающий программное выражение B_2 для набора термов, среди которых осуществляется отбор идентифицируемых с v элементов. В этом случае $A_2 = 2$. Во всех случаях идентификация v выполняется за счет просмотра заданного списка термов и извлечения

из него элементов, составляющих "таблицы" значений идентифицируемых функциональных переменных, встречающихся в v. Заметим, что установки на идентификацию (развертка ...) не вводятся непосредственно при обращении к процедуре "идентификатор", а возникают в процессе реализации этой процедуры.

Обработка установки начинается с контрольной точки "прием(167)". Уровень компиляции x7 равен 2, так как фактически выполняется не очень трудоемкая непосредственная идентификация, аналогичная имевшей место при установке "операнд". Переменной x10 присваивается идентифицируемый подтерм. Если все свободные переменные терма x10 уже идентифицированы, причем этот терм не имеет вида F(отображение(x $F(x)$ $t(x)$)), где $t(x)$ содержит связанные переменные, то начиная с контрольной точки "прием(212)" реализуется простейший подслучай идентификации.

Переменной x11 присваивается заголовок терма x10. Этот заголовок - либо квантор общности, либо символ одноместной операции, применяемой к семейству, заданному термом "отображение(...)". По нему определяется тот символ x12 коммутативно-ассоциативной операции, которая соответствует данной операции над семействами. В случае квантора общности x12 - логическая связка "и"; в случае описателя "отображение" x12 - символ "набор"; в прочих случаях этот символ определяется по x11 с помощью справочника "развертка". В компилируемую программу заносится оператор, присваивающий новой программной переменной "начало(x3)" набор N_1 x12 - членов идентифицирующего терма, соответствующего x10. Находится также программное выражение x14 для набора N_2 x12 - членов терма x10 (это возможно, так как все свободные переменные в x10 уже идентифицированы). Если v - корневое вхождение идентифицируемой части теоремы, причем компилируется прием замены, то проверяется включение N_2 в N_1, и формируется информационный элемент (корни ...), определяющий остаток не использованных корневых операндов идентифицирующего терма. В противном случае вводятся операторы, проверяющие совпадение множеств N_1, N_2.

Если не все переменные x10 уже идентифицированы, либо нарушается приведенное выше условие на отсутствие связанных переменных в $t(x)$, то переход через "иначе 2" к контрольной точке "прием(213)". Здесь снова находится символ ассоциативно-коммутативной операции, соответствующей операции над семействами, являющейся заголовком терма x10; он присваивается переменной x11. Перед контрольной точкой "прием(214)" инициализируются нулями переменные x12, x13 и x14. Им будут переприсваиваться, соответственно: варьируемая в цикле идентификации теоремная переменная, играющая роль номера элемента семейства; теоремный терм для длины семейства; теоремный терм для элемента семейства. После оператора "ветвь 1" начинается анализ подслучаев для определения x12, x13, x14. Здесь рассматриваются следующие подслучаи:

1. x10 имеет вид "длялюбого(i если принадлежит(i номера(1 n))то F)". Тогда x12 полагается равным переменной i; x13 - терму n; x14 - терму F.

2. x10 имеет вид "отображение(i принадлежит(i номера(1 n) t))". Тогда x12 полагается равным переменной i; x13 - терму n; x14 - терму t.

3. x10 имеет вид F(отображение(i и(целое(i) $1 \leq i$ $i \leq n$)t)); вместо "целое" здесь допускается использование символа "натуральное". Тогда x12 полагается равным переменной i; x13 - терму n; x14 - терму t.

4. x10 имеет вид "длялюбого(i j если принадлежит(i номера(1 n)) принадлежит(j номера(1 m)) то F). Это - случай двухпараметрической идентификации конъюнкции утверждений. Переменной x12 здесь присваивается i; переменной x13 - терм n; переменной x14 - терм F. Кроме того, переменной x16 присваивается j, а переменной x17 - m. После перехода через "ветвь 2" попадаем на ветвь программы, начинающуюся с контрольной точки "прием(261)". Она содержит в себе всю процедуру идентификации для данного подслучая и будет рассмотрена ниже в отдельном подразделе.

После определения значений x12, x13, x14 - откат к переходу через "ветвь 1" (кроме указанного выше двухпараметрического подслучая). Далее компиляция разветвляется - в зависимости от того, идентифицирована ли уже длина набора n.

Случай не идентифицированной длины набора

В этом случае проверяется, что x13 - еще не идентифицированная переменная. Если x11 - символ "набор", то вводится оператор, проверяющий, что идентифицирующий терм имеет заголовок "набор"; находится программное выражение x16 для числа корневых операндов этого терма, и регистрируется идентификация x13 с помощью x16. Затем - откат к переходу через "ветвь 1" для рассмотрения подслучая идентифицированной длины набора. Если же x11 отлично от символа "набор", то откат к переходу через "ветвь 2".

Здесь размещена контрольная точка "прием(215)". Далее x15 - переменная для длины набора. Переменной x16 присваивается программное выражение для набора корневых x11- членов идентифицирующего терма.

Терм x14 является шаблоном идентификации - он будет сравниваться с элементами набора x16 при варьировании переменной x12. Не идентифицированные встречающиеся в x14 функциональные переменные вида "значение(f x12)" будут порождать в цикле изменения x12 некоторые наборы термов. Переменной x17 присваивается список всех таких переменных f. Перед рассмотрением общего случая выделяются два простейших подслучая. В первом из них x14 само является функциональной переменной "значение(f x12)". Тогда вводится информационный элемент (вектор f x16), идентифицирующий f непосредственно с набором x16. Одновременно f задается и в формате терма. Кроме того, x15 идентифицируется с длиной набора x16. Во втором случае x14 имеет вид "значение(значение(f x12)X)". Это - случай идентификации набора функций, зависящих от X (например, при решении системы дифференциальных уравнений). Здесь используется информационный элемент (Набор f), указывающий на несколько нестандартную регистрацию идентификации f с помощью информационных элементов (терм f ...) и (функция f ...).

Если x14 не имеет указанного простейшего вида, то - откат к переходу через "ветвь 1". Здесь проверяется, что все переменные терма x14, отличные от варьируемой переменной x12 и функциональных переменных списка x17, уже идентифицированы. В программу приема заносятся операторы, инициализирующие пустыми словами накопители для идентификации переменных x17, причем составляется список x18 программных переменных, используемых в качестве накопителей. Если набор идентифицирующих термов представляет собой остаток операндов некоторой операции, то вводится накопитель x19 остатка тех же операндов после цикла варьирования x12.

Находится список x20 вхождений в терм x14 всевозможных функциональных переменных "значение(f x12)" для f из x17; проверяется, что каждое вхождение x12

в x14 расположено внутри одного из вхождений списка x20. Затем находится результат x21 замены в x14 каждой функциональной переменной "значение(f x12)" на обычную переменную f. Это делается для того, чтобы на текущем шаге цикла варьирования x12 идентифицировать f как обычную переменную и занести результат идентификации в соответствующий накопитель списка x18. Для идентификации x21 создается вспомогательный программный блок x23. Предварительно переменной x22 присваивается не более чем одноэлементный список операторов, идентифицирующих заголовок x21. Если имеется информационный элемент (заголовок ...), указывающий, что x21 может идентифицироваться с термом, имеющим другой заголовок, то x22 оставляется пустым. Очередной идентифицирующий терм, извлекаемый из набора x16 для идентификации x21, будет присваиваться программной переменной "начало(x3)". Это присвоение является по отношению к блоку x23 внешним, так что в его программу сначала заносятся лишь x22 и оператор, присваивающий очередной программной переменной корневое вхождение в текущий идентифицирующий терм. В x23 заносится информационный элемент (операнд ...), связывающий корневые вхождения терма x21 и текущего идентифицирующего терма; переменной x24 присваивается список установок на идентификацию, состоящий из единственной установки (операнд ...) такого же вида. Выполняется коррекция тех информационных элементов программного блока x23, которые ссылались на вхождения в теорему приема, расположенные внутри x14. Вместо этих вхождений подставляются соответствующие вхождения в x21.

Далее - переход через "ветвь 1", и после контрольной точки "прием(217)" - обращение к процедуре "идентификатор". После того, как в программном блоке x23 возникла программа для идентификации x21, к ней добавляются операторы, регистрирующие в накопителях списка x18 термы, идентифицированные с соответствующими переменными списка x17. Переменной x26 присваивается результат навешивания квантора существования по всем новым переменным блока x23 к конъюнкции операторов его программы. Наконец, создается и заносится в программный блок x3 реализующая цикл идентификации кванторная импликация x27. Ее антецедент перечисляет термы списка x16, присваивая текущий такой терм программной переменной "начало(x3)". Консеквентом служит либо x26, либо, если был введен накопитель остатка идентифицирующих операндов, дизъюнкция, регистрирующая в случае ложности x26 неиспользованный текущий идентифицирующий терм в списке x19.

Если был введен накопитель остатка идентифицирующих операндов, то в программу заносится ряд дополнительных операторов. Во-первых, при отсутствии информационного элемента, разрешающего идентифицировать v с пустым списком термов, вводится оператор, проверяющий непустоту первого накопителя списка x18. Корректируется указанный в установке на идентификацию информационный элемент (набороперандов ...). Если v - последний не идентифицированный операнд этого элемента, то вводится оператор, проверяющий пустоту остатка x19.

Далее вводятся информационные элементы, регистрирующие результаты идентификации переменных списка x17. Для каждой переменной создается два таких элемента - (терм ...) и (вектор ...). В заключение регистрируется результат идентификации длины набора.

Случай идентифицированной длины набора

Рассмотрение этого подслучая начинается с контрольной точки "прием(223)". Перед этим переменной x15 присваивается программное выражение для длины набо-

ра. Основное отличие от случая не идентифицированной длины набора состоит в том, что знание диапазона изменения варьируемой переменной x12 позволяет явно использовать текущее значение этой переменной на каждом шаге цикла идентификации. Напомним, что в предыдущем подразделе текущий идентифицируемый терм x21 не содержал варьируемой переменной; здесь же он, вообще говоря, будет содержать эту переменную. В остальном действия аналогичны.

Переменной x16 присваивается программное выражение для набора идентифицирующих термов; x17 присваивается список всех таких переменных f, что x14 имеет подтерм вида $f(X)$, где X - переменная x12 либо набор переменных, включающий x12. Если x14 само имеет вид $f(X)$, то идентификация сводится к созданию информационного элемента (вектор f ...). Иначе формируется набор x18, сопоставляющий каждой многоместной функциональной переменной f списка x17 единицу, а каждой одноместной - 0. Проверяется, что все свободные переменные терма x14, не входящие в x17 и отличные от x12, уже идентифицированы. В компилируемую программу заносятся операторы, инициализирующие пустыми словами накопители наборов, идентифицирующих переменные списка x17. Переменной x19 присваивается список этих накопителей (программных переменных). Если x11 отлично от символа "набор", то вводится накопитель x20 использованных термов идентифицирующего набора. Находится результат x23 замены в x14 каждой одноместной функциональной переменной $f(X)$ на f, а каждой многоместной - на результат исключения переменной x12 из списка аргументов X (имеются в виду f из x17).

Если x17 состоит из единственной переменной, выделенной указателем приема "коэфф", то возникает особый случай, который будет рассмотрен ниже в специальном подразделе (программа здесь начинается с контрольной точки "прием(254)").

Иначе - переход через "ветвь 1" к общему случаю, где создается вспомогательный программный блок x24 для обработки текущего шага идентификации. Этому программному блоку передаются информационные элементы, определяющие текущее значение варьируемой переменной x12. Если x11 - символ "набор", то идентифицирующий терм для x23 берется как x12-й элемент идентифицирующего набора; в противном случае он выбирается произвольным образом среди элементов идентифицирующего набора, не зарегистрированных в x20. Для всех многоместных функциональных переменных f списка x17 в новый программный блок заносится информационный элемент (усмматрица f). Так как многоместная переменная должна будет идентифицироваться с матрицей, то на текущем шаге остаточную функциональную переменную, имеющая на один аргумент меньше, нужно идентифицировать не с термом, а с подматрицей (например, с набором). На это и указывают элементы (усмматрица ...).

Реализуется цикл коррекций информационных элементов блока x24, учитывающих переход к новому идентифицируемому терму x23 и предпринятые вычеркивания вхождений x12 в функциональные переменные. Создается одноэлементный список x25 установок на идентификацию для x24. Если терм x23 сам должен идентифицироваться с указателем "развертка", то вместо установки (операнд ...) в x25 вводится установка (развертка ...). После контрольной точки "прием(259)" происходит обращение к процедуре "идентификатор" для блока x24. К концу программы блока x24 присоединяются операторы, пополняющие накопители идентифицирующих наборов для переменных списка x17. Если был введен накопитель x20, то в него заносится ссылка на использованный терм идентифицирующего набора. Далее находится результат x27 навешивания квантора существования по всем новым программным переменным блока x24 на конъюнкцию операторов этого блока. Наконец, создается

и регистрируется в программном блоке x3 реализующая цикл идентификации кванторная импликация x28. Если эта импликация оказывается чрезмерно громоздкой, то цикл реализуется с помощью оператора "ветвь".

Если v представляло собой корневое вхождение заменяемого терма, то вводится информационный элемент (корни ...), определяющий остаток неиспользованных корневых операндов идентифицирующего терма. Если остаток должен быть пуст, то вводится оператор, проверяющий данное условие. Наконец, результаты идентификации переменных списка x17 регистрируются в информационных элементах (вектор ...) либо (элементыматрицы ...).

Случай указателя "коэфф"

Напомним, что указатель "коэфф(K)" применяется для идентификации вектора либо матрицы K, элементы которых суть коэффициенты членов заданного вида, выделяемых в многоместной коммутативно-ассоциативной операции f. Идентифицируемый с этой операцией по указателю "развертка" терм имеет вид F(отображение(x $P(x)$ $g(K(Y)Z)$)); F - обобщение операции f на произвольные конечные семейства. Для каких-либо комбинаций индексов Y могут отсутствовать члены $g(K(Y)Z)$, и тогда элемент вектора (матрицы) идентифицируется с нулем операции g, являющимся одновременно единицей операции f.

Учет указателя "коэфф" начинается с упоминавшейся выше контрольной точки "прием(254)". Для обработки текущего шага идентификации здесь вводится вспомогательный программный блок x25, которому передается информация о текущем значении варьируемой переменной x12. В программу этого блока включаются операторы, обеспечивающие просмотр элементов списка x16 идентифицирующих термов (операнды операции f), не учтенных в накопителе x20 уже использованных для идентификации термов. Программной переменной "плюссимв(начало(x3)3)" присваивается корневое вхождение выбранного в списке x16 терма. Ссылки из информационных элементов блока x25 на вхождения расположенные внутри x14, заменяются ссылками на соответствующие вхождения в x23. Проверяется, что в результате элиминации перечисляемых индексов идентифицируемый терм x23 приобрел вид $g(K\ Z)$, т.е. функциональная переменная K превратилась в обычную переменную. Добавляются информационный элемент, указывающий, что для K допустимо вырожденное значение - единица операции g, а также, если он нужен, информационный элемент о перенесении на коэффициент K знака одночлена. После контрольной точки "прием(255)" происходит обращение к процедуре "идентификатор". Переменной x29 присваивается программное выражение для коэффициента K в случае успешной идентификации.

Реализующая цикл идентификации кванторная импликация x32 перечисляет значения варьируемой переменной x12 из заданного диапазона. Для текущего такого значения она сначала присваивает накопителю коэффициента K вырожденное нулевое значение, обращаясь сразу после этого к кванторной импликации x31, переприсваивающей этому накопителю значение x29, определяемое с помощью программы блока x25 и регистрирующей в x20 использованный идентифицирующий терм. После x31 в консеквенте оператора x32 размещен оператор, пересылающий текущее значение K в накопитель вектора значений K.

Наконец, выполняются действия по регистрации результатов идентификации, аналогичные рассмотренным выше.

Двухпараметрическая идентификация конъюнкции

В рассмотренных выше случаях указатель "развертка" относился к однопараметрическому семейству - единственная варьируемая переменная принимала целочисленные значения из заданного промежутка. При идентификации конъюнкций утверждений возникла надобность использовать двухпараметрическое семейство, где варьируются две переменные. Такое семейство задается теоремным термом вида "длялюбого(i j если принадлежит(i номера(1 n))принадлежит(j номера(1 m)) то $F(i,j)$)".

Выше мы уже дошли до точки программы, где переменные i,j были присвоены программным переменным x12 и x16; границы диапазона n,m - переменным x13 и x17, а идентифицируемый в цикле терм $F(i,j)$ - переменной x14. Дальнейшие действия начинаются с контрольной точки "прием(261)". Прежде всего, находятся программные выражения x18 и x19 для численных значений n и m. Вводится оператор, присваивающий новой программной переменной список идентифицирующих утверждений, причем эта программная переменная становится значением переменной x20. Находится список x21 всех таких переменных f, для которых в x14 встречается не идентифицированная функциональная переменная $f(i,j)$, и проверяется, что все свободные переменные терма x14, не входящие в x21 и отличные от i,j, уже идентифицированы. Вводятся накопители матриц (представленных как наборы наборов термов) для всех переменных списка x21, причем переменной x23 присваивается список этих накопителей. Инициализируется пустым словом также накопитель x24 вхождений в список x20 использованных при идентификации термов. Определяется результат x27 замены в терме x14 всех функциональных переменных $f(i,j)$ на обычные переменные f. Он будет играть роль идентифицируемого терма на текущем шаге цикла идентификации.

Для организации цикла по переменной i создается список операторов x29, состоящий из оператора "Номера(...)", перечисляющего значения i, а также из операторов, инициализирующих пустыми словами накопители очередной строки для матриц списка x21. Переменной x28 при этом оказывается присвоен список, начинающийся с программной переменной, значением которой служит i, и перечисляющий далее накопители строк.

Для сохранения значения j резервируется программная переменная, присваиваемая x30. Создается вспомогательный программный блок x31, которому передается информация о текущих значениях i,j. Программа этого блока состоит из операторов, перечисляющих еще не использованные идентифицирующие термы списка x20, имеющие тот же заголовок, что и x27. Выполняется обращение к процедуре "идентификатор", создающей в программном блоке x31 процедуру для идентификации x27 с выбранным идентифицирующим термом. К концу этой программы присоединяются операторы, пополняющие накопители строк очередными значениями, а также оператор, регистрирующий в накопителе x24 использованное вхождение идентифицирующего терма в список x20.

Наконец, предпринимается сборка кванторной импликации x37, выполняющей цикл идентификации. Ее антецеденты - операторы списка x29, обеспечивающие внешний цикл по i. Внутренний цикл по j реализуется кванторной импликацией x35, расположенной в консеквенте x37. Консеквентом импликации x35 служит результат навешивания квантора существования на конъюнкцию операторов программного блока x31. После x35 в консеквенте импликации x37 размещены операторы, регистрирующие накопители строк в накопителях матриц. Компиляцию завершает регистрация

идентификации переменных списка x21 в информационных элементах (элементы-матрицы ...).

19.10.14 Установка на идентификацию "транзитпереход"

Установка (транзитпереход v) ссылается на антецедент v, выделенный указателем приема "транзитпереход". Такой антецедент имеет вид $f(A_1\ A_2)$, где f - транзитивное бинарное отношение. При его обработке сначала идентифицируется один из операндов A_1, A_2, а другой доопределяется путем рассмотрения имеющихся в контексте идентификации цепочек утверждений вида $f(B_i\ B_{i+1})$. При этом используется вспомогательный оператор "транзитпереход".

Рассмотрение установки начинается с контрольной точки "прием(173)". Уровень x7, на котором происходит обработка, равен 5. Определяется тот операнд A_i, который представляет собой уже идентифицированную переменную. x13 становится равно программному выражению для значения A_i. Для перечисления возможных значений другого операнда создается обращение x16 к оператору "транзитпереход"; это обращение заносится в компилируемую программу. Если компилируется прием пакетного оператора, то добавляется проверка непересечения цепочки утверждений, использованных процедурой "транзитпереход" при определении противоположного операнда, с утверждениями, зарегистрированными в комментарии (исключение ...) к пакетному оператору. Наконец, для регистрации идентификации противоположного операнда осуществляется обращение к процедуре "учетоперанда".

19.10.15 Установка на идентификацию "группировка"

Установка (группировка $v\ x$) возникает при компиляции приема замены, имеющего указатель "набор(второйтерм)". Такой прием основан на теореме вида $f(P(a)\ P(b)) = P(h(a\ b))$, $f(P(a\ c)\ P(b\ c)) = P(h(a\ b)\ c)$ и т.п., где операции f, h являются коммутативными и ассоциативными. При компиляции теорема обобщается на произвольное число операндов этих операций, причем реализуется в режиме группировки операндов операции f (т.е. в указанном случае замена выполняется слева направо). Предполагается, что в преобразовании участвуют все корневые f - члены идентифицирующего терма. В установке v является корневым вхождением заменяемой части теоремы приема; x - программной переменной, значением которой служит корневое вхождение идентифицирующего терма.

Обработка установки начинается с контрольной точки "прием(28)". Сначала извлекается информационный элемент (перечень $A_1\ A_2\ A_3\ A_4$), сформированный на предварительном этапе компиляции. У него A_1 - вхождение в заменяющий терм теоремы выражения $h(a\ b)$; A_2 - вхождение корневого операнда заменяемого терма, используемого в качестве шаблона идентификации для выделения очередного группируемого f- члена. Этот шаблон может не в точности совпадать с термом $P(a)$ либо $P(a\ c)$, а содержать его в качестве своего подтерма. A_3 - переменная a, используемая в шаблоне для идентификации с операндами операции $h(...)$; A_4 - сначала 0, а впоследствии будет изменено на программное выражение для результата группировки $h(...)$. A_2 и A_3 присваиваются, соответственно, переменным x11 и x12. Переменной x13 присваивается вхождение переменной a в шаблон идентификации; переменной x14 - вхождение внешней операции P; переменной x15 - символ этой операции. Далее компиляция разветвляется.

Коммутативно-ассоциативная операция P

В этом подслучае операция P - коммутативно-ассоциативная и имеет ровно два операнда - переменные a, b, причем вхождение переменной b в шаблон идентификации единственное. Рассмотрение подслучая начинается с контрольной точки "прием(29)". Если для выделения общей части подтермов с заголовком P предусмотрена специальная процедура (она находится с помощью справочника "пересечениесписков"), то ее заголовок присваивается переменной x16. Если же применение такой процедуры для подтерма x14 заблокировано указателем приема, то x16 переприсваивается 0. Выбирается не используемая в описании приема переменная x17, и находится результат x18 замены на нее вхождения x14 в шаблон A_2. Инициализируются нулями переменные x20 и x21. Значение первой из них будет изменено на программное выражение для набора наборов P-членов, соответствующих корневым f - членам идентифицирующего терма. Значением второй станет программное выражение для пересечения наборов P - членов. Для определения x20 и x21 рассматриваются следующие два подслучая:

1. x18 - однобуквенный терм, состоящий из переменной x17 (контрольная точка "прием(30)"). Это означает, что операция P в шаблоне A_2 была корневой. Прежде всего, рассматривается подслучай приема нормализатора, имеющего указатель "попытка(s теквхожд)". Вводятся оператор, проверяющий отсутствие комментария (s t) , где t - заменяемый подтерм, а также заготовка оператора "ветвь" для регистрации данного комментария при откате. Затем - откат к переходу "ветвь 3". Создаются вспомогательные операторы x22 и x23, которые будут использоваться при формировании x20 и x21. Первый из них нужен для определения набора P - членов на текущем шаге цикла; второй - определяет этот набор для первого корневого операнда идентифицирующего терма.

   После контрольной точки "прием(33)" выполняется учет указателя приема "заменазнака(...)". Это происходит лишь в случае, когда x16 равно 0, т.е. не применяется специальная процедура для определения общей части двух термов с заголовком P. Если такая процедура использовалась бы, то учет возможности занесения внешнего знака внутрь операции P предпринимался бы внутри нее. Проверяется, что переменная x12 выделена как указателем "заменазнака(s ...)", так и указателем "единица(e ...)". Создается константный терм x30 вида $s(e)$ (в случае операции умножения - минус единица), и выражения x22, x23 корректируются так, чтобы в случае внешнего знака s рассматриваемого корневого f - члена идентифицирующего терма этот знак заносился внутрь списка P-операндов в виде $s(e)$. Терм x30 сохраняется для дальнейшего использования, будучи переприсвоен переменной x24.

   Затем создается набор x25 операторов, просматривающих корневые f - члены идентифицирующего терма для определения набора наборов их P - членов и пересечения этих наборов. Если x16 равно 0 (контрольная точка "прием(34)"), то пересечение определяется оператором "пересечениесписков", иначе (контрольная точка "прием(35)") - специальной процедурой x16. Операторы набора x25 регистрируются в компилируемой программе, и происходит переприсвоение значений x20, x21. Кванторная импликация из x25, реализующая цикл просмотра, на каждом шаге проверяет непустоту пересечения. В частности, оно считается пустым, если состоит только из "минус единицы" - терма x24.

2. x18 - неоднобуквенный терм (контрольная точка "прием(37)"). Тогда вводится оператор, проверяющий, что заголовки всех корневых f - членов идентифицирующего терма совпадают с заголовком шаблона x18. После него размещается оператор, выделяющий вхождение первого корневого операнда идентифицирующего терма. С помощью процедуры "идентификатор" предпринимается формирование операторов, идентифицирующих первый корневой операнд с шаблоном x18. После них помещается оператор, определяющий набор P - членов терма, идентифицированного с переменной x12 шаблона x18. Переменная, которой присвоен этот набор, и есть новое значение x21. В ней будет накапливаться пересечение всех списков P - членов термов, идентифицируемых с переменной x12 шаблона x18 в цикле просмотра корневых операндов идентифицирующего терма. Вводится оператор, инициализирующий программную переменную для накопителя набора наборов P - членов; этот накопитель сразу же переприсваивается переменной x20. Для компиляции шага идентификации вводится вспомогательный программный блок x27. Предпринимается обращение к процедуре "идентификатор" для блока x27, которое дает программу идентификации шаблона x18 с текущим корневым операндом идентифицирующего терма. После этой программы помещается оператор, присваивающий некоторой новой программной переменной набор P - членов терма, идентифицированного с x12. Далее добавляется оператор, находящий пересечение этого набора с накопителем x21 (применяется либо оператор "пересечениесписков", либо специальный оператор x16). Фактически, в зависимости от x16, компиляция здесь разветвляется, однако дальнейшие действия идентичны - на основе программы блока x27 создается оператор x29, реализующий цикл просмотра корневых операндов идентифицирующего терма (начиная со второго операнда) и коррекции накопителей x20, x21.

После определения x20, x21 - откат к переходу через "ветвь 1". Здесь предпринимается коррекция указателей нормализации, относящихся к подтерму $h(\ldots)$ заменяющего терма - они переключаются на переменную x17, которая будет впоследствии обозначать указанный подтерм. Программной переменной x22 присваивается переменная b подтерма $P(a\ b)$ - она будет идентифицироваться с пересечением наборов P - членов. Если прием имеет указатель "перечень(...)", определяющий идентификацию b не со всем пересечением, а лишь с теми его термами, которые удовлетворяют условию, приведенному в указателе (контрольная точка "прием(93)"), то x21 заменяется на программное выражение для этого подмножества пересечения.

Затем - откат к переходу через "ветвь 1", где регистрируется результат идентификации переменной b. Вводится заготовка x24 программного выражения для текущего операнда подтерма $h(\ldots)$ заменяющего терма. Если имелся указатель нормализации, относящийся к x12, то по нему здесь же выполняется коррекция текущего операнда x24 - он обрабатывается соответствующим нормализатором.

Наконец, создается группа операторов x25, в которой предпринимается цикл просмотра набора x20 наборов P- членов для определения остатков этих наборов после удаления их общей части x21. Каждый такой остаток является входным данным для программного выражения x24, определяющего очередной операнд подтерма $h(\ldots)$. Эти операнды соединяются операцией h, и результирующий терм присваивается программной переменной "плюссимв(x23 1)". Результат идентификации подтерма $h(\ldots)$ регистрируется в упоминавшемся выше информационном элементе (перечень ...), а также в информационном элементе (терм x17 ...). Фактически компиляция распада-

ется здесь на две ветви - при x16 = 0 используется оператор "вычеркивание", иначе остатки наборов определяются процедурой x16.

Не коммутативно-ассоциативная операция P

Если операция P не является коммутативно-ассоциативной, то первый корневой операнд идентифицирующего терма позволяет идентифицировать все переменные шаблона, кроме переменной x12, соответствующей группируемым под операцию h подтермам. После этого просматриваются оставшиеся корневые операнды, идентифицирующие группируемые значения x12.

Рассмотрение подслучая начинается с контрольной точки "прием(145)". Переменной x16 присваивается одноэлементный набор установок на идентификацию, используемых при сопоставлении шаблону A_2 первого корневого операнда идентифицирующего терма. Если имеется указатель "заменазнака", определяющий занесение внутрь текущего корневого f - члена заданного символа одноместной операции, то этот символ присваивается переменной x17. В компилируемую программу включается оператор, проверяющий, что каждый корневой операнд идентифицирующего терма имеет (с точностью до отбрасывания "знака" x17) тот же заголовок, что и шаблон A_2. После этого первой неиспользуемой программной переменной присваивается вхождение первого корневого операнда идентифицирующего терма. Затем предпринимается обращение к процедуре "идентификатор", позволяющее создать программу идентификации шаблона с первым корневым операндом. Переменной x19 присваивается программное выражение для значения переменной x12, определившегося в результате данной идентификации. Фактически это первый из операндов группирующей операции $h(\ldots)$. x20 становится накопителем группируемых операндов.

Далее вводится вспомогательный программный блок x23 для идентификации шаблона с остальными корневыми операндами идентифицирующего терма. Из него удаляются информационные элементы блока x3, определяющие идентификацию переменной x12. С помощью справочника "прогрблок" повторно обрабатываются относящиеся к x12 указатели приема "заменазнака", чтобы они были учтены в x23. Происходит обращение к процедуре "идентификатор" для вспомогательного блока (контрольная точка "прием(176)"). Переменной x24 присваивается программное выражение для подтерма, идентифицированного с переменной x12. Создается оператор x26, представляющий собой конъюнкцию операторов блока x23 и оператора, регистрирующего в накопителе x20 очередное значение x24. Наконец, вводится кванторная импликация x27, реализующая цикл просмотра оставшихся корневых операндов идентифицирующего терма. После этого программное выражение для группирующего терма $h(\ldots)$, определяемого по накопителю x20, регистрируется в последнем разряде информационного элемента (перечень ...).

19.10.16 Установка на идентификацию "усм"

Установка (усм v) ссылается на антецедент v, который должен обрабатываться идентифицирующим оператором. Такие операторы представляют собой сильно упрощенные и ускоренные версии проверочных операторов и синтезаторов. Они извлекают результат почти непосредственно из имеющихся утверждений контекста, используя лишь малое число нетрудоемких логических переходов. Для ускоренного поиска посылок идентифицирующие операторы пользуются специальными древовидными структурами данных, хранящимися в комментариях (списокпосылок ...), (отр

...), (выражение ...). С целью ускорения применяются также комментарии (Буфер ...), сохраняющие результаты нескольких предшествующих обращений к оператору. Жесткие органичения на быстродействие идентифицирующих операторов продиктованы тем, что они обычно реализуются на ранних этапах попытки применения приема, и поэтому поток обращений к ним достаточно интенсивен. Главным образом, идентифицирующие операторы применяются пока в приемах по геометрии.

Программы идентифицирующих операторов реализуются непосредственно на ЛОСе, а чтобы компилятор мог воспользоваться ими, служат справочники "усм" и "См". При обращении к справочнику "усм" задается заголовок идентифицирующего оператора f. Результатом обращения служит набор (вход($A_1 \dots A_m$) выход($B_1 \dots B_n$) усм(T)$C_1 \dots C_k$). Здесь T - шаблон обращения к оператору, т.е. конъюнкция обрабатываемых им антецедентов. Заметим, что один оператор может обрабатывать сразу несколько различных антецедентов теоремы приема, выделенных указателем "усм". Разбиение антецедентов на группы, обрабатываемые одним и тем же идентифицирующим оператором, предпринимается компилятором самостоятельно. $A_1, \dots, A_m$ - подтермы шаблона, поступающие на входы оператора; $B_1, \dots, B_n$ - подтермы шаблона, получаемые как значения выходных переменных оператора. Обращение к оператору имеет вид $f(x_1\ x_2 \dots x_{m+1}\ y_1 \dots\ y_n)$, причем x_1 - текущая задача на исследование либо на доказательство, в контексте списка посылок которой работает оператор; $x_2, \dots, x_{m+1}$ соответствуют входным термам $A_1, \dots, A_m$; $y_1, \dots, y_n$ соответствуют выходным термам $B_1, \dots, B_n$. $C_1, \dots, C_k$ суть информационные элементы, уточняющие формат идентифицирующего оператора. Типы этих элементов перечислены в справочной информации для символа "усм". Сразу выделим из них элемент "уровень(u)", который указывает на приоритет использования идентифицирующего оператора при компиляции (в первую очередь рассматриваются операторы с меньшими значениями u). Такие приоритеты позволяют во многих случаях уменьшать перебор, идентифицируя в первую очередь объекты, определенные с меньшей степенью неоднозначности. Например, выгоднее сразу идентифицировать точку как пересечение двух прямых, чем просматривать все точки прямой либо все прямые, проходящие через точку.

Чтобы найти заголовки идентифицирующих операторов, которые могли бы позволить обработать заданный антецедент, применяется справочник "См". Перед обращением к нему вводится одноэлементный набор, состоящий из пустого накопителя. Этот набор передается справочнику как значение его входной переменной x1. Текущий символ обращения - какой-либо выделенный в антецеденте логический символ s. Справочник "См" заполняет накопитель x1 заголовками всех тех идентифицирующих операторов, у которых в их шаблоне символ s был выделен в качестве точки привязки. Выбор точки привязки выполняется компилятором, создающим прием справочника "См", достаточно произвольным образом. Он не очень существенен - важно лишь при обработке антецедента просмотреть все входящие в него логические символы.

Обработка установки (усм v) начинается с контрольной точки "прием(121)". Уровень x7, на котором будет происходить компиляция, определится впоследствии, по информационному элементу "уровень(...)" используемого идентифицирующего оператора. Пока же проверяется отсутствие установок на идентификацию (корень ...) и (операнд ...), так как они дают менее трудоемкую идентификацию и должны обрабатываться в первую очередь. Учитываются также указатели приема "конец(...)". Если некоторый такой указатель ссылается на текущий антецедент v, в то время как имеются установки (усм ...) либо (извлекается ...), не относящиеся к анте-

цеденту, выделенному указателем "конец(...)", то обработка установки отменяется.

Далее обрабатываемый антецедент (терм) присваивается переменной x10. Компиляция начинается с рассмотрения вырожденного случая - антецедент имеет вид "несовп(A_1 A_2)". Такие антецеденты вводятся процедурой "развязка" при модификации теоремы и представляют собой указания на формальное различие термов A_1, A_2. Компилятор находит программные выражения t_1, t_2 для A_1, A_2 и вставляет оператор "не(равно(t_1 t_2))".

В общем случае предпринимается просмотр терма x10 для определения точки привязки, по которой будет найден идентифицирующий оператор. Переменной x11 присваивается текущее вхождение в x10, а x12 - символ по этому вхождению. Если антецедент имеет вид "не(равно(x t))", где x - переменная, то по теореме приема предпринимается попытка определить тип значений данной переменной, и x12 переприсваивается этот тип. Затем - откат к переходу через "ветвь 1".

Проверяется, что x12 не является квантором либо логической связкой. С помощью справочника "См" по символу x12 находится список идентифицирующих операторов, в шаблоне которых он выделен как точка привязки. Переменной x13 оказывается присвоен одноэлементный набор, состоящий из этого списка. Затем происходит просмотр списка x13, и переменной x15 присваивается текущий заголовок идентифицирующего оператора. Для него выполняется обращение к справочнику "усм", дающему формат x16 обращения к оператору. После контрольной точки "прием(123)" наступает момент учета текущего уровня x7 - проверяется, что он равен значению, указанному в информационном элементе "уровень(...)" из x16. Переменной x18 присваивается список всех отличных от x10 антецедентов (термов), упоминаемых в остающихся установках на идентификацию (усм ...). Этот список, вместе с x10, будет использоваться при реализации шаблона идентифицирующего оператора. Шаблон извлекается из x16, и переменной x24 присваивается список его конъюнктивных членов, в которых предпринято такое переобозначение переменных, чтобы они не встречались в x18 и x10. При этом x21 - список "старых" переменных шаблона; x22 - список его новых переменных.

Учитывается возможность, что x10 - квантор существования либо его отрицание. Тогда переменной x25 присваивается пара (указатель отрицания перед квантором - связывающая приставка квантора). В этом случае переменной x26 присваивается подкванторное утверждение; иначе x26 равно x10.

После перехода через "ветвь 1" происходит обращение к процедуре "реализация" (контрольная точка "прием(124)"). Здесь определяется такой набор термов x27, что при подстановке их вместо переменных x22 в утверждения шаблона x24 получается подмножество утверждений списка x18, пополненного утверждением x26. Требуется, чтобы первый элемент шаблона переводился в x26. Переменной x28 присваивается неиспользованный остаток утверждений списка x18. Если формат идентифицирующего оператора не имеет элемента "точкапривязки", то переменной x29 присваивается набор входных термов шаблона идентификации (эти термы представлены в "старых" переменных списка x21). Если элемент "точкапривязки" имеется, то проверяется, что антецедент был выделен указателем "теквхожд(...)" - в этом случае должен найтись информационный элемент (входвтерм v). Далее определяется набор x30 результатов переформулировки входных термов шаблона идентификации через переменные теоремы приема. Для этого реализуется подстановка термов x27 вместо переменных x21 и применяется стандартизирующая перестановка операндов коммутативных операций. Проверяется, что у каждого входного терма либо идентифицированы все его свободные переменные, либо его значение идентифицировано

с помощью элемента (выражение ...) "в целом", без обязательной идентификации внутренних переменных. Аналогичным образом определяется набор x32 результатов переформулировки выходных термов шаблона идентификации через переменные теоремы. После контрольной точки "прием(193)" проверяется, что ни один из выходных термов еще не идентифицирован. Переменной x33 присваивается программная переменная, значением которой служит текущая задача. Эта переменная представляет собой первое входное данное идентифицирующего оператора.

Создается накопитель x34 программных выражений для операндов обращения к идентифицирующему оператору x15. Сначала в него заносится x33. После контрольной точки "прием(195)" начинается цикл занесения в x34 выражений для оставшихся входных данных. Те термы набора x30, которые отличаются от переменных и не имеют информационного элемента (выражение ...), непосредственно определяющего их значение, обрабатываются процедурой "метаперевод". Для выходных термов обращения вводится список x36 новых программных переменных. Они регистрируются в x34, и создаются информационные элементы, связывающие их значения с выходными термами.

Если в формате идентифицирующего оператора был указан элемент "терм(D)", выделяющий подтерм D некоторого выходного терма, то предпринимается дополнительная идентификация D как подтерма выходного значения, путь к которому (последовательность номеров проходимых операндов) совпадает с путем к D в выходном терме. Если при этом значение D уже было идентифицировано ранее, то в накопитель операторов x35 заносится проверка совпадения старого и нового значений D.

По окончании обработки выходных термов - откат к переходу через "ветвь 1", где переменной x36 присваивается обращение к идентифицирующему оператору. Если x25 было отлично от 0, то на x36 навешивается внешний квантор существования, а при необходимости - также отрицание. Затем x36 регистрируется в компилируемой программе; добавляются операторы из накопителя x35, и удаляются все установки (усм ...), относящиеся к обработанным оператором x36 антецедентам.

19.10.17 Установка на идентификацию "свертка"

Установка на идентификацию "свертка" относится к приемам, имеющим заголовок "свертка". Теорема такого приема имеет вид $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\,\& A_m \to \forall_{y_1 \ldots y_k}(B_1 \,\&\ \ldots\ \&\ B_s \to B_0))$. Прием применяется в задачах на доказательство, причем утверждения $B_1, \ldots, B_s$ идентифицируются с некоторыми посылками данной задачи, а B_0 - с ее условием. При идентификации проверяется, что оставшиеся посылки задачи не имеют свободных вхождений переменных $y_1, \ldots, y_k$. Установка "свертка" инициирует выделение списка таких оставшихся посылок.

Обработка установки начинается с контрольной точки "прием(204)". Уровень x7 достаточно большой (равен 7), так как предварительно должны быть идентифицированы $y_1, \ldots, y_k$ и все переменные утверждений $B_1, \ldots, B_s$. Для этого же в начале обработки проверяется отсутствие установок (корень ...), (операнд ...). Далее переменной x10 присваивается список программных выражений для посылок, с которыми были идентифицированы антецеденты теоремы приема и антецеденты ее консеквента. Затем x10 заменяется на программное выражение для набора этих посылок. Переменной x13 присваивается связывающая приставка $y_1 \ldots y_k$. Проверяется, что все ее переменные уже идентифицированы, и переменной x14 присваивается набор программных выражений для переменных y_i. Наконец, вводится оператор, опреде-

ляющий список не использованных при идентификации посылок задачи, а также оператор, проверяющий, что переменные x14 не имеют свободных вхождений в эти посылки.

19.10.18 Установка на идентификацию "теквхожд"

Установка (теквхожд v) возникает в тех случаях, когда компилируется прием замены, точка привязки которого выбрана не в заменяемом подтерме, а в антецеденте. Тогда v - вхождение корня заменяемой части теоремы приема, и установка инициирует поиск в задаче вхождения, к которому будет применяться замена.

Обработка установки начинается после контрольной точки "прием(206)". Уровень x7 равен 4. Рассматриваются два различных подслучая.

Заменяемый подтерм представляет собой функциональную переменную

В этом подслучае переменной x10 присваивается вхождение v. По этому вхождению располагается функциональная переменная $f(x)$, где f уже идентифицировано с помощью информационного элемента (функция $f\ A_1\ A_2\ A_3$), а x - еще не идентифицированная переменная. f присваивается переменной x11. С помощью указанного информационного элемента находится программное выражение x14 для терма, определяющего значения функциональной переменной $f(z)$. Программное выражение для переменной z, используемой в данном функциональном шаблоне, присваивается переменной x15. Затем в компилируемую программу вставляется обращение к процедуре "смфунк", которая находит в задаче вхождения подтермов, представимых в виде $f(t)$. Она выдает в качестве выходных значений координаты данных вхождений, а также термы t. Регистрируется идентификация x с t. В информационных элементах (переходник ...) и (корень ...) сохраняются программные выражения для координат заменяемого подтерма.

Общий случай

Здесь сначала определяется индикатор x10, определяющий область поиска в задаче вхождения заменяемого терма. Если поиск ведется по посылкам и условиям, то x10 равно 0. Если же в фильтрах приема явно указано, где расположен заменяемый подтерм, то x10 равно логическому символу "посылка" либо "условие". На основе индикатора x10 создается оператор x11, перечисляющий координаты утверждений из соответствующей области задачи. Переменной x13 присваивается заголовок заменяемого подтерма. Добавляется оператор, проверяющий, что текущее утверждение не находится в посылке, с которой идентифицирован антецедент привязки. Добавляется оператор, перечисляющий все вхождения в текущее утверждение логического символа x13. Эти вхождения и рассматриваются как идентифицирующие для заменяемого терма. Для их обработки вводится новая установка на идентификацию (операнд ...). Координаты выбранного вхождения заменяемого терма регистрируются в информационных элементах (переходник ...) и (корень ...).

19.10.19 Установка на идентификацию "унификация"

Данная установка встречается только у приемов вывода теорем, имеющих заголовок "теорема". При получении новой теоремы иногда применяется процедура унификации, определяющая наименьшую подстановку, отождествляющую (унифициру-

ющую) заданные пары термов $(A_1(x), B_1(y)), \ldots, (A_n(x), B_n(y))$. Эти пары идентифицируются вместе со своими наборами переменных x, y из других частей теоремы приема. Обращение к процедуре унификации для них обозначается в теореме с помощью группы антецедентов вида "$A_1(r) = B_1(s), \ldots, A_n(r) = B_n(s)$". Такие антецеденты выделяются указателем приема "унификация". Процедура унификации определяет для наборов переменных x, y (обычно эти переменные относятся к различным теоремам и поэтому различны) наборы термов r, s, дающие унифицирующую подстановку. Результат идентификации набора r фиксируется в информационном элементе программного блока (унификация r X T). Здесь X - программное выражение для набора переменных x; T - программное выражение для набора термов r. Аналогичным образом регистрируется s. Если A_i и B_i относятся к одной и той же теореме, то y может совпадать с x, а r - с s.

Установка на идентификацию (унификация s) определяет наборы s вхождений антецедентов для унифицируемых пар термов. Обработка ее начинается с контрольной точки "прием(224)". Переменной x10 присваивается набор антецедентов s (в формате термов). Проверяется, что все эти антецеденты суть равенства. Далее заполняется ряд наборов: x12 - набор теоремных переменных, обозначающих переменные либо группы переменных, вместо которых осуществляется унифицирующая подстановка (выше см. x, y); x13 - набор переменных для подставляемых термов либо групп термов (выше - r, s); x14 - набор наборов вхождений в антецеденты s функциональных переменных, участвующих в унификации (выше - $A_i(r), B_i(s)$); x15 - набор наборов заменяющих эти вхождения термов, используемых при формировании пар унифицируемых термов (выше - $A_i(x), B_i(y)$). По окончании заполнения перечисленных накопителей - переход через "иначе 2". Здесь инициализируется пустым словом накопитель x16 программных выражений для установок на унификацию, передаваемых процедуре "унификация". Далее реализуется цикл синхронного просмотра наборов x10, x14 и x15 для заполнения x16. Там, где это необходимо, в программу вставляются операторы для проверки совпадения заголовков унифицируемых термов. По окончании цикла предпринимается регистрация в x16 ряда специальных дополнительных элементов. Наконец, в программу заносится обращение к процедуре "унификация" и предпринимается регистрация результатов идентификации переменных списка x13.

19.10.20 Установка на идентификацию "тип"

Установка (тип x A) означает, что теоремную переменную x нужно идентифицировать с логическим символом, являющимся типом объектов, определяемых теоремным выражением A. Обработка установки начинается с контрольной точки "прием(237)". Переменной x11 присваивается программное выражение для терма, идентифицированного с A. Затем в компилируемую программу заносятся операторы, определяющие с помощью справочника "тип" тип значения заголовка терма x11. Регистрируется результат идентификации переменной x с этим типом.

19.10.21 Установка на идентификацию "типзначения"

Эта установка аналогична предыдущей. Она имеет вид (типзначения v), где v - вхождение антецедента $\forall_x(A(x) \rightarrow P(t(x)))$, выделенного указателем "типзначения". Происходит идентификация переменной P с логическим символом - типом значения выражения $t(x)$, принимаемого им в предположении истинности антецедентов $A(x)$. Компиляция начинается с контрольной точки "прием(242)". Для определения типа

значения относительно заданного контекста применяется процедура "типзначения(...)".

19.10.22 Установка на идентификацию "чертеж"

Установка используется только при компиляции приемов анализатора "чертеж", позволяющего строить эскизы по условиям планиметрических задач. Так как она тесно связана с используемыми при посторении эскиза специальными структурами данных, рассмотрение ее будет отложено до описания анализатора "чертеж".

19.10.23 Установка на идентификацию "числоценка"

Установка (числоценка v) возникает при компиляции антецедентов "равно(x t)", выделенных указателем "числоценка". Эти антецеденты обеспечивают присвоение переменной x десятичной записи приближенного численного значения константного числового терма t. Приближенное значение получается при помощи оператора "числоценка". Данная возможность фактически в решателе сейчас не используется.

19.10.24 Установка на идентификацию "выч"

Установка (выч v) возникает при компиляции антецедентов, выделенных указателем "выч". Они реализуются путем вычислений в машинном формате "с плавающей запятой". Установки "выч" достаточно интенсивно используются при компиляции приемов анализатора "чертеж" и приемов справочника "опр", обслуживающего данный анализатор.

Обработка установки начинается с контрольной точки "прием(272)". Если заголовок приема отличен от "выч(...)", то переход через "иначе 2". В этом случае антецедент должен иметь вид присвоения "равно(x t)" переменной x значения выражения t. Все переменные t до этого должны быть идентифицированы с константными числовыми термами. Переменной x13 присвавается программное выражение для определяемого по t константного числового терма. В программу приема заносится обращение к процедуре "вычконст", определяющей приближенное значение терма с помощью арифметики чисел машинного формата "с плавающей запятой". Найденное значение переводится в формат десятичного числа ЛОСа. В этом же формате идентифицируется и x.

Если прием имеет заголовок "выч(...)" (случай справочника "опр"), то рассматривается более широкий список возможных типов антецедентов. Допускаются присвоения, проверочные равенства и отрицания равенств, а также проверка выполнения неравенств. При компиляции уже не используется обращение к процедуре "вычконст". Для получения программного выражения, находящего численное значение того или иного подтерма, применяется блок компилятора "прогрвыражение", создающий цепочку вычислений в терминах операторов "выч(...)". Для проверки условий равенства и неравенств используется оператор "Выч(...)". При присвоении значение программной переменной x регистрируется в формате данных "с плавающей запятой" с помощью информационного элемента (выч x ...).

19.10.25 Установка на идентификацию "вычисл"

Данная установка, хотя и имеет название, похожее на названия предыдущих установок, никак не связана с численными процедурами. Она возникает в тех ситуациях, когда при решении задачи на исследование требуется выразить какие-либо неизвестные выражения через известные параметры задачи, быть может, пополненные заданными неизвестными параметрами. Для такого выражения ("вычисления") используется вспомогательная задача на описание. Обращение к вспомогательному вычислению представлено в виде группы антецедентов "равно(x_1 t_1)", ..., "равно(x_n t_n)". Эти антецеденты порождают условия вспомогательной задачи, причем $x_1, \ldots, x_n$ играют роль неизвестных. Для выделения антецедентов применяется указатель приема "вычисл(N P)". Здесь N - список номеров антецедентов; P - дополнительная информация. Основные типы элементов набора P таковы: "посылки(A)" - указание списка A дополнительных посылок, используемых при вычислении; "удалениепосылок(B)" - перечисление списка B названий разделов, посылки которых не используются при вычислении; "известно(C)" - перечисление неизвестных, которые считаются при вычислении известными. После обработки антецедентов переменные $x_1, \ldots, x_n$ оказываются идентифицированы с результатами вспомогательного вычисления.

Установка имеет вид (вычисл V P), где V - набор вхождений выделенных антецедентов, P - то же, что в указателе приема. Обработка ее начинается с контрольной точки "прием(268)". Переменной x11 присваивается набор переменных $x_1, \ldots, x_n$; переменной x12 - набор программных выражений для термов $t_1, \ldots, t_n$. Переменной x13 присваивается программное выражение для списка посылок вспомогательной задачи, с учетом элемента "посылки(...)" набора P. Переменной x14 присваивается программное выражение для списка названий разделов, посылки которых не будут использоваться во вспомогательной задаче. Переменной x15 присваивается список программных выражений для переменных (обычно - неизвестных текущей задачи), которые будут рассматриваться во вспомогательной задаче как известные. В программу приема заносится обращение к оператору "вычисл", использующего x12-x15 для создания и решения вспомогательной задачи на описание. После этого остается лишь зарегистрировать результаты идентификации переменных списка x11.

19.10.26 Установка на идентификацию "рекурсия"

Если нужно сформировать некоторый набор термов $a(1) \ldots a(n)$ с помощью рекурсивного определения $a(1) = t, \forall_i (i \in \{1, \ldots, n-1\} \rightarrow a(i+1) = F(a(i)))$, то пара образующих определение утверждений используется в качестве антецедентов теоремы приема. Эта пара выделяется указателем приема "рекурсия(...)". Аналогичным образом можно использовать рекурсивное определение, определяющее группу наборов. При этом связанная переменная i и длина набора n должны быть одни и те же у всех кванторных импликаций. Установка на идентификацию (рекурсия A) ссылается на набор A вхождений антецедентов, образующих определение.

Обработка установки начинается с контрольной точки "прием(270)". Переменной x10 присваивается набор программных выражений для первых элементов определяемых наборов; переменной x11 - список теоремных переменных, обозначающих эти наборы. Далее создается набор x12 термов, полученных из термов $F(a(i))$ заменой всех встречающихся в них функциональных переменных $f(i)$ для f из x11 на обычные переменные f. Эти термы будут использоваться в качестве шаблонов при компиляции шага рекурсии. Одновременно определяются общая связанная перемен-

ная i всех кванторных импликаций, присваиваемая переменной x14, а также теоремный терм для длины набора, присваиваемый переменной x13. После перехода через "иначе 1" переменной x15 присваивается программное выражение для длины набора, представленной в формате десятичного числа. Заполняются накопители x16 и x17 программных переменных, обозначающих, соответственно, последние (текущие) элементы сформированной части наборов и сами наборы. Они инициализируются по известным первым элементам наборов, извлекаемым из x10.

Далее вводится вспомогательный программный блок x18 для компиляции шага рекурсии. В нем зарегистрирована идентификация переменных списка x11 с соответствующими текущими элементами наборов. Предпринимается коррекция информационных элементов (быстрпреобр ...) блока x18, учитывающая замену функциональных переменных $f(...)$ на обычные переменные. Относительно программного блока x18 определяется набор x19 программных выражений для новых элементов наборов. Переменной x20 присваивается список новых переменных блока x18, который будет играть роль связывающей приставки кванторной импликации, реализующей цикл итеративного заполнения наборов. Переменной x21 присваивается список антецедентов этой кванторной импликации. Сначала в нем идет оператор "Номера(...)", перечисляющий значения i номера текущего элемента набора, а затем расположены операторы программного блока x18. Переменной x22 присваивается набор конъюнктивных членов консеквента импликации. Здесь размещаются операторы изменения текущих элементов наборов и регистрации их в накопителях наборов. Наконец, кванторная импликация заносится в программу, и выполняется регистрация результатов идентификации переменных списка x11.

19.10.27 Компиляция обращения к техническому анализатору

При вычислениях на ГЕНОЛОГе могут использоваться специальные пакеты продукционного типа, называемые техническими анализаторами. Технический анализатор выполняет рассмотрение некоторой технической структуры данных и определяет для ее элементов значения заданного набора параметров. Эти значения регистрируются в двойнике анализируемой структуры данных, у которой вместо элементов "оригинала" размещаются наборы значений их параметров. Примерами вычислений такого рода могут служить разметка шахматной позиции указаниями числа ходов, необходимых заданной фигуре для достижения заданной позиции; разметка какой-либо вычислительной схемы указаниями глубины ее элементов, и т.п. Обращение к техническому анализатору f как оператору ЛОСа имеет вид $f(x_1\ \ldots\ x_n\ y)$, где $x_1, \ldots, x_n$ - входные значения, причем x_1 - анализируемая структура данных, а $x_2, \ldots, x_n$ уточняют тип выполняемой разметки (например, указывают исходное поле шахматной фигуры, которая будет перемещаться по доске). y - выходная переменная, которой присваивается структура данных с наборами значений параметров. Чтобы обратиться к тому же техническому анализатору из приема ГЕНОЛОГа, применяется несколько модифицированная запись $f(t_1\ \ldots\ t_n\ e\ r_1\ \ldots\ r_k)$. Здесь $t_1, \ldots, t_n$ - входные данные, т.е. t_1 представляет анализируемую структуру данных, а $t_2, \ldots, t_n$ - уточняют тип разметки. e - обозначение того элемента структуры данных t_1, параметры разметки которого нужно вычислить для дальнейшего использования их в теореме приема; $r_1, \ldots, r_k$ - переменные, соответствующие этим параметрам. Антецедент такого вида выделяется указателем приема "теханализатор". При обработке его предполагаются уже идентифицированными все переменные термов $t_1, \ldots, t_n, e$. "Выходные" переменные $r_1, \ldots r_k$ должны быть попарно различны и еще не иденти-

фицированы.

Для обработки антецедента, выделенного указателем "теханализатор", применяются две различные установки на идентификацию - (теханализатор v) и (сманализатор v R e). Первая из них ссылается на вхождение антецедента v и обеспечивает создание оператора, обращающегося к техническому анализатору для получения необходимой разметки анализируемого объекта. Вторая нужна для извлечения из найденной разметки данных, относящихся к конкретному элементу e. У нее R - программное выражение для разметки; e - то же, что в антецеденте. Вторая установка вводится компилятором после обработки первой.

Установка на идентификацию "теханализатор"

Обработка установки (теханализатор v) начинается после контрольной точки "прием(318)". Переменной x11 присваивается название анализатора f; переменной x12 - набор элементов, определяющих формат технического анализатора. В этом наборе находится элемент x13 вида "знач(x A набор($y_1 \ldots y_k$))", определяющий выделение в шаблоне A антецедента v переменной x для текущего элемента структуры данных и набора ($y_1 \ldots y_k$) его параметров, вводимых при разметке. Определяется подстановка (x16 - x17), переводящая шаблон A в рассматриваемый антецедент. С помощью этой подстановки и элементов "вход(x_i d_i)" формата, указывающих типы d_i значений входных данных x_i, составляется набор x18 пар (t_i, вычисл(d_i)), необходимый для компиляции входных данных t_i в требуемых форматах. Находится список x19 программных выражений для входных данных $t_1, \ldots, t_n$. После этого в компилируемую программу заносится оператор, обращающийся к процедуре "сманализатор(...)", которой передаются название анализатора и набор его входных данных. Эта процедура сначала проверяет наличие комментария (теханализатор f ...) к текущей задаче, сохранившего результат предыдущих обращений к процедуре f с теми же входными данными, и если он есть, извлекает разметку непосредственно из него. Иначе она обращается к техническому анализатору, сохраняет результат обращения в указанном комментарии, и затем выдает его в качестве значения своей выходной переменной. Результат R обращения к процедуре "сманализатор" регистрируется не в информационном элементе программного блока, а в новой установке на идентификацию (сманализатор v R e), которая вводится вместо установки (теханализатор v).

Установка на идентификацию "сманализатор"

Обработка установки на идентификацию "сманализатор" начинается после контрольной точки "прием(319)". Сначала здесь выполняются те же действия, что и в случае установки "теханализатор" - определяется формат оператора и находится подстановка (x17,x18), преобразующая шаблон антецедента A в антецедент v. Переменной x14 присваивается переменная шаблона A, обозначающая тот элемент анализируемой структуры данных, для которого вычисляется набор значений параметров; переменной x19 - соответствующий терм e из антецедента v. Переменной x21 присваивается программное выражение для e. Переменной x25 присваивается программное выражение для анализируемой структуры данных t_1. Далее используется справочник "элемент", позволяющий по программным выражениям для структуры данных, ее элемента и разметки структуры данных определить программное выражение x26 для набора параметров разметки, соответствующего элементу. В компилируемую программу заносится оператор, присваивающий новой программной переменной значе-

ние x26, а также оператор, проверяющий, что набор параметров x26 не содержит элемента "неопред" (не определенное при разметке значение). После этого, в соответствии с нумерацией переменных $r_1, \ldots, r_k$ в шаблоне "знач(...), предпринимается регистрация их идентификации с элементами набора x26.

19.10.28 Установка на идентификацию "пересечподст"

В приемах вывода теорем иногда возникает необходимость выделения таких подмножеств антецедентов двух кванторных импликаций, которые оказываются одинаковыми после применения к ним подстановки, унифицирующей для каких-то двух заданных термов. Пусть, например, имеются кванторные импликации $\forall_x(A(x)\ \&\ B(x) \rightarrow P(x))$ и $\forall_y(C(y)\ \&\ D(y) \rightarrow Q(y))$. При унификации $P(x)$ и $Q(y)$, определяемой антецедентом $P(r) = Q(s)$, возникает подстановка групп термов r, s вместо x, y. Если нужно выделить теперь подгруппы $A(x)$ и $C(y)$ всех антецедентов двух импликаций, совпадающих после применения к ним найденной унифицирующей подстановки, то добавляется еще один антецедент теоремы приема, имеющий вид $A(r) = C(s)$. Этот антецедент выделяется указателем приема "пересечподст".

Установка на идентификацию (пересечподст v) ссылается на вхождение v антецедента теоремы, выделенное указателем "пересечподст". Обработка этой установки начинается после контрольной точки "прием(11)". Переменным x17 и x18 присваиваются, соответственно, программное выражение для набора переменных, по которым действует унифицирующая подстановка, и для набора подставляемых термов. Находятся информационные элементы x19 и x22 вида (операнд ...), определяющие вхождения кванторных импликаций с антецедентами $A(x)$ и $C(y)$, и в программу заносится оператор "пересечподст(...)", который по наборам антецедентов данных кванторных импликаций, а также по паре x17, x18 определяет наборы антецедентов, идентифицируемые, соответственно, с $A(x)$ и $C(y)$.

19.10.29 Действия процедуры "идентификатор" по исчерпании установок на идентификацию

По завершении цикла просмотра установок на идентификацию (см. выше подраздел "Цикл просмотра установок на идентификацию") - переход через "иначе 1". Здесь проверяется, исчерпан ли список установок x2. Если не исчерпан, то текущий уровень x7, управляющий включением приемов компилятора, увеличивается на 1, и откат к началу идентификации. Если же уровень x7 достиг максимально допустимого значения 9, вместо его увеличения происходит отказ от компиляции приема.

Если список установок на идентификацию исчерпан (контрольная точка "прием(128)"), то предпринимаются перечисляемые ниже действия, завершающие создание идентифицирующей программы.

Анализ зависимостей переменных, расположенных под связывающими символами

Если в идентифицируемой части теоремы приема встречается квантор либо описатель A по переменным списка X, то нужно убедиться в том, что термы t, с которыми идентифицированы свободные в A переменные y, не содержат переменных, соответствующих переменным из X. Исключение составляет случай функциональной переменной $f(x_1 \ldots x_k)$. Если она идентифицирована с термом t, то выполняется

проверка независимости t лишь от тех переменных X, которые не упомянуты среди $x_1, \ldots, x_k$. Вставка компилятором операторов, обеспечивающих такие проверки, начинается с контрольной точки "прием(129)".

Прежде всего, проверяется, что не имеет места обработка процедурой "идентификатор" вспомогательного программного блока, введенного при рассмотрении программно реализуемого антецедента. Затем просматриваются такие вхождения x10 в идентифицируемую часть теоремы приема, внутри которых расположен квантор либо описатель. Для x10 проверяется отсутствие информационного элемента (обобщпост ...), отменяющего указанные выше проверки. Переменной x11 присваивается подтерм по вхождению x10. Затем начинается цикл просмотра свободных переменных x12 подтерма x11, играющих при анализе зависимостей роль указанной выше переменной y.

Для переменной x12 проверяется сначала отсутствие фильтра приема, определяющего, что она идентифицируется с десятичной записью числа. Такая запись вообще не имеет переменных, и сопоставление ее со связывающими приставками не требуется. Отсекаются другие случаи, в которых x12 идентифицируется с константным термом - если значением x12 служит объект, встречающийся в вычислениях, либо, согласно указателям "символ", "симметрично", значением служит логический символ. Наконец, отбрасывается случай вхождения переменной x12 внутри подтерма, выделенного указателем "функподст".

После этих проверок составляется список x13 всех тех теоремных переменных внешних по отношению к вхождениям x12 в x11 связывающих приставок, которые (точнее, их идентифицирующие переменные) не должны входить в терм, идентифицированный с x12. Отбрасываются те связывающие приставки, для которых указанная проверка не имеет смысла (например, вырождающиеся при идентификации из-за наличия указателя "развертка", и т.п.). Переменной x14 присваивается программное выражение для списка свободных переменных терма, идентифицированного с x12. Затем - переход через "ветвь 1".

Здесь просматриваются переменные x15 списка x13, с пропуском тех переменных, для которых либо уже гарантировано невхождение их в x14, либо такое вхождение допускается согласно указателю приема. Наконец, создается оператор x16, проверяющий невхождение идентифицирующей переменной для x15 в x14 (либо непересечение группы переменных, идентифицированных с x15, и списка x14). Этот оператор заносится в компилируемую программу с помощью процедуры "вставкафильтра".

Учет вхождений символа "примечание" в фильтры

Если фильтры приема имеют вхождение символа "примечание", то при их реализации может понадобиться накопитель комментариев, передаваемых преобразующей процедуре приема. После контрольной точки "прием(130)" происходит ввод такого накопителя, регистрируемого в информационном элементе (примечание ...).

Учет указателей "подствпосылки"

После контрольной точки "прием(156)" происходит учет указателя "подствпосылки", определяющего подстановку некоторого терма вместо заданной переменной во все утверждения комментария "коррекцияпосылок". Предполагается, что компилируется прием некоторого нормализатора. Определяются программные выражения x10 и x11 для указанных переменной и терма, после чего в программу заносится обращение

к оператору "подствпосылки".

Учет элементов "транслвыражения" для исключения избыточных фильтров

Если программный блок имеет информационный элемент (транслвыражения ...), определяющий, что при идентификации некоторой переменной x ее значением становится десятичное число, то в информационном элементе (комментарии ...) регистрируется указание на то, что обработка фильтра "десчисло(x)" избыточна.

После перечисленных действий применение процедуры "идентификатор" завершается.

19.10.30 Процедура "учетоперанда"

Для регистрации в программном блоке результата идентификации некоторого теоремного вхождения v с термом или вхождением, относящимся к текущему контексту вычислений, используется процедура "учетоперанда". Значительная часть приемов оператора "идентификатор" завершается обращением именно к этой процедуре. Она вставляет в компилируемую программу операторы, анализирующие заголовок идентифицирующего терма (например, проверяющие совпадение его с символом по вхождению v); вводит информационные элементы программного блока, сохраняющие различную информацию об идентификации вхождения v; определяет новые установки на идентификацию для вхождений, расположенных внутри v.

Обращение к процедуре имеет вид "учетоперанда(x1 x2 x3 x4 x5 x6 x7 x8). Здесь x1 - вхождение v в теорему приема; x2 - программное выражение либо программная переменная, значением которых служат либо идентифицирующий терм t для v, либо корневое вхождение этого терма в некоторый надтерм; x3 - программный блок. Если x2 - программное выражение для вхождения, то x4 равно 0, иначе x4 равно 1. x5 - описание приема; x6 - заголовок приема. Выходной переменной x7 присваивается список новых установок на идентификацию; переменной x8 - список тех подчиненных v вхождений, которые далее будут рассматриваться процедурой "идентификатор" как уже учтенные. В этот список само вхождение v не заносится.

Ветвь оглавления программ, посвященная процедуре "учетоперанда", расположена в том же списке, что и корневые ветви процедуры "идентификатор", т.е. в подразделе "Формирование идентифицирующей процедуры". К началу программы "учетоперанда" можно перейти также непосредственно из главного меню, набрав название этой программы.

Учет указателя "смравно"

Прежде всего, проверяется наличие информационного элемента (смравно A), в списке A которого содержится вхождение v. Это - достаточно редкая ситуация. В ней предполагается, что x2 определяет вхождение идентифицирующего терма t, причем продолжать идентификацию можно либо рассматривая само это вхождение, либо перейдя к корневому вхождению произвольного выражения s, для которого в контексте идентификации имеется равенство $t = s$. Перечисление таких альтернативных вхождений, "ответвляющихся" от вхождения x2, реализуется оператором "смравно" (случай приема сканирования задачи) либо оператором "Смравно" (прием пакетного оператора). Переменной x2 переприсваивается та программная переменная, которая

перечисляет указанные вхождения. Затем - откат к переходу через "ветвь 2" для продолжения обработки вхождения v.

Идентификация переменной с числовой константой

После контрольной точки "прием(9)" рассматривается подслучай, в котором на текущем теоремном вхождении v расположена переменная x, причем имеется фильтр "десчисло(x)" либо "целое(x)" либо "натуральное(x)". Проверяется, что этот фильтр не был учтен ранее, и в информационном элементе (комментарии ...) указывается, что далее он будет считаться учтенным. Переменной x14 присваивается программное выражение для корневого вхождения терма, идентифицирующего для v. Если ко вхождению v должен относиться внешний знак "минус", то переменной x15 присваивается программное выражение для вхождения, на котором может оказаться такой знак. Далее в компилируемую программу заносится оператор, проверяющий, что по вхождению x14 расположена десятичная запись числа. В случае фильтра "натуральное(x)" либо "целое(x)" вводится также оператор, проверяющий, что число имеет соответствующий тип. Наконец, создаются информационные элементы (транслвыражения x десчисло ...) и (вхождение x x14), регистрирующие идентификацию переменной x.

Идентификация переменной с указателем шахматной позиции

После контрольной точки "прием(18)" рассматривается подслучай, в котором на текущем теоремном вхождении v расположена переменная x, идентифицируемая с константной структурой данных, используемой при анализе шахматной позиции. Эта ситуация распознается по наличию внешнего "шахматного" отношения f, которому подчинено v. Если такое отношение указывает, что x идентифицируется с направляющим вектором шахматной линии либо с координатами шахматного поля, то переменной x13 присваивается идентифицирующий терм для v, и вводится информационный элемент (транслвыражения x ... числнабор(x13)), обеспечивающий переход от терма для пары координат (вектора или поля) к самой паре координат. В остальных случаях x идентифицируется с десятичным числом (например, с отдельной координатой шахматного поля).

Далее переменной x9 присваивается список всех вхождений, расположенных внутри текущего вхождения v и отличных от него.

Текущее вхождение выделено указателем "развертка"

Если текущее вхождение выделено указателем "развертка" (контрольная точка "прием(19)"), то сразу выдается результат, в котором указана новая установка на идентификацию (развертка ...). Все вхождения списка x9 здесь считаются учтенными, так как при обработке установки (развертка ...) будет использоваться рекурсивное обращение к процедуре "идентификатор".

Текущее вхождение выделено указателем "подтерм"

Данная ситуация распознается по наличию информационного элемента (стандтерм T), где T - терм, расположенный по текущему вхождению v. Здесь вместо идентификации T предпринимается определение программного выражения x11 для T и регистрация в программе оператора, сравнивающего идентифицирующий терм (см.

x2) со значением выражения x11. Все вхождения, расположенные внутри v, после этого считаются учтенными.

Учет указателя "знаксуммы"

После контрольной точки "прием(8)" проверяется наличие указателя "знаксуммы", выделяющего ту ассоциативно-коммутативную операцию, операндом которой служит текущее вхождение v. Для этого используется информационный элемент (знаксуммы A_1 A_2 A_3). Здесь A_1 - изменяемый знак; A_2 - вхождение ассоциативно-коммутативной операции; A_3 - либо 0, либо программное выражение для индикатора одновременной замены знака A_1 у всех операндов операции A_2. Если индикатор A_3 уже определен, то выражение x2 корректируется с учетом замены знака; при этом используется оператор "коррекциязнака" (x2 задает вхождение) либо операторное выражение "измзнака" (x2 задает терм). Если же индикатор еще не определен, то он определяется по текущему вхождению, в зависимости от наличия или отсутствия у v знака A_1. Одновременно корректируется x2.

Идентификация набора

После контрольной точки "прием(2)" рассматриваются случаи, когда заголовком терма T, расположенного по текущему вхождению v, является один из символов "префикс", "конкатенация", "суффикс". В этих случаях идентифицирующий терм может иметь вид "набор(...)", и при идентификации из него будут извлекаться те или иные поднаборы. Одновременно компилятор обеспечивает и альтернативный режим "обычной" идентификации для T.

Сначала переменной x11 присваивается указатель учета порядка элементов набора: если порядок несущественен, то x11 равно 1, иначе - 0. Далее рассматриваются следующие подслучаи:

1. T имеет вид "префикс(A B)". Заголовки подтермов A, B присваиваются переменным x12 и x13. Выделяются подслучаи:

   (a) x12 и x13 - символы "значение", т.е. A и B имеют вид $f(x)$, $g(y)$. Проверяется, что A, B - функциональные переменные. После этого вводится оператор, устанавливающий, что идентифицирующий терм имеет заголовок "набор". Формируется пара (вхождение ...), определяющая идентифицирующее вхождение для A, а также пара (терм ...), определяющая идентифицирующий терм для B. Эти пары регистрируются в соответствующих функциональным переменным f, g информационных элементах (функция ...).

   (b) x12 и x13 - символы переменных. Если значение переменной x12 не идентифицировано, то для выделения идентифицирующего ее элемента набора используется оператор "элементнабора" (x11 = 1) либо операторное выражение "элементномер" (x11 = 0). Идентифицирующий терм для x13 теперь можно получить с помощью операторного выражения "остатокнабора". Если x13 еще не было идентифицировано, то регистрируются значения для x12 и x13; если же x13 уже было идентифицировано, то вводится оператор, сравнивающий старый и новый идентифицирующий термы, а регистрируется только значение x12.

Если значение переменной x12 уже было идентифицировано, то вводится оператор, выделяющий элемент набора для сопоставления его переменной x12, а также оператор, сравнивающий этот элемент с ранее найденным значением x12. После этого x13 обрабатывается так же, как в предыдущем случае.

Заметим, что оператор "элементнабора" и операторное выражение "элементномер" могут использоваться для работы с наборами, заданными как в виде терма "набор(...)", так и с наборами, заданными с помощью операций "префикс", "суффикс", "конкатенация".

(c) A - неоднобуквенный терм, x13 - переменная. В этом случае вводится оператор, выделяющий элемент набора, сопоставляемый A, и предпринимается еще одно обращение к оператору "учетоперанда" - теперь уже для идентификации вхождения A с выбранным элементом. Предварительно обрабатывается x13 - если оно еще не было идентифицировано, то регистрируется результат его идентификации с остатком набора, а если уже было, то проверяется совпадение старого и нового идентифицирующих термов.

(d) A - логический символ, x13 - переменная. Вводится оператор, выделяющий элемент набора для сопоставления его A, и далее - оператор, проверяющий, что выделенным элементом служит логический символ A. Если x13 еще не идентифицировано, то регистрируется его идентификация с остатком набора, иначе - проверяется совпадение старого и нового идентифицирующих термов для x13.

2. T имеет вид "конкатенация($A_1 \dots A_n$)". Сначала здесь рассматривается подслучай, когда A_1 имеет вид "набор($B_1 \ \dots \ B_k$)", $n = 2$, причем A_2 - переменная x. Эта переменная присваивается переменной x13. Выполняется просмотр операндов B_i и заполнение накопителей x14, x15, x16. В накопитель x15 заносятся программные выражения для корневых вхождений идентифицирующих термов, сопоставляемых идентифицируемым термам B_i. В наборе x14 сохраняются новые установки на идентификацию, возникающие при рассмотрении операндов; значением переменной x16 после просмотра становится символьная константа, равная числу операндов k. Обработка очередного операнда происходит по схеме, аналогичной применявшейся в случае "префикс"а. Если B_i - еще не идентифицированная переменная, то с помощью оператора "элементнабора" либо операторного выражения "элементномер" для нее выбирается еще не использованный, согласно списку x15, элемент идентифицирующего набора. Если B_i - уже идентифицированная переменная, то в наборе находится элемент, равный ранее сопоставленному этой переменной. Наконец, если B_i - неоднобуквенный терм, то выбирается неиспользованный элемент набора (если x11 = 0, то - элемент с номером i), и предпринимается вторичное обращение к процедуре "учетоперанда" - для учета идентификации операнда B_i с этим элементом. Здесь может быть пополнен новыми установками список x14. Обработка их и отсечение неподходящих вариантов идентификации будут происходить уже после выхода из процедуры "учетоперанда". Однако, при вложенном обращении к процедуре проверяется соответствие заголовков B_i и выбранного для него элемента, что обычно сокращает перебор вариантов до приемлемого.

По окончании просмотра операндов B_i рассматривается переменная x. Если она

еще не идентифицирована, то идентифицируется с остатком элементов набора, иначе - проверяется совпадение этого остатка с ранее найденным значением x.

Если конкатенация не имеет указанного выше вида, то переход через "ветвь 2". Здесь сначала проверяется, что идентифицирующий терм имеет своим заголовком один из символов "набор", "префикс", "конкатенация". Если программное выражение x2 не является переменной, задающей корневое вхождение идентифицирующего терма, то вводится вспомогательная программная переменная для такого вхождения. Она переприсваивается переменной x2, а указатель вхождения x4 заменяется на 0. Далее используется операторное выражение "компонентынабора", дающее список возможно меньших фрагментов идентифицирующего набора, конкатенацией которых он является. С помощью этого выражения создается информационный элемент (набороперандов v ...), в котором найденные фрагменты перечисляются для последующей идентификации их с операндами конкатенации A_i. Чтобы выполнить эту идентификацию, процедура повторно вводит установку (операнд v ...).

3. T имеет вид "суффикс(A B)", где A, B - переменные. Случай совершенно аналогичен рассмотренному выше случаю "префикс(A B)". Отличие состоит в том, что вместо операторного выражения "элементномер", использованного ранее для нахождения i-го от начала набора элемента, применяется операторное выражение "Элементномер" для нахождения i -го от конца элемента.

Идентификация функциональной переменной

После контрольной точки "прием(3)" рассматривается случай, в которых терм T по текущему вхождению v имеет вид "значение(f t)". Здесь выделяются следующие подслучаи:

1. Переменная f выделена указателем приема "отображение" (контрольная точка "прием(15)"). Эта переменная присваивается программной переменной x10. Создается пара x11 вида (вхождение R) либо (терм R), определяющая программное выражение R для вхождения идентифицирующего значения функции f терма либо для самого идентифицирующего терма. Если имелся указатель "заменазнака", определяющий отнесение к f заданного внешнего знака, то предпринимается коррекция выражения R. Так как новая версия этого выражения задает терм, то одновременно заголовок пары x11 заменяется на символ "терм". Затем - переход через "ветвь 3". Начиная с этой точки, разбирается ряд подслучаев.

   (a) Имеется указатель "новаргумент" либо "функаргумент", выделяющий функциональную переменную f (контрольная точка "прием(11)"). Тогда для продолжения обработки текущего вхождения v вводится новая установка на идентификацию (новаргумент ...), и выход из процедуры "учетоперанда".

   (b) Функциональная переменная f выделена указателем "повторение" (контрольная точка "прием(13)"). В этом случае T имеет вид $f(y = z)$, где y, z - различные переменные. Идентификация должна заключаться в нахождении внутри идентифицирующего для T терма A такого подтерма s, который имеет не менее двух вхождений в A и содержит не менее двух

свободных переменных. После этого создается шаблон B функциональной идентификации для f, получаемый из A заменой всех вхождений s на новую переменную y. Сам подтерм s идентифицируется с переменной z. Для перечисления возможных s в компилируемую программу вставляется обращение к оператору "повтор". Создается информационный элемент (функция f ...), регистрирующий результаты идентификации переменной f.

(c) Уже создан информационный элемент (функция f X A B) (контрольная точка "прием(14)"). Если последний разряд B этого элемента равен 0, т.е. идентификация f еще не состоялась, то данный разряд заменяется на пару x11, и завершение процедуры. Иначе - переход через "иначе 3" к ветви, сравнивающей старую и новую версии идентификации f. Переменной x14 присваивается набор заголовков аргументов идентифицируемой функциональной переменной $f(t)$. Если X совпадает с t, т.е. ранее f идентифицировалось для того же набора переменных, то рассмотрение подслучая сводится к занесению в компилируемую программу оператора, проверяющего совпадение старого и нового выражений, определяющих значение f. В противном случае - переход через "иначе 1".

Сначала рассматривается один специальный подслучай. Переменной x16 присваивается список, образованный термами - аргументами функциональной переменной $f(t)$. Проверяется, что списки X и x16 имеют одинаковую длину. Переменной x17 присваивается список вхождений аргументов $f(t)$. Рассматриваемый подслучай состоит в том, что элементы списка X и заголовки термов x16 отличаются лишь в одной позиции - для X пусть это будет некоторая переменная x, а для x16 - терм u. Значением x18 становится одноэлементный список, состоящий из u, а значением x19 - одноэлементный список, состоящий из x. Проверяется отсутствие указателя приема, определяющего, что x должно рассматриваться не как единственная переменная, а как список переменных. Переменной x20 присваивается одноэлементый список, состоящий из вхождения аргумента u.

Если переменная x уже идентифицирована, то находится программное выражение x21 для нее. По B находится программное выражение для терма C, определяющего значение $f(X)$, и в компилируемую программу заносится оператор "усмподст", предпринимающий попытку представить идентифицирующий терм D для $f(t)$ в виде результата подстановки в C вместо x некоторого терма s. Для обработки сопоставления аргументу u терма s предпринимается повторное обращение к процедуре "учетоперанда".

Если же переменная x еще не идентифицирована, но зато u представляет собой уже идентифицированную переменную, то предпринимается попытка представить C как результат подстановки в D вместо u некоторого терма p. В информационном элементе (функция f ...) вместо X помещается u, шаблон B корректируется по терму D, и регистрируется идентификация x с p.

Если указанный выше подслучай различения списков X и x16 в единственной позиции не имеет места, то откат к переходу через "ветвь 1". Здесь проверяется, что все аргументы $f(t)$ суть переменные. Предпринимается сравнение списка x14 указанных аргументов и списка X. Переменным x14 и x15 переприсваиваются подсписки сравниваемых списков, образованные теми

разрядами, в которых они различаются. Если все переменные из x14 уже идентифицированы, то определяется набор x16 программных выражений для них. Если все переменные из x15 идентифицированы, то аналогичым образом определяется список x17 программных выражений, значения которых суть переменные, идентифицированные с переменными x15. Затем в компилируемую программу вставляются оператор, находящий идентифицирующий терм для $f(t)$ согласно информационному элементу (функция f ...), а также оператор, сравнивающий его с текущим идентифицирующим термом, определяемым программным выражением x2.

Рассматриваются также подслучаи, в которых x14, x15 одноэлементны, причем переменная одного из них не идентифицирована, а другого - идентифицирована со списком переменных. Для сравнения старой и новой версий идентифицирующего термов здесь используются операторы "усмподст" и "Реализация". Они определяют набор значений не идентифицированной переменной и сохраняют его в информационном элементе (результподст ...).

(d) Функциональная переменная f выделена указателем "вхождение" (контрольная точка "прием(12)"). В этом случае текущий идентифицируемый терм T имеет вид $f(x,t)$, где x - некоторая уже идентифицированная переменная либо список переменных. Дальнейшая идентификация должна происходить за счет выделения в текущем идентифицирующем терме произвольного вхождения w подтерма, который будет идентифицироваться с t. После этого новые термы вида $f(A,B)$ будут получаться путем замены вхождения w на B и подстановки терма (или термов) A вместо всех расположенных вне w вхождений x. В вырожденных случаях терм T может не содержать ссылки на x.

Переменной x13 присваивается вхождение аргумента функциональной переменной f. Если x пропущено, то заголовком этого аргумента не будет служить символ "набор", и тогда - переход через "иначе 2" к обработке вырожденного случая. Иначе находится список x15 теоремных переменных, входящих в подтерм x, и вводится информационный элемент (функция f x15 0 x11), регистрирующий идентификацию f. В компилируемую программу заносится оператор, перечисляющий вхождения w внутри идентифицирующего терма. Вводится элемент (вхождениетерма f ...), указывающий программное выражение для вхождения w, идентифицируемого с t. Наконец, для инициализации идентификации w с t предпринимается повторное обращение к оператору "учетоперанда". Вырожденный случай обрабатывается аналогично.

(e) Общий случай. Проверяется отсутствие информационного элемента вида (функция f ...). Находится список x всех переменных терма t, и вводится информационный элемент (f x 0 x11), где x11 - ранее созданная пара, определяющая выражение для значений функциональной переменной.

2. Идентифицируемый терм T имеет вид $f(t)$, где f ранее было идентифицировано с набором переменных X, а t определяет номер переменной этого списка (контрольная точка "прием(16)"; обычно вместо $f(t)$ в таких случаях используется обозначение $x(i)$). Находится программное выражение x11 для численного значения t, и с его помощью создается программное выражение x12 для

рассматриваемой переменной, извлекаемой из списка X. В компилируемую программу заносится оператор, проверяющий совпадение текущего идентифицирующего терма с этой переменной.

3. Имеется информационный элемент (элементыматрицы $f\ A\ n\ m$), согласно которому f уже идентифицировано с матрицей, причем A - программное выражение для набора наборов элементов матрицы (термов); n, m - программные выражения для размеров матрицы (контрольная точка "прием(17)"). В этом случае t имеет вид "набор($i\ j$)". Определяются программные выражения x11, x12 для i, j, строится программное выражение x13 для текущего элемента матрицы, и в компилируемую программу заносится оператор для сравнения этого элемента с текущим идентифицирующим термом.

4. В ряде случаев обработка функциональной переменной, выделенной указателем специального типа, предусмотрена в рамках основной части процедуры "идентификатор". В этих случаях процедура "учетоперанда" просто создает новую установку на идентификацию (операнд x1 x2), вводя при необходимости для x2 вспомогательную переменную, ссылающуюся на идентифицирующее вхождение, и завершает свою работу. К числу таких случаев относится выделение f указателями: "заменатермов", "Бинарнаяоперация", "смногочлен", "функподст". Обработка их процедурой "идентификатор" была описана выше.

Идентификация с применением справочника "вид"

После контрольной точки "прием(4)" предпринимается попытка использовать при идентификации текущего вхождения v справочник "вид". Этот справочник служит для указания специальных процедур идентификации выражений t, имеющих единственную переменную x. Например, в случае идентификации степенного выражения x^n с константным натуральным показателем степени n он дает ссылку на процедуру "выделениестепени". Все такие вспомогательные процедуры P имеют одинаковый формат обращения P(x1 x2 x3 x4 x5 x6). Здесь x1 - вхождение первого символа идентифицирующего терма; x2 - дополнительная информация (в приведенном примере - показатель выделяемой степени n); x3 - список утверждений контекста идентификации; x4 - список комментариев, накладывающих возможные ограничения на идентификацию. Значением выходной переменной x5 становится терм, идентифицированный с x; x6 - список посылок, использованных для обоснования корректности преобразований, применявшихся при идентификации.

Переменной x10 присваивается список переменных текущего идентифицируемого терма T; проверяется, что он одноэлементный, и переменной x11 присваивается его единственная переменная x. Находится результат x12 обращения к справочнику "вид" на логическом символе - заголовке терма T. Если он ненулевой, то представляет собой пару (название вспомогательной процедуры - программное выражение для передаваемой ей дополнительной информации). Проверяется отсутствие явного запрета на использование справочника "вид". Переменной x13 присваивается програмное выражение для списка посылок контекста идентификации; переменной x14 - программное выражение для списка комментариев, по умолчанию извлекаемых из текущих задачи или пакетного оператора. Если имеются указатели приема "выделениестепени(...)", дополняющие список комментариев обращения к вспомогательному оператору, то x14 корректируется с их учетом. Затем создается и регистрируется в компилируемой программе обращение x15 к вспомогательному оператору.

Если переменная x уже была идентифицирована, то добавляется оператор, проверяющий совпадение ее "старого" и "нового" значений. Иначе - вводится информационный элемент (терм x ...) для регистрации идентификации x.

Общий случай идентификации терма, отличного от переменной

Далее переменной x10 присваивается заголовок идентифицируемого терма. Сначала рассматривается случай, в котором x10 - логический символ. Здесь переменной x11 присваивается оператор, проверяющий наличие заголовка x10 у идентифицирующего терма.

После контрольной точки "прием(5)" проверяется наличие указателя "альтернатива", ссылающегося на текущее идентифицируемое вхождение. Такой указатель разрешает находиться на заданных позициях не тем символам, которые указаны в теореме, а некоторым альтернативным символам. По нему был создан используемый здесь информационный элемент (альтернатива A B), у которого A - набор вхождений в теорему приема, B - набор соответствующих этим вхождениям альтернативных заголовков подтермов. Требуется, чтобы альтернативные значения, указанные в B, использовались одновременно для всех вхождений A. Если какое-либо из вхождений w списка A уже идентифицировано (что распознается по информационному элементу (операнд w ...)), то x11 заменяется на оператор, анализирующий заголовок идентифицирующего терма в зависимости от того, каким был заголовок для w. Если там он был изменен, то и здесь берется альтернативная версия из B, иначе - берется текущий заголовок x10. В случае, когда вхождение из A идентифицируется впервые, x11 заменяется на дизъюнкцию, допускающую оба варианта заголовка. После коррекции x11 - откат к переходу через "ветвь 2".

Если прием имеет указатель "вариант", допускающий в качестве заголовка текущего идентифицирующего терма любой символ из заданного списка S, то x11 корректируется на соответствующую дизъюнкцию. Список S извлекается из ранее созданного информационного элемента (замены ...). Еще две коррекции x11, расширяющие список возможных заголовков идентифицирующего терма, связаны с указателями приема "частнпроизв" и "комплексное". В первом случае разрешается вместо символа "частнпроизв" рассматривать символ "производная" (обычная производная идентифицируется как частная производная первого порядка); во втором - одновременно рассматриввать комплекснозначные и веществennозначные аналоги операций.

После перечисленных коррекций оператора x11 попадаем на контрольную точку "прием(6)". Если идентифицируемый терм T неоднобуквенный, то x11 переприсваивается одноэлементный набор, состоящий из оператора x11. Если имеется информационный элемент (заголовок ...), указывающий, что заголовок идентифицирующего терма вообще несущественен, то x11 заменяется на пустой набор. Если программное выражение x2 отлично от переменной либо определяет идентифицирующий терм, а не идентифицирующее вхождение, то вводится вспомогательная переменная X для идентифицирующего вхождения, переприсваиваемая x2. При этом в накопитель x12 заносится оператор, присваивающий значение переменной X. Наконец, операторы накопителей x11 и x12 заносятся в компилируемую программу. Если идентифицируется набор T с несущественным порядком элементов, то в компилруемую программу заносится также оператор, проверяющий совпадение числа элементов идентифицирующего и идентифицируемого наборов. Для продолжения обработки идентифицируемого терма T создается новая установка на идентификацию (операнд ...).

Если идентифицируемый терм T - однобуквенный, то обработка его сводится к

занесению в программу оператора x11. Предварительно, для служебных слов "свой", "чужой", обозначащих в шахматах цвет своих либо чужих фигур, предпринимается коррекций x11 на соответствующий оператор "свой(...)" либо "чужой(...)".

Общий случай идентификации переменной

Этот случай обрабатывается после контрольной точки "прием(7)". Значением программной переменной x10 служит переменная x, расположенная на текущем идентифицируемом вхождении.

Если имеется информационный элемент (параметры ...), указывающий, что идентификация переменной x не нужна, то процедура завершается. В противном случае проверяется наличие информационного элемента, который уже определял бы ранее найденную идентификацию x. Если такой элемент имеется, то создается оператор, проверяющий совпадение старой и новой версий идентификации; иначе - предпринимается регистрация идентификации x согласно x2. В последнем случае, если имелась ссылка на x из указателя "заменазнака", выражение для идентифицирующего терма корректируется с учетом возможной замены знака.

19.11 Компиляция операторных выражений

Компиляция операторных выражений ГЕНОЛОГа выполняется процедурой "прогрвыражение". Обращение к ней имеет вид "прогрвыражение(x1 x2 x3 x4)". Оно представляет собой программное выражение ЛОСа. Здесь x1 - теорема приема; x2 - операторное выражение ГЕНОЛОГа, для которого требуется получить вычисляющее его значения операторное выражение ЛОСа R; x3 - набор указателей, уточняющих формат значений выражения R; x4 - программный блок, относительно которого строится R. Выражение R выдается в качестве значения операторного выражения "прогрвыражение(x1 x2 x3 x4)". Если компиляция не удалась, то в качестве значения выдается 0. Заметим, что для получения программных выражений, создающих новые термы по используемым в качестве шаблонов "теоремным" термам, применяется другая процедура - "метаперевод". Ее описание будет приведено в следующем разделе.

Перечислим используемые типы элементов списка x3 указателей формата значения:

1. "вхождение" - значением выражения R должно служить вхождение в терм либо набор;
2. "терм" - значением R должен служить терм или набор;
3. "переменная" или "логсимвол" - значением должен служить символ переменной либо, соответственно, логический символ;
4. "число" - значением должно служить десятичное число;
5. "операнд" - нет жесткого ограничения на выбор формата "вхождение" либо "терм", однако предпочтительнее формат "вхождение";
6. "блокпроверок" - предпочтительно не вводить при компиляции вспомогательные программные переменные (т.е. не изменять указанную в программном блоке x4 первую неиспользуемую переменную);

7. "набор" - значением должен служить набор;

8. "комментарий" - x2 представляет собой элемент комментария;

9. "выч" - значением должно служить число в формате "с плавающей запятой";

10. "вычисл(A)" - значением должен быть объект типа A, согласно классификации типов данных, используемых при вычислениях на ГЕНОЛОГе. Эта классификация может быть найдена в справочной информации для символа "тип";

Для перехода к началу процедуры "прогрвыражение" через оглавление программ выбирается подраздел ("Компилятор ГЕНОЛОГа", "Процедура ПРОГРВЫРАЖЕНИЕ").

Компиляция начинается с преобразования формата значения "вычисл(A)" в A, если A - один из символов "вхождение", "терм", "логсимвол". Это делается для того, чтобы в перечисленных случаях работала ветвь процедуры "прогрвыражение", ориентированная не на вычислительные, а на обычные логические форматы данных. Появление логических форматов данных под указателем "вычисл" возможно из-за того, что они иногда явно указываются в шаблонах справочника "вычисл", обеспечивающего компиляцию программно реализуемых антецедентов. В редких случаях, когда для получения значения x2 в "логическом" формате должен снова использоваться справочник "вычисл", указанная коррекция формата блокируется.

Далее - переход через "ветвь 2" к точке разветвления компиляции.

19.11.1 Вычислительные форматы данных

После контрольной точки "прием(7)" начинается ветвь процедуры, обрабатывающая форматы значений "вычисл(A)". Это - сравнительно редкий случай, относящийся к обработке программно реализуемых антецедентов теорем приемов. Прежде всего, терм "вычисл(A)" присваивается переменной x5, а наименование формата A - переменной x6. Далее компиляция разветвляется.

Случай переменной

После контрольной точки "прием(9)" рассматривается случай, когда x2 - однобуквенный терм, образованный переменной x7. Здесь разбираются следующие подслучаи:

1. Имеется информационный элемент (транслвыражения x7 F P). Тогда P - программное выражение, дающее значение x7 в формате F. Если этот формат совпадает с требуемым форматом x6, либо хотя бы один из форматов не определен (имеет вид логического символа "объект"), то происходит выдача результата P. В случае, когда F определен, а x6 - нет, происходит коррекция набора x3 - указатель "вычисл(объект)" в нем заменяется на "вычисл(F)". Это может позволить внешней процедуре уточнить формат, в котором представлено результирующее программное выражение. При рассогласовании числовых форматов F и x6 создается программное выражение, конвертирующее значения P требуемым образом.

2. x6 - шахматный формат "полефигуры" (координаты шахматного поля). Тогда предпринимается обращение к процедуре "прогрвыражение" для получения соответствующего x2 программного выражения x8 в формате "терм". Далее x8

переводится в требуемый формат с помощью операторного выражения "числнабор".

3. Остаточный случай, в котором предпринимается попытка получить значение x2 в логическом формате и от него перейти к числовому формату. Сначала программной переменной x8 присваивается логический формат "вхождение". Если обнаруживается, что для теоремной переменной x7 имелся указатель "заменазнака", то x8 заменяется на формат "терм". Далее с помощью повторного обращения к процедуре "прогрвыражение" находится программное выражение x9 для x2 в формате x8. Если использовался формат "терм", то x9 корректируется на формат "вхождение". Затем в компилируемую программу вставляется оператор, проверяющий, что по вхождению x9 находится десятичная запись числа. Этот оператор не вводится, если из информационного элемента (комментарии ...) видно, что в процессе идентификации переменной x7 расположение по вхождению x9 десятичной записи уже гарантировано. Наконец, в программу вводится оператор, присваивающий новой программной переменной "начало(x4)" численное значение выражения по вхождению x9. Это значение регистрируется в информационном элементе (транслвыражения ...), и выполняется его преобразование в требуемый формат x6.

Случай десятичной записи

Если x2 - десятичная запись числа, то переменной x7 присваивается данное число. Затем строится программное выражение, переводящее x7 в требуемый формат x6. В случае не определенного формата x6 происходит его доопределение как "вычисл(десчисло)".

Случай константного однобуквенного терма

Рассматриваются следующие подслучаи:

1. x2 - числовая константа "е" либо "пи". Тогда определяется численное значение данной константы с точностью до 14 десятичных знаков после запятой, которое переводится в формат x6.

2. x2 - единица измерения "м" либо "сек". Она преобразуется при вычислениях в числовую константу 1.

3. x2 - символ "текпозиция" при x6 = "шахматы". Создается программное выражение для шахматной позиции, определяемой текущей задачей.

4. x2 - символ "свой" либо "чужой", понимаемый в задаче на выбор хода в шахматах как указатель цвета своих либо чужих фигур. Создается соответствующее программное выражение, определяющее цвет фигур по текущей задаче.

5. В остальных случаях результатом служит сам логический символ x2.

Условное выражение

После контрольной точки "прием(27)" рассматривается случай, когда x2 имеет вид "вариант(B t_1 t_2)". Здесь B - терм теоремного уровня, который должен компилироваться с помощью установки на идентификацию (программа ...), т.е. так же, как

компилируются программно реализуемые антецеденты. Для обработки его создается вспомогательный программный блок x8. После обращения к процедуре "идентификатор" в блоке x8 формируется программа, проверяющая условие B. Она группируется в конъюнкцию операторов x12, причем по всем новым программным переменным на нее навешивается квантор существования. Далее находятся программные выражения x13, x14 для t_1, t_2, и выдается операторное выражение "вариант(x12 x13 x14)".

Конкатенация

После контрольной точки "прием(28)" рассматривается случай, когда x2 имеет вид "конкатенация($t_1 \dots t_n$)". Определяется список x8 программных выражений для поднаборов $t_1, \dots, t_n$, и по нему строится результирующее выражение.

Набор

После контрольной точки "прием(29)" рассматривается случай, когда x2 имеет вид "набор($t_1 \dots t_n$)". Если формат x6 не имеет вида "набор(t)", определяющего формат t элементов набора, то определяется список x8 программных выражений для элементов в неопределенном формате "вычисл(объект)". Иначе программные выражения для элементов определяются в формате t. В обоих случаях по x8 строится результирующее выражение "набор(...)".

Вектор

После контрольной точки "прием(61)" рассматривается случай, когда x2 имеет вид "Матрица(B n 1)", а x6 - вид "вектор(t)". Такой формат означает представление числового вектора в машинном формате t. Для представления используется пара (M N), где M - ссылка на числовой массив; N - число разрядов вектора в формате "целое со знаком". Фактически реализуется переход от задания вектора с помощью числового массива M к заданию его в виде пары (M N). Программное выражение M' для M берется из информационного элемента (транслвыражения B массив(t) M'); выражение для N находится по n с помощью обращения к процедуре "прогрвыражение".

Описатель "класс"

Случай, когда x2 имеет вид "класс($x_1 \dots x_n$ $B(x_1 \dots x_n)$)", разбивается на два подслучая:

1. Утверждение $B(x_1 \dots x_n)$ представляет собой конъюнкцию программно реализуемых условий, определяющих перечисление допустимых значений переменных $x_1, \dots, x_n$ без использования вспомогательных параметров. Этот случай рассматривается после контрольной точки "прием(31)". Переменной x8 присваивается набор конъюнктивных членов утверждения $B(x_1 \dots x_n)$. Вводится вспомогательный программный блок x9 для компиляции элементов списка x8. После применения к блоку x9 процедуры "идентификатор" из него извлекается конъюнкция x13 операторов, реализующих необходимое перечисление. Относительно x9 определяется программное выражение x17 для набора текущих значений переменных $x_1, \dots, x_n$. Если $n = 1$, то выражение x17 определяет значение единственной переменной, а не набора. Наконец, по x13 и x17 формируется результирующее выражение "выписка(...)".

2. Утверждение $B(x_1 \ \ldots x_n)$ задает условия на переменные $x_1, \ldots, x_n$ с помощью вспомогательных параметров. Содержащие эти вспомогательные параметры утверждения сгруппированы внутри $B(\ldots)$ под общим квантором существования. Фактически в компиляторе рассматривается только случай $n = 1$. При этом утверждение $B(x_1)$ имеет вид "и(существует($y_1 \ldots y_m$ и(равно(x_1 t) принадлежит(Y s) C))$D(x_1)$))", где $y_1, \ldots, y_m$ - вспомогательные параметры; "принадлежит(Y s)" - утверждение, инициирующее перечисление вспомогательных параметров путем просмотра некоторого известного списка s; C - утверждения, доопределяющие вспомогательные параметры и проверяющие их допустимость; $D(x_1)$ - дополнительные ограничения на регистрируемые в результирующем списке значения x_1.

   Рассмотрение данного случая начинается с контрольной точки "прием(45)". Программной переменной x8 присваивается теоремная переменная x_1. Вхождение квантора существования в консеквент теоремы приема присваивается переменной x10; список параметров $y_1, \ldots, y_m$ - переменной x11. Список конъюнктивных членов утверждения под квантором существования присваивается переменной x12. В нем находится равенство x13, выражающее переменную x_1 через вспомогательные параметры.

   Далее инициируются нулевые значения переменных x16, x17, x18. Присвоение фактических значений этим переменным происходит после контрольной точки "прием(46)". Сначала в списке x12 находится утверждение x19 вида "принадлежит(Y s)". Компиляция происходит лишь в тех случаях, когда Y - либо вспомогательный параметр y_i, либо имеет вид $y_i(R)$, где все переменные внутри R уже идентифицированы. Последний случай обычно означает, что вспомогательный параметр y_i представляет собой некоторую рекурсивно определяемую последовательность, а $y_i(R)$ - ее первый член. Тогда вводится новая переменная y', которая будет идентифицироваться в процессе просмотра списка s; для учета ее связи с y_i в накопителе x21 сохраняется равенство "$y_i(R) = y'$. В обоих случаях программной переменной x16 присваивается переменная, по которой ведется просмотр списка s (y_i либо введенная во втором случае переменная y'), а переменной x17 - терм s. Будем далее обозначать варьируемую переменную x16 через y. x18 становится равно остатку списка x12 после исключения из него утверждений x13 и x19, пополненному равенством для y' (если оно есть). Далее - откат к переходу через "ветвь 3".

   После контрольной точки "прием(47)" проверяется, что x16 было определено, т.е. утверждение x19 имело указанный выше допустимый вид. Затем переменной x19 присваивается переменная компилируемой программы, которая играет роль накопителя результата. Вводится оператор, инициализирующий x19 пустым словом. Здесь компиляция разветвляется.

   (а) После контрольной точки "прием(48)" рассматривается случай, в котором s имеет вид "номера(N_1 N_2)", т.е. перечисление ведется по заданному отрезку целых чисел. Переменным x21 и x22 присваиваются программные выражения для $N_1 - 1$ и N_2 соответственно. В компилируемую программу заносятся операторы, присваивающие эти значения двум новым переменным. Первая из них используется как указатель текущего значения из s ; вторая - как указатель максимального значения из s. Заметим, что первая переменная (она обозначается далее как x23) уменьшена на единицу; это

происходит из-за того, что увеличение ее значения в цикле будет выполняться до ее использования.

Чтобы упростить вычисления, связанные с шагом цикла, внутри равенства x13 и утверждений списка x18 находятся все подвыражения вида $a+by$, где a, b не содержат y. Для каждого такого подвыражения вводится своя новая переменная x, которая будет инициироваться и изменяться в цикле независимо от других. Все вхождения подвыражений $a + by$ в x13 и x18 заменяются на такие переменные x. Переменной x25 сначала присваивается список троек $(x\ a\ b)$. После контрольной точки "прием(49)" вторые и третьи компоненты троек списка x25 заменяются на программные выражения. Первое из них инициирует x, второе - дает шаг изменения x. Как и в случае основной варьируемой переменной y, инициализация происходит с учетом того, что увеличение значения x в цикле выполняется до его использования. Заметим, что пока данная ветвь компилятора ориентирована только на вычисления в формате "с плавающей запятой" (своим возникновением она обязана рассмотрению задач на программирование процедур численного решения дифференциальных уравнений).

После контрольной точки "прием(50)" вводится список x26 программных переменных, хранящих текущие значения параметров списка x25. В компилируемую программу заносятся операторы, инициирующие эти переменные.

После контрольной точки "прием(51)" вводится оператор "повторение", представляющий собой точку отката к началу цикла. Сразу после него размещаются операторы, увеличивающие значения переменной y и параметров списка x25. Затем размещается оператор, проверяющий, что y не превосходит максимально допустимого значения. После него вставляется вспомогательный терм "иначе(фикс)", который впоследствии будет преобразован в оператор перехода.

После контрольной точки "прием(52)" вводится вспомогательный программный блок x27 для компиляции шага цикла. Затем происходит обращение к процедуре "идентификатор", выполняющей эту компиляцию. Переменной x31 присваивается программное выражение для вычисления значения выражения t - правой части равенства x13. Ветвь для обработки дополительного условия $D(x_1)$ на найденное текущее значение x_1 в компиляторе лишь обозначена - на рассматривавшихся примерах надобности в ней не возникало. После контрольной точки "прием(53)" вводится оператор, регистрирующий текущее значение x_1 в накопителе x19, а завершается программа оператором "продолжение", обеспечивающим продолжение перечисления. Наконец, происходит подключение программы вспомогательного блока x27 к программе основного блока x4. Здесь же введенный ранее вспомогательный терм "иначе(фикс)", соответствующий переходу по завершении цикла, преобразуется в оператор "иначе(...)", ссылающийся на добавленный в конце программы блока x4 новый пустой фрагмент. Далее происходит выдача результата - программного выражения x19 для накопителя значений x_1.

(b) После контрольной точки "прием(55)" рассматривается случай, когда s - переменная, имеющая своим значением ранее идентифицированное конечное множество. Это множество определяется программным выражением P

из информационного элемента (транслвыражения s h P). Указатель типа значения h имеет вид "класс(p)", т.е. множество представлено как набор своих элементов. Данный случай компилируется значительно проще, чем предыдущий. Перечисление здесь реализуется не по схеме "инициализация значений параметров и последующие итеративные их изменения", а с помощью перечисляющего оператора ЛОСа "входит". Для реализации утверждений "остаточного" списка x18 создается вспомогательный программный блок x24. После контрольной точки "прием(57)" выполняется обращение к процедуре "идентификатор", создающей программу обработки этих утверждений. Переменной x29 присваивается программное выражение для вычисления правой части t равенства x13. Для учета дополнительных ограничений $D(x_1)$ (если они есть, т.е. x9 не равно x10) снова предпринимается обращение к процедуре "идентификатор", дополняющей программу блока x24 проверяющими данные ограничения операторами. В конце программы блока x24 вводится оператор, пополняющий накопитель x19 текущим значением x_1, а также оператор "продолжение". Дальнейшие действия аналогичны случаю перечисления по отрезку целых чисел.

Конечная таблица

Если x2 имеет вид "отображение(x $\exists_i(i \in \{0,\ldots,n\}$ & $x = a + bi)$ $f(x)$)", а тип данных x6 - вид "набор(r)", то результат компиляции должен представлять собой программное выражение для массива значений $f(a+bi)$ для $i = 0, 1, \ldots, n$, представленных в машинном числовом формате r. Обработка данного случая начинается с контрольной точки "прием(34)". Переменной x7 присваивается формат r; переменной x8 - переменная x; переменной x10 - переменная i. Теоремные выражения a, b присваиваются, соответственно, переменным x16 и x17. Затем создаются программные выражение x18 и x19 для начала и конца промежутка изменения индекса i, а также программные выражения x20 и x21 для значений a, b. В компилируемую программу заносится оператор, определяющий длину результирующего массива, а также оператор, выделяющий память для этого массива. Инициализируются изменяемые в цикле значения x, i; величина x21 шага изменения x присваивается отдельной программной переменной. Затем размещается оператор "повторение", представляющий собой начало цикла заполнения массива. Для компиляции операторов, обеспечивающих вычисление значения $f(x)$, создается вспомогательный программный блок x22. Программное выражение, вычисляющее значение $f(x)$, присваивается переменной x23. Операторы из блока x22 переносятся в текущий блок x4, после чего помещается группа операторов, завершающих шаг цикла: запись текущего значения в массив, проверка окончания и коррекция значений x, i. По окончании цикла управление передается на новый пустой фрагмент программы, размещаемый в конце списка фрагментов блока x4.

Конечная сумма

Если x2 содержит конечную сумму $\sum_{i=m}^{n} a(i)$, то создается программа для вычисления этой суммы в формате "с плавающей запятой" либо "целое со знаком". Указанная сумма в логической записи представляет собой терм "суммавсех(отображение(i и(целое(i)меньшеилиравно(m i)меньшеилиравно(i n))$a(i)$))".

Обработка данного случая начинается после контрольной точки "прием(39)".

Внутри x2 усматривается вхождение x7 конечной суммы указанного вида. Проверяется, что ни одна из свободных переменных этой суммы не связана внешним квантором или описателем. В противном случае ее обработка откладывается до устранения внешних связывающих символов. Выражение x2 будет компилироваться в два этапа: сначала определится значение выделенной конечной суммы, затем в x2 вместо этой суммы будет подставлена новая переменная с известным значением, и к результату подстановки снова будет применена процедура "прогрвыражение", которая и даст окончательный результат.

Переменной x8 присваивается вхождение символа "отображение"; x10 - индекс суммирования i; x16, x17 - выражения m, n; x18 - суммируемое выражение $a(i)$. После контрольной точки "прием(40)" предпринимается выделение в x18 подвыражений вида $a + bi$, заменяемых на вспомогательные переменные, изменяемые в цикле суммирования каждая со своим шагом b. Одновременно с преобразованием терма x18 составляется список x19 пар $((a, b), x)$, где x - вспомогательная переменная, соответствующая значениям a, b. Переменным x20 и x21 присваиваются программные выражения для границ суммирования m, n, представленные в машинном формате "целое со знаком".

Далее вводится список x22 форматов, в которых будут представляться значения вспомогательных переменных x. По умолчанию выбирается формат "с плавающей запятой", но в тех случаях, когда видно, что переменная используется как индекс массива, этот формат заменяется на "целое со знаком". После контрольной точки "прием(41)" создается набор x23 программных выражений для инициализации вспомогательных переменных, а после контрольной точки "прием(42)" - набор программных выражений для шага изменения вспомогательных переменных. Затем переменной x25 присваивается программная переменная X, значением которой будет служить индекс i, и вводится список x26 программных переменных для текущих значений вспомогательных переменных. Накопителем суммы будет служить переменная, номер которой на единицу больше номера переменной X.

В компилируемую программу заносятся операторы инициализации цикла: присвоение исходного значения переменной X, присвоение нуля накопителю суммы, присвоение начальных значений переменным списка x26, оператор "ветвь" для продолжения вычислений по завершении цикла суммирования, и оператор "повторение" для откатов к очередному шагу цикла. В текущий программный блок x4 заносятся информационные элементы, задающие идентификацию определенной после инициализации переменной i, а также вспомогательных переменных. Теперь все готово для определения программного выражения, вычисляющего x18. Такое выражение x29 и определяется путем повторного обращения к процедуре "прогрвыражение". Далее в компилируемую программу вставляются операторы завершения шага цикла: контроль достижения индексом i максимального значения; коррекция накопителя суммы; коррекции параметров цикла. Переход по завершении цикла организуется к фрагменту с оператором "обрыв", переводящим далее к расположенному перед началом цикла оператору "ветвь". В свою очередь, этот оператор приводит к новому пустому фрагменту в конце накопителя фрагментов текущего программного блока x4.

Однако, после вычисления суммы компиляция выражения x2 еще не завершена. Для ее продолжения выбирается новая переменная x33; определяется результат x34 замены в терме x2 вычисленной суммы на эту переменную; вводится информационный элемент (транслвыражения x33 ...), регистрирующий найденное значение суммы в программном блоке, и предпринимается завершающее компиляцию обращение

к процедуре "прогрвыражение" для x34.

Использование информационного элемента "выражение"

Информационный элемент (выражение t r P) сохраняет ранее найденное каким-либо способом программное выражение P для вычисления значений выражения t в формате r. Если такой элемент имеется, причем t входит в компилируемый терм x2, то можно упростить компиляцию, воспользовавшись программным выражением P. Данный случай рассматривается начиная с контрольной точки "прием(58)". Выбирается новая переменная x9; в программном блоке регистрируется информационный элемент (транслвыражения x9 r P), и находится результат x12 замены всех вхождений в x2 выражения t на x9. Затем компилируется x12.

Матрица либо вектор, заданные описателем "отображение"

Если выражение x2 имеет вид "отображение(i j и($i \in \{1,\ldots,m\}$ $j \in \{1,\ldots,m\}$) $a(i,j)$)" либо вид "отображение(i $i \in \{1,\ldots,m\}$ $a(i)$)", т.е. определяет, соответственно, матрицу либо вектор, причем элементы этих матрицы или вектора идентифицируются с константными числовыми термами, то для перехода к представлению x2 в вычислительном формате используются процедуры "конвертматр" либо "конвертвект". Компиляция со вводом обращений к данным процедурам начинается: для матриц - с контрольной точки "прием(62)"; для векторов - с контрольной точки "прием(63)". Программное выражение для матрицы либо вектора с константными термами берется из информационного элемента (элементыматрицы ...) или (вектор ...).

Использование набора псевдокоманд

Для вычисления значения числовой функции, заданной формулой в базисе элементарных функций, можно использовать оператор "Выч(программа N r m)", реализуемый за одно обращение к интерпретатору ЛОСа. У этого оператора N - набор псевдокоманд, кодирующих последовательность применений элементарных операций к вход-выходным данным r формата "с плавающей запятой" и вход-выходным данным m формата "целое со знаком". Если к моменту компиляции выражения x2 уже имеется информационный элемент (усмфункция f ...), хранящий программное выражение для набора псевдокоманд N, вычисляющих некоторую функцию f, причем в x2 встречается подтермы $f(a)$, то компилятор использует N для вычисления $f(a)$. Этот случай рассматривается начиная с контрольной точки "прием(44)". После регистрации в программе операторов, вычисляющих $f(a)$, выбирается новая переменная x16. В программный блок заносится информационный элемент (транслвыражения x16 ...), указывающий для x16 вычисленное значение $f(a)$. Затем находится результат x18 замены всех вхождений $f(a)$ в x2 на переменную x16, и выполняется компиляция x18.

Получение элемента матрицы либо массива

Остаточные случаи компиляции выражения x2 вида "значение(f n)" рассматриваются после контрольной точки "прием(32)". Сначала рассматривается случай, когда переменная f идентифицирована с помощью информационного элемента (элементыматрицы f ...), определяющего программное выражение для набора наборов

термов - элементов матрицы. Здесь находятся программные выражения x10, x11 для номеров строки и столбца элемента $f(n)$, и с их помощью создается программное выражение, извлекающее нужное значение из набора наборов.

Затем рассматривается случай, когда f идентифицирована с помощью информационного элемента (транслвыражения f ...), причем значение ее вычислено в одном из форматов "матрица(t)", "вектор(t)" или "массив(t)". Во всех этих случаях значение $f(n)$ берется из известного массива, причем терм n определяет номер элемента массива. Находится программное выражение x10 для n, которое затем корректируется - либо с помощью информационного элемента (сдвигнабора ...), указывающего величину смещения n по отношению к начинающейся с нуля нумерации элементов массива, либо, в случае элемента (вектор ...), из x10 просто вычитается единица. При необходимости уточняется входной формат для x2 (полагается равным t), и выдается программное выражение, извлекающее из массива нужный элемент.

Создание набора псевдокоманд

Если x2 представляет собой выражение, построенное из числовых констант и переменных только с помощью элементарных числовых функций, причем нужно вычислить значение в формате "с плавающей запятой", то в компилируемую программу вставляется обращение к оператору "Выч(программа ...)". Этот оператор использует набор псевдокоманд, определяющих последовательность действий при вычислении значения, и дает некоторое ускорения вычислений на ЛОСе по сравнению с цепочкой обращений к операторным выражениям "выч", выполняющим те же действия. Рассмотрение данного случая начинается после контрольной точки "прием(33)". Сначала проверяется, не имеет ли x2 вида $f(a_1 \dots a_n)$ где $n \leq 2$, а каждое a_i - либо переменная, либо константа. В этом подслучае вместо набора псевдокоманд используется одноразовое обращение к операторному выражению "выч". Иначе (переход через "ветвь 2") происходят обращение к процедуре "вычпрог", находящей программное выражение для указанного набора псевдокоманд, и вставка в программу оператора "Выч(программа ...)". В тех случаях, когда рассматривается константное выражение, применяется процедура "вычконст", находящая его приближенное значение непосредственно при компиляции.

Вычисление значения элементарной числовой функции

После контрольной точки "прием(59)" рассматривается случай, в котором для вычисления значения элементарной числовой функции в формате "с плавающей запятой" применяется обращение к операторному выражению "выч". После контрольной точки "прием(60)" аналогичные действия (но для существенно меньшего списка операций) предпринимаются при вычислении в формате "целое со знаком".

Использование справочника "вычисл"

Наконец, после контрольной точки "прием(8)" реализуется попытка использовать для компиляции x2 справочник "вычисл". Этот справочник уже встречался выше в описании процедуры "идентификатор", где он применялся для обработки программно реализуемых утверждений. Если для обработки утверждений специального вида в заданных форматах вход-выходных данных была создана своя программа, то справочник "вычисл" дает ссылку на нее. Аналогичным образом этот справочник применяется и для реализации выражений. Тогда в теореме приема "вычисл" помещается

указатель "выход(t)", означающий, что реализуется выражение, и t - формат данных результата. Так как схема компиляции здесь похожа на схему, уже приведенную в описании процедуры "идентификатор", то мы ее опускаем.

19.11.2 Особые случаи компиляции

Перед основной частью процедуры "прогрвыражение", ориентированной на получение результата в одном из логических форматов (терм, вхождение в терм, символ), располагается ветвь, обрабатывающая ряд особых случаев, в основном, относящихся к нелогическим типам данных. Некоторые из используемых здесь указателей типов представляют собой устаревшие версии и дублируют указатели "вычисл(...)".

Указатель "выч"

Этот указатель предназначен для получения результата в машинном формате "с плавающей запятой". Он дублирует указатель "вычисл(число)" и был использован при компиляции вычислений, сопровождающих построение эскиза чертежа в геометрических задачах. Обработка его начинается после контрольной точки "прием(2)". В отличие от ветви, обрабатывающей указатель "вычисл(число)", здесь не используется оператор "Выч(программа ...)", работающий с набором псевдокоманд, а значения элементарных функций определяются с помощью операторного выражения "выч".

Применение справочника "оператор"

Если заголовок компилируемого выражения представляет собой заголовок программы ЛОСа для вычисления значения данного выражения, то компиляция выполняется с помощью справочника "оператор" (контрольная точка "прием(6)"). Этот справочник определяет по заголовку выражения f набор типов данных входных переменных его программы, а также тип значения результата (он помещается в конце набора). Переменной x8 присваивается список программных выражений для входных переменных, вычисленных в требуемых форматах, и создается операторное выражение x10 для обращения к программе f. Далее x10 корректируется в зависимости от требуемого формата результата вычислений.

Получение набора объектов

После контрольной точки "прием(3)" рассматривается компиляция выражения x2, значение которого должно представлять собой набор. Если x2 - однобуквенный терм, состоящий из переменной x5, то находится такой информационный элемент текущего программного блока, который позволяет идентифицировать x5 в виде набора. Здесь рассматриваются информационные элементы (наборпеременных ...), (наборчленов ...), (вектор ...). В первом случае выдается выражение для набора переменных, в остальных - для набора термов.

Если терм x2 выделен указателем "развертка", то компиляция происходит аналогично тому, как это делалось ранее. После контрольной точки "прием(4)" определяются: ассоциативно-коммутативная операция x7, которая будет соединять порождаемые при развертке термы; шаблон общего члена x8, а также пара теоремных переменных (варьируемая при развертке переменная i - верхний предел ее значений n). В качестве нижнего предела берется 1. Если все вхождения индекса i в шаблон x8

располагаются только внутри подтермов вида $f(i)$ для одной и той же функциональной переменной f, причем n равно длине набора x13, с которым идентифицирована f, то развертка получается путем последовательной подстановки вместо $f(i)$ в шаблон x8 элементов набора x13 (контрольная точка "прием(5)"). Иначе перечисление значений индекса i при развертке обеспечивается оператором "Номера".

Во всех прочих случаях для компиляции выражения x2 в формате "набор" применяется справочник "прогрвыражение". Этот справочник объединяет множество специальных случаев компиляции, соответствующих конкретным заголовкам выражения x2. Кроме данной ветви, он используется также при обработке рассматриваемого ниже общего случая компиляции.

19.11.3 Общий случай компиляции

После контрольной точки "прием(10)" размещается основная ветвь программы, предназначенная, главным образом, для получения логических типов данных.

Переменная

Прежде всего, рассматривается случай, когда x2 - однобуквенный терм, состоящий из переменной x5. Началом этой ветви служит контрольная точка "прием(11)". Если имеется указатель типа значения "число", причем x5 идентифицируется по информационному элементу (транслвыражения x5 ... t), то выдается ответ - преобразованное к формату десятичного числа ЛОСа значение t. Иначе находится список x6 всех информационных элементов E, идентифицирующих переменную x5.

Если имеется указатель типа результата "вхождение" или "операнд", то выполняется преобразование программного выражения для x5, извлекаемого из x6, к формату "вхождение". В противном случае - проверяется наличие указателя "переменная" или "логсимвол". Здесь определяемое по x6 программное выражение преобразуется к формату отдельного символа. В остальных случаях результат преобразуется к формату терма.

Наконец, при отсутствии идентифицирующего x5 информационного элемента проверяется наличие указателя приема, определяющего выбор для x5 новой переменной. Если такой указатель есть, то в компилируемую программу вводится оператор выбора новой для контекста срабатывания приема переменной. Регистрируется идентификация переменной x5, и выдается программное выражение для новой переменной либо однобуквенного терма - в зависимости от указанных в наборе x3 требований.

Служебный символ

При компиляции служебного символа x2 рассматриваются следующие подслучаи:

1. Символ "теквхожд". Этот символ обозначает подтерм, идентифицируемый с тем термом теоремы, который содержит точку привязки. Он определяется через информационный элемент программного блока (корень ...) либо (теквхожд ...). Первый используется для приема сканирования задачи, второй - для пакетного оператора.

2. Символ "корень". Этот символ обозначает терм задачи, содержащий точку привязки, либо преобразуемый терм пакетного оператора. В случае приема сканирования задачи определяется через переменную сканирования x3, ссылающуюся на вхождение текущего терма.

3. Символ "внешкорень". Ссылка на терм, обрабатываемый во внешнем стэковом кадре. Извлекается из информационного элемента (внешкорень ...).

4. Символы "свой", "чужой" для цвета шахматных фигур. Определяются через текущую задачу x1.

5. Символ "результат". Ссылка на ранее найденный заменяющий терм. Определяется через информационный элемент (результат ...).

6. Символ "фикс". x2 имеет вид "фикс(...)" и представляет собой указатель вхождения в теорему некоторого подтерма t. Определяется программное выражение для вхождения идентифицированного с t подтерма A либо самого подтерма A (в зависимости от указанного в x3 формата).

7. Символ "посылки". Обозначает список утверждений, относящихся к контексту применения приема. Программное выражение для него определяется процедурой "посылки".

8. Символ "вход". Обозначает терм, передаваемый нормализатору для обработки. Используется при компиляции обращения к этому нормализатору. Определяется по информационному элементу (нормализатор ...).

9. Символ "значение". x2 имеет вид "значение(f t)". Если уже имеется информационный элемент (функция f t ...), то программное выражение для терма (либо вхождения), идентифицированного с $f(t)$, берется непосредственно из этого элемента. Иначе это программное выражение получается при помощи описываемой ниже процедуры "метаперевод".

10. Символ "модификатор". Обозначает остаток неидентифицированных операндов корневой ассоциативно-коммутативной операции в приеме замены. По информационному элементу (модификатор x) определяется переменная x, вводимая перед компиляцией в теорему приема для обозначения остатка корневых операндов, и далее процедура "прогрвыражение" определяет программное выражение для x в формате терма.

11. Символ "копия". x2 имеет вид "копия(t)" и обозначает теоремный терм t, который должен строиться без применения к нему указанных в описании приема нормализаторов. Для получения искомого программного выражения используется процедура "метаперевод", применяемая к терму "фикс(t)".

12. Символ "величина". x2 - десятичная запись числа. Выдается программное выражение, значением которого служит это число в формате десятичных чисел ЛОСа.

13. Символ "логсимвол". x2 имеет вид "логсимвол(x)", где x - переменная. Выдается программное выражение для символа, идентифицированного с x.

Обращение к справочнику "прогрвыражение"

Для обработки прочих случаев используется справочник "прогрвыражение". Ему передаются все входные значения процедуры "прогрвыражение", причем логическим символом обращения служит заголовок терма x2. Таким образом, расширение компилятора при появлении новых заголовков операторных выражений ГЕНОЛОГа происходит путем подключения новых процедур данного справочника.

Логический символ, не являющийся служебным

Логические символы, отличные от служебных, либо выдаются непосредственно, либо (в зависимости от x3) преобразуются к формату терма.

19.12 Компиляция теоремных термов

Чтобы получить программное выражение, значением которого является терм, определяемый заданным подтермом теоремы приема после идентификации его переменных, используется процедура "метаперевод". При построении программного выражения учитываются указатели и нормализаторы приема. Отличие процедуры "метаперевод" от процедуры "прогрвыражение" состоит в том, что там обрабатывались операторные выражения ГЕНОЛОГа, т.е. термы "структурного уровня", здесь же рассматриваются подтермы теоремы приема, т.е. термы "предметного уровня". Исключение составляли лишь обращения к процедуре "прогрвыражение" при компиляции термов теоремного уровня в вычислительном формате.

Операторное выражение "метаперевод(x1 x2)" имеет своими входными параметрами: теоремный терм x1, для которого нужно построить программное выражение, а также программный блок x2. Если внутри x1 размещена служебная конструкция "фикс(t)", то символ "фикс" отбрасывается, а t компилируется без применения к нему нормализаторов приема. В случае, когда терм t должен обрабатываться лишь первыми k из имеющихся нормализаторов, применяется запись "фикс(t k)".

Выйти на начало программы процедуры "метаперевод" можно через пункт "Формирование программного выражения для заданного теоремного терма" оглавления программ.

19.12.1 Развертка в дизъюнкцию квантора существования

В начале процедуры рассматривается один редко встречающийся особый случай - развертка квантора существования, выделенного указателем приема "или", в конечную дизъюнкцию. Обработка его происходит после контрольной точки "прием(2)". Она начинается с обнаружения информационного элемента (или x1 A B). Здесь A - терм "набор($A_1 \ldots A_n$)", составленный из конъюнктивных членов подквантор-ного утверждения квантора существования x1, определяющих перечисление значений переменных связывающей приставки; B - конъюнкция остальных членов подквантор-ного утверждения, играющая роль шаблона дизъюнктивного члена. Вводится вспомогательный программный блок x4, и с помощью процедуры "идентификатор" в нем создается программа для перечисления значений, удовлетворяющих условиям $A_1, \ldots, A_n$. Последние обрабатываются как программно реализуемые антецеденты. Затем относительно x4 находится программное выражение x8 для терма B. С помощью программы блока x4 создается терм "выписка(...)" для набора дизъюнктивных членов, определяемых при перечислении значений выражением B, и далее создается программное выражение x10 для искомой дизъюнкции.

19.12.2 Учет нормализаторов

После контрольной точки "прием(4)" начинается рассмотрение нормализаторов, которые должны применяться к теоремному терму x1. Сначала здесь будет выполнено

рекурсивное обращение к процедуре "метаперевод" с заблокированным учетом нормализаторов, чтобы получить программное выражение, определяющее x1 "в чистом виде", а затем будет компилироваться обработка x1 нормализаторами.

Инициализируется нулем переменная x3. Если x1 имеет вид "фикс(t k)", то переменной x3 переприсваивается значение k, а x1 в этом случае заменяется на t. Таким образом, далее ненулевое значение x3 указывает на ограниченное применение нормализаторов. Если x1 имело вид "фикс(t)", то вход в данную ветвь вообще блокируется - нормализаторы здесь не вводятся.

Находится информационный элемент (быстрпреобр x1 A B C D), содержащий информацию об обработке x1 нормализаторами. Здесь A - список заголовков пакетных нормализаторов и термов "задача(...)", соответствующих обращениям к вспомогательным задачам для преобразования x1. B - набор той же длины, что и A, элементы которого суть наборы программных выражений для остальных входных переменных обращений к пакетным нормализаторам. C - набор той же длины, что и A, составленный из сопровождающих обращения к нормализаторам указателей "условие(...)", "вариант(...)", "лимит(...)". D - либо сигнализатор блокировки нормализации (0 - нет блокировки, 1 - есть), либо программное выражение для результата обработки терма x1 нормализаторами. Информационные элементы (быстрпреобр ...) формируются заблаговременно, по мере того, как появляется возможность компиляции выражений B. Для этого служат обращения к процедуре "учетнормализаторов" из прочих блоков компиляции, в основном из процедуры "идентификатор".

Если ограниченная обработка нормализаторами не предусмотрена, а результат D уже имеется в найденном информационном элементе, то он и выдается в качестве ответа. Иначе - переход через "иначе 2". Здесь проверяется, что D равно 0, и сразу же D заменяется на 1. Таким образом подготавливается блокировка обработки x1 нормализаторами при последующем рекурсивном обращении к процедуре "метаперевод".

Учет цели "вспомописание"

Если в наборе A имеется указатель нормализации "задача(...)", содержащий подтерм "цель(вспомописание($y_1 \ldots y_m$))", то выполняются некоторые предварительные действия. В этом случае нормализатор определяет решение вспомогательной задачи на описание, условие которой получается из утверждения x1 заменой подвыражений $Y_1, \ldots, Y_m$, идентифицированных с переменными $y_1, \ldots, y_m$, на новые переменные $z_1, \ldots, z_m$ - неизвестные задачи. По окончании решения применяется обратная замена z_i на Y_i.

После контрольной точки "прием(29)" вводятся операторы, выбирающие новые переменные $z_1, \ldots, z_m$, и создается информационный элемент (вспомописание ...), перечисляющий программные переменные, значениями которых служат $z_1, \ldots, z_m$. Переменной x5 при этом присваивается копия программного блока x2, в которой исключены старые данные об идентификации переменных $y_1, \ldots, y_m$, а вместо этого указана их идентификация с $z_1, \ldots, z_m$. Далее - откат к переходу через "ветвь 1" для продолжения основной линии компиляции.

Компиляция с заблокированной нормализацией

Переменной x6 присваивается программное выражение для теоремного терма x1, не обработанного нормализаторами. Это выражение получается путем рекурсивно-

го обращения к процедуре "метаперевод". Указателем на блокировку нормализации служит единица, помещенная в разряд D информационного элемента (быстрпреобр x1 ...). Компиляция выполняется с помощью программного блока x5. После учета возможного отказа от компиляции (x6 равно 0) предпринимается обращение к процедуре "нормоператора", корректирующей номера связанных переменных в x6 так, чтобы они были не меньше номера первой неиспользуемой в программном блоке x2 переменной. Вводится информационный элемент (нормализатор x6); если элемент с таким заголовком уже имелся, то его последний разряд заменяется на x6. Затем - переход через "ветвь 2".

Коррекция посылок нормализации

Проверяется наличие информационного элемента (чисткапосылок x1 y T). Такие элементы вводятся упомянутой выше процедурой "учетнормализаторов". Они сопровождают информационные элементы (быстрпреобр x1 ...) и означают наличие указателя нормализации "удалениепосылок(y T)". Последний определяет отбрасывание из списка посылок, передаваемых нормализаторам при обработке x1, всех утверждений y, удовлетворяющих условию T.

Происходит просмотр соответствующих друг другу вхождений x8 и x9 в списки A, B элемента (быстрпреобр x1 A B C D). x10 - текущий элемент списка A; x11 - текущий элемент списка B. Если x10 является обращением к вспомогательной задаче либо к нормализатору, имеющему список посылок, то предпринимается коррекция программного выражения для списка посылок S, расположенного на первой позиции набора x11. Для этого создается вспомогательный программный блок x16, в котором размещается оператор перечисления элементов S, и относительно которого далее компилируется отрицание фильтра T. Таким образом получается программа перечисления элементов S, не удовлетворяющих условию T, и по ней создается программное выражение x21 для скорректированного списка посылок, на которое и заменяется первый разряд набора x11. Затем - откат к переходу через "ветвь 1".

После контрольной точки "прием(55)" происходит учет указателей нормализации "посылки(...)", определяющих дополнительные посылки, используемые нормализаторами. Эти указатели усматриваются через информационные элементы (посылки x1 N); N - набор утверждений, добавляемых к посылкам. Переменной x8 присваивается список программных выражений для элементов списка N, и далее реализуется цикл просмотра наборов A, B из элемента (быстрпреобр x1 ...), аналогичный предпринятому выше для элемента (чисткапосылок x1 ...). Программные выражения для списка посылок, упоминаемые в B, корректируются путем присоединения утверждений списка x8. Предусмотрена обработка служебного символа "одз", который может встретиться в N и перейти в x8. Тогда присоединяются все утверждения из области допустимых значений терма x1. Последние задаются программным выражением "Одз(x6)".

Цикл обработки обращений к нормализаторам

После контрольной точки "прием(41)" реализуется цикл просмотра нормализаторов, упоминаемых в информационном элементе (быстрпреобр x1 A B C D). Переменной x7 присваивается текущая позиция списка A; x8 - текущая позиция в B; x9 - текущая позиция в C. Если x3 отлично от 0, то проверяется, что номер текущего элемента не превосходит x3.

Если текущий набор списка C содержит элемент "лимит(...)", то в компилируемую программу вводится соответствующий оператор "лимит(...)". Если обработка текущим нормализатором будет занимать время, большее указанного в данном операторе, то произойдет откат. По завершении компиляции обращения к нормализатору будет выполнен откат к "ветвь 2", где произойдет вставка оператора "Обрыв", отменяющего контроль трудоемкости применения нормализатора после получения результата.

Далее - переход через "ветвь 3", где процесс компиляции разветвляется. Рассматриваются следующие случаи:

1. На позиции x7 расположен логический символ - название некоторого пакетного нормализатора f (контрольная точка "прием(48)"). Тогда переменной x10 присваивается оператор для обработки нормализатором f текущего терма. Его первым операндом служит программное выражение x6 (текущий терм), прочие операнды берутся из набора по вхождению x8. После коррекции обращения x10 для редко встречающейся ситуации компиляции приемов вывода теорем - переход через "ветвь 2".

   Здесь выполняется такое переобозначение связанных переменных оператора x10, чтобы их номера были не меньше номера первой неиспользуемой программной переменной блока x2.

   Затем происходит учет информационного элемента (нормусм P Q). У него P - некоторое утверждение; Q - программное выражение для результата обработки P указанными в приеме нормализаторами. Если обращение к нормализатору сопровождалось вводом дополнительных посылок, одна из которых совпадает с утверждением P либо его отрицанием, то предпринимается контроль ложности этой посылки. Для этого x10 заменяется на условное выражение "вариант(...)", анализирующее заголовок утверждения, являющегося значением Q. Если этот заголовок - логическая константа, для которой дополнительная посылка имеет значение "ложь", то значением условного выражения становится константа "ложь", иначе - сохраняется значение x10. Смысл преобразования x10 состоит в экономии на обращениях к пакетным нормализаторам, если согласно контексту они не требуются. Так может случаться при компиляции дизъюнкций и условных выражений.

   Далее - переход через "ветвь 1" и обращение к справочнику "быстрпреобр" для определения формата x11 нормализатора f. Здесь происходит еще одно разветвление компиляции.

   Сначала рассматривается случай, в котором формат x11 содержит элемент "коррекцияпосылок". Этот элемент означает, что нормализатор использует комментарий (коррекцияпосылок M), сохраняющий информацию об утверждениях, выведенных из посылок нормализатора при его применении. Такие посылки могут возникать для поддержания сопровождения по о.д.з. преобразуемого терма. В свою очередь, далее подслучай разбивается на два - сначала рассматривается случай компиляции приема нормализатора, затем - компиляция приемов других типов. Начнем с рассмотрения первого из них.

   Если текущий набор списка C не содержит элементов "условие(...)", "вариант(...)", "фильтр(...)", управляющих обращениями к нормализатору в зависимости от внешних обстоятельств, то происходит лишь ввод оператора, корректирующего список посылок нормализатора с помощью процедуры "кор-

рекцияпосылок". Последняя выполняется на основе содержимого комментария (коррекцияпосылок ...), в котором к данному моменту компиляции уже содержатся те новые посылки, которые должен ввести прием.

Если же текущий набор списка C содержит элемент одного из указанных типов, то переход через "иначе 4". Здесь происходит просмотр элементов x16 списка C, имеющих заголовки "фильтр", "условие", "вариант". Первый терм любого такого элемента представляет собой некоторое условие U на текущий контекст, заданное на языке фильтров приема. Переменной x17 присваивается оператор, проверяющий данное условие.

Если x16 имеет заголовок "фильтр", то x17 заносится в компилируемую программу, блокируя применение приема при нарушении условия.

Если x16 имеет заголовок "условие", то в программу заносится оператор, присваивающий при выполнении условия U новой программной переменной X нормализованное значение x10 и корректирующий в этом случае список посылок, а иначе - передающий X старое значение x6, минуя обращение к нормализатору. Затем переменной x10 переприсваивается X.

Если x16 имеет заголовок "вариант", то находится программное выражение x20 для обращения к другому нормализатору, которое будет выполняться при выполнении условия U. Далее в компилируемую программу заносятся: оператор, присваивающий новой переменной X значение условного выражения, обращающегося к нужному нормализатору в зависимости от U, и оператор, выполняющий коррекцию посылок.

Если x11 не имеет элемента "коррекцияпосылок" либо компилируемый прием не является приемом нормализатора, то выполняются другие ветви, начинающиеся, соответственно, с перехода через "иначе 2" и "иначе 3". Действия здесь аналогичны разборанным выше, но проще: не рассматривается случай заголовка "фильтр" и не выполняются коррекции посылок.

По завершении рассмотрения указанных выше альтернатив - переход через "ветвь 1", расположенную непосредственно перед присвоением значения переменной x11.

Сначала здесь проверяется наличие указателя приема "различны(...)", требующего, чтобы текущий нормализатор f изменял подаваемый на него терм. Если такой указатель есть, то в компилируемую программу вставляется оператор, проверяющий различие значений x10 (после преобразования) и x6 (до преобразования).

В заключение текущего шага цикла переменной x6 переприсваивается значение x10. Если выражение x10 оказывается чрезмерно длинным, то для него вводится вспомогательная переменная. Затем корректируется информационный элемент (нормализатор x6) - соответственно изменению x6, и откат к продолжению цикла просмотра нормализаторов.

2. На позиции x7 расположен терм "задача($A_1 \dots A_n$)", определяющий обращение к вспомогательной задаче для преобразования текущего терма x6 (контрольная точка "прием(40)"). Этот терм присваивается переменной x10; переменной x11 присваивается текущий набор списка B. Для обращения к вспомогательной задаче из приема сканирования задачи будут использоваться процедуры

"преобразование" и "вспомописание". Первая из них создает вспомогательную задачу на преобразование, вторая - применяется, если некоторое A_i имеет вид "тип(описать)", и создает задачу на описание. Если обращение происходит из пакетного оператора, то берутся упрощенные версии - "вспомпреобразование" и "быстроописание". Таким образом, основную часть обработки текущего нормализатора составляет компиляция входных данных для указанных процедур.

Переменной x12 присваивается накопитель программных выражений для построения списка целей вспомогательной задачи. Кроме собственно целей, в нем будут использоваться также специальные информационные элементы, воспринимаемые процедурами, выполняющими создание и решение вспомогательной задачи. Накопитель представляет собой пару: первым элементом ее служит набор программных выражений для отдельных целей задачи; вторым элементом - набор программных выражений для групп целей. Сначала в накопителе регистрируется элемент (уровеньобращения A_1).

Далее происходит просмотр элементов A_i, определяющих цели и комментарии задачи. Текущий такой элемент присвоен переменной x13. Элементы с заголовками "коррекцияпосылок", "тип", "посылка" в этом просмотре пропускаются. В зависимости от заголовка A_i, рассматриваются следующие подслучаи:

(a) Элемент A_i имеет вид "цель(C)". Тогда предпринимается обращение к справочнику "подцель" на логическом символе - заголовке C. Справочник определяет пару, первый элемент которой - программное выражение для отдельной цели вспомогательной задачи либо для группы целей; второй элемент - указатель типа первого элемента (1 - отдельная цель, 2 - группа целей). Соответственно, пополняется первый или второй набор накопителя x12.

(b) Элемент A_i имеет вид "комментарий(условие(U)$K_1 \ldots K_m$)". Терм "условие(U)" определяет условие, при выполнении которого новая задача должна снабжаться комментарием ($K_1 \ldots K_m$). Этот терм может отсутствовать. В x12 комментарий S регистрируется как программное выражение для пары (комментарий S).Если комментарий начинается с символа "титр", т.е. используется для текстового сопровождения процесса решения, то программное выражение x18 для него создается процедурой "транститр". Иначе происходит раздельное рассмотрение случаев наличия либо отсутствия терма "условие(...)". Операторы для проверки условия U определяются процедурой "фильтр"; программное выражение для комментария находится при помощи процедуры "транскоммент".

(c) Прочие элементы A_i рассматриваются как термы "теоремного" уровня и регистрируются в x12 после применения к ним процедуры "метаперевод". Исключение составляют однобуквенные термы, которые переносятся в x12 непосредственно.

По окончании цикла обработки элементов A_i - переход через "иначе 2". Здесь происходит регистрация в накопителе x12 комментария (коррекцияпосылок ...), если такой комментарий уже был введен на предыдущих этапах компиляции. Затем компиляция разветвляется. Если создается программа приема пакетного оператора, то в нее заносится обращение к процедуре "быстроописание" либо "вспомпреобразование". Если создается программа приема вывода теорем,

то аналогичным образом используются процедуры "Вспомописание" и "вспомпреобразование". Наконец, для приема сканирования задачи применяются обращения к процедурам "вспомописание" и "преобразование". Во всех этих случаях x6 заменяется на результат обращения к процедуре, и далее - откат к следующему шагу цикла просмотра нормализаторов.

3. Элемент A_i представляет собой переменную, значением которой служит название применяемого нормализатора. Эта возможность используется крайне редко - только в приемах вывода теорем. С помощью процедуры "прогрвыражение" определяется программное выражение x11 для названия нормализатора; создается программное выражение x12 для обращения к нормализатору, и x6 заменяется на x12.

Завершение цикла обработки терма нормализаторами

По окончании цикла просмотра нормализаторов выполняются два несложных дополнительных преобразования. Во-первых, проверяется наличие информационных элементов (свертка ...), определяющих выполнение некоторой подстановки в результирующий терм. Если эти элементы есть, причем текущее обращение к процедуре "метаперевод" не является внутренним звеном цепи рекурсивных обращений к этой же процедуре, то x6 заменяется на результат применения к x6 соответствующей подстановки. Во-вторых, если компилируемый терм является заменяющим термом приема и имеются информационные элементы (учетзнака ...), (знаксуммы ...), указывающие, что нужна коррекция знака терма, то вводится обращение к процедуре "измзнака", обеспечивающей эту коррекцию. Затем выдается результат компиляции x6.

Далее продолжается рассмотрение той ветви процедуры "метаперевод", которая обеспечивает получение программного выражения для терма x1 до его нормализации. Переход к этой ветви - через оператор "ветвь 1", расположенный перед контрольной точкой "прием(4)". Сразу же здесь вводится индикатор x3 заголовка "фикс" терма x1, после чего заголовок "фикс" (если он есть) отбрасывается.

19.12.3 Случай однобуквенного терма

После контрольной точки "прием(5)" рассматривается случай однобуквенного терма x1. Если заголовок x1 - логический символ, то в качестве ответа выдается сам терм x1. Исключение составляют служебные слова "свой", "чужой", обозначающие в теоремах шахматных приемов цвет фигур. Они компилируются с помощью операторных выражений "свойцвет(...)", "чужойцвет(...)".

После контрольной точки "прием(7)" начинается рассмотрение случая, когда заголовок x1 - переменная x. Эта переменная присваивается программной переменной x4.

Переменная выделена указателем "свертка"

Сначала рассматривается крайне редкий подслучай выделения переменной указателем "свертка". Такая переменная вводится в антецеденте "равно(x a)" и идентифицируется как новая переменная . По окончании определяемых нормализаторами вспомогательных преобразований терма, содержащего x, в этот терм подставляется

вместо x выражение a. Во время преобразований считаются истинными заданные в указателе "свертка" сопровождающие x посылки.

Если найден информационный элемент (свертка x A B C), то проверяется, что программное выражение A для теоремного выражения a уже найдено (отлично от 0). Если имеется (отлично от 0) также программное выражение C для новой переменной, идентифицированной с x, то выдается ответ C. Если же C равно 0, то в компилируемую программу заносится оператор, выбирающий новую переменную; по нему доопределяется C; по конъюнкции B сопровождающих посылок находятся набор x9 программных выражений для этих посылок; такими посылками пополняются информационные элементы (быстрпреобр ...), относящиеся к нормализации содержащих x термов, и снова выдается ответ C.

Общий случай уже идентифицированной переменной

После контрольной точки "прием(10)" и до фрагмента с контрольной точкой "прием(12)" рассматриваются случаи, в которых идентификация переменной x уже зарегистрирована в информационном элементе с заголовком одного из видов: "вхождение", "унисборка", "терм", "переменная", "наборчленов", "наборпеременных". Здесь находится программное выражение для представления значения x в формате терма, которое и выдается как ответ. В случае элемента "вхождение" предпринимается попытка сразу же получить небольшую экономию за счет непосредственного преобразования выражений "первыйоперанд", "второйоперанд", ... в выражения "первыйтерм", "второйтерм", ...

Ввод новой связанной переменной

После контрольной точки "прием(12)" рассматривается случай, когда еще не идентифицированная переменная x4 представляет собой связанную переменную заменяющего терма приема замены. Тогда в компилируемую программу вводятся операторы, выбирающие новую переменную для идентификации ее с x4.

Переменная, обозначающая функцию

После контрольной точки "прием(11)" рассматривается практически не используемая возможность идентифицировать функциональную переменную как терм "отображение(...)", задающий функцию. Не используется она, в частности, из-за того, что для построения терма "отображение(...)" необходимо идентифицировать и зарегистрировать в информационном элементе (функция x4 ...) условия на область определения функции. Однако, функциональные переменные нужны лишь в качестве шаблонов для варьирования подтермов, и никакой области определения с ними обычно не связывается. Тем не менее, в компиляторе предусмотрена возможность построения указанного терма "отображение(...)" по полностью укомплектованному информационному элементу (функция x4 ...). Сначала здесь определяется программное выражение x4 для связывающей приставки описателя "отображение", затем создается программное выражение x8 для результирующего терма.

Переменная, обозначающая матрицу

Если имеется информационный элемент (элементыматрицы x4 ...), определяющий программное выражение для набора наборов термов, образующих элементы матрицу,

то создается результирующее программное выражение x7 для терма, определяющего саму матрицу.

Выбор новой переменной согласно указателю приема

После контрольной точки "прием(13)" рассматривается ситуация, в которой переменная x4 выделена специальным указателем приема, определяющим выбор для нее новой переменной либо группы новых переменных. Переменной x8 присваивается программное выражение для списка использованных в контексте срабатывания приема "теоремных" переменных. Если указатель приема "переменные(x4 n)" определял идентификацию x4 с набором переменных длины n, то находится программное выражение x11 для n, и в компилируемую программу заносится обращение к операторному выражению "кортежпеременных(...)", выполняющему выбор группы новых переменных. Переменная x4 идентифицируется одновременно и как терм "набор(...)", перечисляющий новые переменные, и как список этих переменных. Аналогичным образом обрабатывается указатель "элементыматрицы(...)", определяющий идентификацию x4 с матрицей новых переменных. В остальных случаях применяется либо оператор "обозначения", выбирающий новую переменную с учетом типа ее прорисовки, либо оператор "новаяпеременная".

Переменная идентифицирована при вычислениях

Если имеется информационный элемент (транслвыражения x4 ...), то значение переменной x4 переводится из вычислительного формата в формат терма.

Прочие случаи

Если переменная x4 идентифицирована при унификации, то для нахождения соответствующего ей терма используются информационные элементы (Таблзначение ...) и (унификация ...).

Если x4 идентифицирована с помощью информационного элемента (значение x4 T), в неопределенном формате значений, то выдается программное выражение T.

19.12.4 Специальные случаи неоднобуквенного терма

Использование ранее проведенной компиляции терма

После контрольной точки "прием(14)" рассматривается случай, в которых терм x1 уже компилировался, причем информационный элемент (прогртерм x1 ...) сохраняет результат этой компиляции. Тогда повторно выдается данный результат. Если имеется информационный элемент (выражение x1 терм ...), означающий, что терм x1 был идентифицирован как единое целое, без идентификации составляющих его подтермов, то тоже выдается извлекаемое из данного элемента выражение для x1. Исключение здесь составляет ситуация, когда указатель приема "сборка" блокирует применение в заменяющей части копий равных им подтермов заменяемой части.

Заменяющий терм приема группировки

Если прием имел указатель "набор(второйтерм)", то его заменяющий терм создается на основе информационного элемента (перечень x1 ...). Проверяется, что последний

разряд этого элемента отличен от нуля; тогда он представляет собой программное выражение для заменяющего терма и выдается как результат.

Длина набора

Если x1 имеет вид "длинанабора(x)", то после контрольной точки "прием(36)" предпринимается попытка найти информационный элемент (наборчленов набор x B), в котором B - программное выражение для набора корневых операндов выражения "набор(...)", с которым была идентифицирована переменная x. Результатом является программное выражение x6 для длины набора B.

Коэффициенты многочлена

Если x1 имеет вид "коэффициенты(x)", то после контрольной точки "прием(39)" предпринимается попытка найти информационный элемент (смногочлен x A), задающий программное выражение A для набора термов - коэффициентов многочлена x. По этому выражению строится выражение для терма "набор(...)", перечисляющего коэффициенты.

Учет указателя "развертка"

После контрольной точки "прием(37)" рассматривается случай, когда терм x1 должен создаваться в режиме "развертки", т.е. вместо указанной в теореме одноместной корневой операции над набором (или квантора общности) будет использоваться соответствующая двуместная ассоциативная корневая операция f (в случае квантора общности - конъюнкция). Компиляция разветвляется, в зависимости от того, является ли заголовком x1 квантор общности.

1. После контрольной точки "прием(53)" рассматривается случай, когда заголовок x1 отличен от квантора общности. Тогда x1 - либо терм "отображение(...)", либо одноместная операция от такого терма. В первом случае f полагается равным символу "набор", во втором - определяется с помощью справочника "развертка". Найденное значение присваивается переменной x6. Если прием имел указатель "комплексное", то вместо операций "плюс", "умножение" берутся их комплексные аналоги. Затем - переход через "ветвь 4". Здесь переменной x7 присваивается корневое вхождение в x1 символа "отображение".

   Далее компиляция снова разветвляется.

   (a) Сначала рассматривается случай, когда перечисление ведется по единственному параметру, т.е. терм по x7 имеет вид "отображение(i $A(i)$ $t(i)$)". Параметр i присваивается переменной x8, вхождение $A(i)$ - переменной x9. Здесь компиляция разветвляется еще раз.

      i. Рассматривается случай, когда $A(i)$ определяет перечисление целочисленных значений i в заданном конечном промежутке. Переменной x14 присваивается терм для начала этого промежутка, переменной x12 - для его конца. Находится пара x15 программных выражений для десятичных чисел ЛОСа - значений x14 и x12.
      Если $t(i)$ имеет вид "значение(v i)", причем переменная v была идентифицирована с набором термов N при помощи информационного элемента (вектор v ...), а промежуток x14, x12 - от начала до конца этого

набора, то выдается результат, в котором операция x6 применяется к элементам набора N. Особо рассматривается случай символа "перечень".

В противном случае для копиляции текущего терма $t(i)$ при перечислении значений i вводится вспомогательный программный блок x16. Процедура "метаперевод" дает программное выражение x17 для текущего $t(i)$, после чего строится программное выражение x19 для всего набора термов $t(i)$. Далее результат находится аналогично предыдущему случаю. Для обеспечения корректности сопровождения срабатываний приемов поясняющими текстами оператор, содержащий x19, помещается в "скобки" из вспомогательных процедур "титрбуф".

ii. Альтернативный случай - $A(i)$ перечисляет все разбиения i заданного натурального числа n в сумму неупорядоченных по возрастанию натуральных чисел. Для перечисления значений i используется процедура "Разбиения(...)"; в остальном схема компиляции такая же, как при перечислении конечного отрезка целых чисел.

(b) После контрольной точки "прием(57)" рассматривается случай, когда перечисление ведется по двум параметрам. Здесь терм по вхождению x7 имеет вид "отображение(i j и(принадлежит(i $\{m,\ldots,n\}$) принадлежит(j $\{p,\ldots,q\}$)) $t(i,j)$)". Хотя массив $t(i,j)$ двумерный, он разворачивается в одномерный список термов, рассматриваемый как набор операндов операции x6.

Индексы i,j присваиваются переменным x8 и x9; границы диапазонов m,n,p,q присваиваются, соответственно, переменным x13, x14, x15 и x16. Для компиляции текущего терма $t(i,j)$ вводится вспомогательный программный блок x18, программа которого инициирована операторами "Номера", перечисляющими значения i,j. Программное выражение для текущего терма присваивается переменной x19, и далее находится программное выражение x21 для набора операндов операции x6.

2. После контрольной точки "прием(8)" рассматривается случай, когда x1 - кванторная импликация. Переменной x6 присваивается список антецедентов импликации x1. Затем компиляция разветвляется.

(a) Сначала рассматривается случай перечисления по единственному параметру i. Здесь x1 имеет вид "длялюбого(i если $i \in \{m,\ldots,n\}$ то $A(i)$)".

Индекс i присваивается переменной x8; границы диапазона перечисления m,n присваиваются переменным x9, x10. Если x1 содержит еще не идентифицированную функциональную переменную $X(i)$, причем имеется указатель приема "переменные(X ...)", определяющий выбор для X набора новых переменных, то в компилируемую программу вставляется оператор, реализующий такой выбор. Затем - откат к переходу через "ветвь 2".

Переменной x11 присваивается пара программных выражений для границ диапазона m,n. Вводится вспомогательный программный блок x13 для компиляции текущего утверждения $A(i)$, и переменной x14 присваивается программное выражение для этого утверждения. Переменной x16 присваивается программное выражение для набора утверждений $A(i)$. Если это выражение не чрезмерно громоздко, то по нему строится результирующее

выражение для конъюнкции утверждений. Иначе - x16 использоваться не будет, а создание набора утверждений реализуется по схеме заполнения накопителя. Новой переменной (накопителю) присваивается пустое слово. Затем помещается оператор "ветвь" для перехода по завершении перечисления; после него помещаются реализующие перечисление операторы из вспомогательного программного блока x13, и далее - операторы, заносящие в накопитель очередное $A(i)$ (т.е. x14).

(b) Рассматривается случай перечисления по двум параметрам. x1 имеет вид "длялюбого($i\ j$ если $i \in \{m, \ldots, n\}$ & $j \in \{p, \ldots, q\}$ то $A(i, j)$)". Этот случай аналогичен уже рассмотренным выше.

Заменяющий терм для группы антецедентов

Этот крайне редкий случай обрабатывается после контрольной точки "прием(15)". x1 - конъюнкция либо дизъюнкция, представляющая собой заменяющий терм приема. Заменяемые термы представляют собой операнды конъюнкции, либо дизъюнкции, либо антецеденты кванторной импликации, рассматривавшиеся при идентификации как члены некоторой фиктивной конъюнкции или дизъюнкции. Чтобы получить в такой ситуации заменяющий терм, используется специальный оператор "логзамена".

Компиляция условного выражения с непосредственной проверкой условия

Если x1 имеет вид "вариант($A\ t_1\ t_2$)", причем указатель приема "нормвариант(...)" определяет программную реализацию проверки условия A, то результирующее программное выражение будет определять не условное выражение x1, а одно из выражений t_1, t_2, в зависимости от результатов проверки. Обработка данного случая начинается после контрольной точки "прием(60)". Чтобы получить программу проверки A, создается вспомогательный программный блок x6, и выполняется обращение к процедуре "идентификатор" с установкой (программа ...) на идентификацию A. Находятся программные выражения x11, x12 для выражений t_1, t_2, и создается результирующее условное программное выражение x13.

Компиляция дизъюнкции либо конъюнкции с учетом возможности вырождения ее из-за появления логической константы

Если компилируется конъюнкция либо дизъюнкция утверждений $A_1, \ldots, A_n$, обрабатываемых нормализаторами, то после того, как получен результат нормализации некоторого A_i, обработка последующих утверждений может оказаться излишней. Это произойдет, если данный результат будет представлять собой константу "ложь" для конъюнкции либо константу "истина" для дизъюнкции. Чтобы ввести соответствующую экономию, используется специальная ветвь компиляции. Она начинается после контрольной точки "прием(3)". Находится программное выражение x6 для A_1, и далее в цикле компиляции утверждений $A_2, \ldots, A_n$ формируется программное выражение для результата. На текущем шаге переменная x8 имеет своим значением программную переменную, значением которой служит частичная конъюнкция либо дизъюнкция R - от первого до текущего утверждения. При рассмотрении очередного A_i находится программное выражение для него, присваиваемое переменной x11. Новая версия R представляет собой условное выражение: если заголовком ранее найденной версии R является логическая константа K, обращающая в константу всю

конъюнкцию либо дизъюнкцию, то эта константа сохраняется, иначе к R добавляется новый член x11. При добавлении логической константы K новая версия R также будет равна K. Таким образом, при появлении на некотором шаге константы K необходимые для получения последующих утверждений A_i обращения к нормализаторам будут пропущены.

Идентифицировано вхождение терма

После контрольной точки "прием(16)" рассматривается случай, когда имеется ссылка на ранее идентифицированное вхождение v терма, равного x1. При выполнении определенных дополнительных условий, в этом случае можно выдать в качестве результата подтерм по вхождению v. В частности, должны отсутствовать нормализаторы, применяемые к подтермам x1. Программное выражение для v извлекается из информационного элемента (операнд ...).

Регистрация обычной переменной как функциональной

Перед рассмотрением серии случаев, в которых x1 имеет вид "значение(f t)", предпринимается учет указателя "функция(f x)", определяющего переход от обычной переменной f к функциональной. Данный указатель присваивается переменной x6. Проверяется отсутсвие информационного элемента (функция f ...), означающего, что уже имела место регистрация f как функциональной переменной. Находятся программное выражение x10 для переменной x - аргумента f как функциональной переменной, а также x11 - для вхождения v, по которому f была идентифицирована как обычная переменная. В компилируемую программу вводятся операторы, проверяющие, что терм по вхождению v имеет вид "отображение(x $a(x)$ $t(x)$)". Эта информация позволяет создать информационный элемент (функция f (x) ...), в котором имеются ссылки на вхождение выражения $t(x)$ для значений функции, а также на список конъюнктивных членов утверждения $a(x)$, определяющий область определения. После создания такого информационного элемента - откат к переходу через "ветвь 1". Фактическая компиляция x1 будет выполняться нижеследующими пунктами.

Одз для заданного терма

Если x1 имеет вид "одз(t)", то с помощью операторного выражения "Одз" находится список утверждений для области допустимых значений терма t, и выдается программное выражение для конъюнкции этих утверждений.

Заголовок "значение"

После контрольной точки "прием(17)" разбирается серия случаев компиляции выражения x1, имеющего вид "значение(f t)". Переменная f присваивается программой переменной x4, и далее компиляция разветвляется

1. Имеется информационный элемент (элементыматрицы f ...), определяющий идентификацию f как набора наборов термов - элементов прямоугольной матрицы (контрольная точка "прием(47)"). Выражение t для координат элемента матрицы имеет вид "набор(i j)". Предполагается, что после обработки нормализаторами выражения i, j суть десятичные константы. Создается программное

выражение x9, извлекающее из матрицы требуемый терм. В нем используется двухэтапное преобразование термов i, j - сначала к виду десятичного числа ЛОСа, а затем к виду символьного числа для выделения нужного элемента набора.

2. Имеется информационный элемент (вектор f ...), определяющий идентификацию f как набора термов (контрольная точка "прием(51)"). Предполагается, что терм t после обработки нормализаторами становится десятичной константой. Для извлечения нужного элемента набора строится программное выражение x8.

3. Имеется информационный элемент (наборпеременных f ...), определяющий идентификацию f с набором переменных (контрольная точка "прием(52)"). В этом случае предполагается, что t есть переменная, идентифицированная с десятичным числом (вычислительный формат ГЕНОЛОГа "десчисло"). Для извлечения нужной переменной набора строится программное выражение x8.

4. Имеется информационный элемент (транслвыражения f ...), определяющий идентификацию f с числовым вектором в вычислительном формате "вектор(число)". Этот формат - пара, состоящая из ссылки на массив чисел формата "с плавающей запятой" и на целое число в машинном формате, равное длине вектора. Строится программное выражение x9, извлекающее нужный элемент массива и переводящий его в формат терма.

5. x1 имеет вид "значение(значение(f i)t)", причем имеется информационный элемент (функции f A), определяющий идентификацию f как набора функций (контрольная точка "прием(61)"). Здесь A - программная переменная для набора пар (X_i, T_i), где T_i - выражение для значения i - й функции; X_i - набор переменных, рассматриваемых как аргументы функции. Переменной x9 присваивается программное выражение для набора t значений, подставляемых вместо аргументов i-й функции; переменной x10 - выражение для i. Затем переменной x11 присваивается i - я пара набора A, и в компилируемую программу заносится оператор, находящий результат подстановки в T_i вместо аргументов X_i их значений t.

6. f - переменная, идентифицированная с термом "набор($A_1 \dots A_n$)" (контрольная точка "прием(62)"), а t - переменная, идентифицированная с десятичным числом. В этом случае создается программное выражение x11 для выделения операнда A_t.

   После контрольной точки "прием(63)" переменной x5 присваивается терм t и находится информационный элемент (функция f X ...), определяющий идентификацию f. Из этого элемента извлекается пара x7, определяющая программное выражение R для значений функции. Первый элемент пары - символ "терм" либо "вхождение" либо "функподст", уточняющий способ получения значения по x7. При необходимости предпринимается коррекция связанных переменных в R - чтобы их номера были не меньше номера первой не определенной переменной программного блока.

7. f - функциональная переменная, выделенная указателем "вхождение" (контрольная точка "прием(45)"). В этом случае теорема приема имеет подтерм

$f(x,s)$ (либо $f(s)$), где s идентифицируется с произвольным подтермом терма, идентифицирующего для $f(x,s)$. Находится информационный элемент (вхождениетерма f A), у которого A - программное выражение для вхождения s. В соответствии с видом указанного подтерма, x1 имеет вид $f(P,Q)$ либо $f(Q)$. Оно обозначает результат замены s на Q и одновременную подстановку вместо всех вхождений переменной x, расположенных вне s, терма P. Рассматриваются различные возможности идентификации P. Во всех случаях результат получается с помощью оператора "подстзамена" либо "Подстзамена"; если же P отсутствует, то используется оператор "внутрзамена".

Перед контрольной точкой "прием(64)" переменной x7 переприсваивается терм, определяющий значения функциональной переменной f.

8. Список X аргументов функциональной переменной f, согласно ее идентификации, совпадает с t. Тогда результатом компиляции служит терм x7.

9. Список X аргументов функциональной переменной f одноэлементный и состоит из переменной x, идентифицированной с помощью информационного элемента (наборпеременных x P). Определяется программное выражение x11 для набора t значений аргументов, и находится результат подстановки x11 вместо P в x7.

10. t содержит переменную, идентифицированную по указателю "унификация"; контрольная точка "прием(44)". В этом случае $f(t)$ определяется с помощью подстановки, определяемой информационным элементом (унификация ...).

11. После контрольной точки "прием(65)" рассматривается ряд подслучаев, в которых результирующий терм строится как подстановка в шаблон x7 вместо аргументов функциональной переменной f термов, определяемых выражением t. Прежде всего, находится набор x8 последних термов, а также набор x9 аргументов f. Те позиции, на которых эти наборы совпадают, отбрасываются. Если набор x8 оказывается пуст, выдается результат x7. Если наборы x8 и x9 одноэлементны, причем имеются информационные элементы, позволяющие сразу идентифицировать их переменные, то находится указанная выше подстановка в x7. Иначе - переход через "ветвь 2", где наборы x10, x11 программных выражений для аргументов и подставляемых термов определяются с помощью процедур "прогрвыражение" и "метаперевод". Затем снова находится подстановка. Особо рассмотрен крайне редкий подслучай выделения терма x1 указателем приема "мультизамена".

12. После контрольной точки "прием(32)" рассматривается случай, в котором переменная f выделена указателем "заменатермов". По этому указателю, ссылающемуся на идентифицируемое вхождение v в теорему приема терма $f(A)$, был создан информационный элемент (заменатермов f v B C). Здесь B,C - программные выражения для идентифицирующего терма, соответствующего $f(A)$, и для набора вхождений в этот идентифицирующий терм терма, с которым идентифицировано A. Результат компиляции $f(t)$ - терм, полученный из B заменой всех вхождений набора C на терм, определяемый по t.

13. Переменная f выделена указателем "смногочлен" (контрольная точка "прием(33)"). Информационный элемент (смногочлен f ...) определяет набор термов - коэффициентов многочлена. Переменной x6 присваивается программное

выражение для t. Построение многочлена по набору коэффициентов и выражению x6 выполняется процедурой "сборкамногочлена" либо (в комплексном случае) "Сборкамногочлена").

14. Переменная f выделена указателем "функподст" (контрольная точка "прием(46)"). Это означает, что в идентифицируемой части теоремы выделен подтерм $f(g(A)B)$ либо $f(g(A))$, и в соответствующий ему идентифицирующий терм T переменная g входит только как функциональная переменная. Тогда выражения $f(t\ B)$ (соответственно, $f(t)$) обозначают термы, получающиеся из T заменой всех вхождений $g(\ldots)$ на t.

    Извлекается информационный элемент (функподст $f\ B\ C$), у которого B - программное выражение для T; C - программное выражение для набора вхождений в T функциональных переменных $g(\ldots)$. Находится программное выражение x7 для t, и далее предпринимается указанная выше замена вхождений.

15. Переменная f выделена указателем "Бинарнаяоперация". Терм $f(a,b)$ идентифицируется либо с термом вида $F(A,B)$, либо с термом вида $G(F(A,B))$, где F,G - логические символы. Порядок расположения операндов A,B по отношению к a,b - произвольный. Переменная f (как обычная переменная) идентифицируется с логическим символом F.

    Информационный элемент (Бинарнаяоперация $f\ C_1\ C_2\ C_3$) имеет следующие компоненты: C_1 - программное выражение для символа F; C_2 - программное выражение для символа G (если одноместной операции нет, то берется символ 0); C_3 - программное выражение для индикатора перестановки операндов (1 - есть перестановка, 0 - нет). x1 имеет вид $f(p,q)$. Находятся программные выражения x7, x8 для p и q. По ним и F,G, с учетом индикатора перестановки, собирается программное выражение x9 для результирующего терма.

16. Переменная f выделена указателем "символ" либо "симметрично" (контрольная точка "прием(35)"). Эта переменная идентифицируется с логическим символом s, программное выражение для которого определяется информационным элементом (значение $f\ \ldots$). Обозначение $f(t)$ понимается как терм вида $s(A_1 \ldots A_n)$, где операнды $A_1,\ldots,A_n$ определяются по t. В случае указателя "симметрично" допускается идентификация без учета порядка операндов, но тогда фактический порядок сохраняется в информационном элементе (новоперанд $f\ \ldots$), чтобы новые термы вида $f(t)$ создавались с сохранением этого порядка.

17. Переменная f выделена указателем "типзначения". Это означает, что теорема имеет антецедентом кванторную импликацию; консеквент ее имеет вид $f(s)$, и f идентифицируется с логическим символом - типом значения выражения s. Программное выражение для этого символа сохраняется в информационном элементе (значение $f\ \ldots$). Далее терм $f(t)$ компилируется так же, как в случае указателя "символ".

Заголовок "область"

В тех чрезвычайно редких случаях, когда функциональная переменная f идентифицируется с явным указанием области значений своих аргументов, предусмотрена

компиляция выражения "область(f)" (контрольная точка "прием(18)"). Если имеется информационный элемент (область f ...), то результат извлекается из него. Иначе - переменной x6 присваивается программное выражение для набора аргументов функции; проверяется, что информационный элемент (функция f ...) имеет программное выражение для списка утверждений, задающих область определения функции, и строится программное выражение x7 для представления области определения в виде описателя "класс(...)".

Квантор либо описатель

После контрольной точки "прием(19)" рассматривается случай, когда заголовком x1 служит квантор или описатель.

Если имеется указатель приема "матрица", выделяющий x1, то результатом компиляции должно быть программное выражение для прямоугольной матрицы, заданной с помощью операции "строки(...)" над набором наборов элементов. Этот подслучай рассматривается после контрольной точки "прием(56)". Здесь x1 имеет вид "отображение(i j и(принадлежит(i $\{m,\dots,n\}$)принадлежит(j $\{p,\dots,q\}$)) $a(i,j)$)". Находятся программные выражения x8,x9,x11,x12 для границ диапазонов изменения индексов. Чтобы получить программное выражение x16, дающее текущий элемент матрицы $a(i,j)$, создается вспомогательный программный блок x15.

Если указанный особый подслучай не имеет места, то переход через "ветвь 2". Здесь переменной x4 присваивается связывающая приставка квантора либо описателя. Проверяется, что все свободные переменные x1, для которых не предусмотрен выбор идентифицирующих их новых переменных, уже определены.

Сначала рассматривается особый подслучай, когда связывающая приставка x4 состоит из единственной переменной x, выделенной указателем "связприставка". Предполагается также, что x1 имеет вид $\forall_x P(x)$, а переменная P выделена указателем "импликация". Это - редко встречающийся случай компиляции приемов вывода теорем. Результат x7 компиляции $P(x)$ определяет вспомогательный терм вида "то(A_0 $A_1 \dots A_n$)", где A_0 - консеквент; $A_1,\dots,A_n$ - антецеденты результирующего квантора общности. Для получения этого квантора применяется процедура "квантимплик". Роль связывающей приставки квантора играют все свободные переменные его подкванторных утверждений.

Если указанный подслучай не имеет места, то - переход через "ветвь 2".

Переменной x5 присваивается список программных выражений, определяющих фрагменты списка переменных связывающей приставки результирующего квантора. Здесь учитывается возможность выбора новых переменных, в том числе при наличии указателя приема "связприставка". После составления списка x5 - переход через "иначе 1".

Расположенные подряд в списке x5 выражения "набор(...)" склеиваются, и переменной x6 присваивается итоговое программное выражение для связывающей приставки.

В случае квантора существования либо описателя "класс" находится программное выражение x7 для подкванторного утверждения, и определяется результирующее программное выражение x8.

В случае квантора общности сначала рассматривается ситуация, возникающая при компиляции приемов вывода теорем. Здесь консеквент x1 имеет вид функциональной переменной $P(x)$, выделенной указателем "импликация". Часть антецедентов результирующего квантора общности здесь заданы непосредственно, а часть -

размещена внутри вспомогательного терма "то(...)" для $P(x)$. Чтобы получить объединенный список антецедентов, используется процедура "новпосылки". После этого рассматривается общий случай, в котором результат создается с помощью процедуры "Импликация".

Наконец, в случае описателя "отображение" находятся программные выражения x7, x8 для утверждения, задающего область определения, а также выражения, задающего значение функции. Затем строится результирующее выражение x9.

Суффикс либо префикс

Если x1 имеет вид "суффикс(a b)" либо "префикс(b a)", причем удается найти программное выражение x6 для набора термов a, то находится программное выражение x7 для терма b, и строится результирующее выражение для терма вида "набор(...)" (контрольная точка "прием(20)").

Числитель либо знаменатель

Если x1 имеет заголовок "числитель" либо "знаменатель", причем имеется одноименный информационный элемент для x1, то из него извлекается ранее найденное программное выражние для численного значения n числителя либо знаменателя x1. Оно преобразуется в формат терма (контрольная точка "прием(21)").

19.12.5 Общий случай компиляции неоднобуквенного терма

После контрольной точки "прием(22)" располагается ветвь, обрабатывающая общий случай компиляции неоднобуквенного терма x1 (как и в предыдущем разделе, имеется в виду компиляция без учета нормализаторов). Предварительно инициализируется нулем накопитель результата x4.

Переменной x5 присваивается набор программных выражений для корневых операндов терма x1. Эти программные выражения получены с помощью рекурсивных обращений к процедуре "метаперевод". В тех случаях, когда выражение имеет вид "набор(s)", где s - логический символ, оно заменяется на однобуквенный терм s. Происходит коррекция обозначений связанных переменных в выражениях списка x5 - чтобы их номера были не менее номера первой не используемой программной переменной. Затем - переход через "ветвь 3".

Далее предпринимается цепь коррекций списка x5 с учетом информационных элементов программного блока.

Информационный элемент "учетзнака"

После контрольной точки "прием(23)" рассматривается случай, когда x1 - заменяющий терм приема, компилируемого с указателем "знаксуммы", причем идентификация происходит для части операндов преобразуемого терма. В этой ситуации при компиляции была введена переменная - модификатор, идентифицируемая с неизменяемым остатком операндов. Элементы списка x5, соответствующие изменяемым операндам, обрабатываются процедурой "измзнака", обеспечивающей необходимую коррекцию знака.

После контрольной точки "прием(30)" рассматривается аналогичный случай - но при отсутствии модификатора и отсутствии применяемых к x1 нормализаторов.

Здесь корректируется уже не набор x5, а ранее сформированная с помощью x5 заготовка результата x4.

Ввод обозначений для больших подвыражений

После контрольной точки "прием(24)" анализируется суммарная длина программных выражений списка x5. Если она чрезмерно велика, то для всех достаточно длинных выражений списка x5 вводятся вспомогательные переменные. Присвоение значения такой переменной регистрируется в компилируемой программе; номера входящих в оставшиеся выражения списка x5 связанных переменных сдвигаются так, чтобы они были больше номера новой переменной.

Перестановки операндов

Если имеется информационный элемент (сдвиг ...), указывающий, что операнды терма x1 должны быть подвергнуты циклической перестановке, такой же, как некоторая предпринятая при идентификации, то результат x4 строится с помощью обращения к процедуре "сдвигнабора".

Аналогично, если имеется информационный элемент (модифсборка ...), указывающий, что операнды терма x1 должны быть переупорядочены так же, как при идентификации переупорядочивались элементы некоторого набора (контрольная точка "прием(50)"), то результат строится с помощью процедуры "модифсборка".

Далее с помощью набора x5 программных выражений для корневых операндов терма x1 строится выражение "запись(...)" для самого терма x1. Оно переприсваивается переменной x4.

Информационные элементы "дробь" и "альтернатива"

После контрольной точки "прием(25)" рассматривается случай наличия информационного элемента (дробь ...), указывающего на необходимость перестановки корневых операндов терма x1, если при идентификации имела место перестановка корневых операндов некоторой другой двуместной операции. Тогда результирующее выражение x4 - условное, проверяющее наличие перестановки при идентификации. Перед формированием его проверяется также наличие информационного элемента (альтернатива ...), требующего изменения заголовка терма x1, если при идентификации имело место изменение заголовка некоторого другого терма. Соответственно корректируется накопитель заголовка результата x7.

После контрольной точки "прием(26)" аналогичная коррекция выполняется, если терм x1 выделен только указателем "альтернатива".

Информационный элемент "единица"

После контрольной точки "прием(28)" рассматривается случай, когда x1 имеет два корневых операнда, причем один из них, согласно информационному элементу (единица ...), может обратиться в единицу корневой операции терма x1. Здесь результирующее выражение x4 строится так, чтобы не нужный единичный операнд отбрасывался.

После перечисленных коррекций - откат к переходу "ветвь 1", расположенному непосредственно перед контрольной точкой "прием(22)". Далее - цепочка преобразований, оптимизирующих результат компиляции x4.

Оптимизация результата компиляции

После контрольной точки "прием(31)" начинается цикл упрощающих тождественных преобразований программного выражения x4. Здесь рассматриваются следующие случаи (перечисление их ведется в стандартном порядке - для каждого фрагмента программы переходы просматриваются от конца к началу):

1. x4 имеет вид "запись(f унисборка(f t) p)", где f - ассоциативно - коммутативный символ, причем t - набор f - членов, идентифицированный с переменной - модификатором (т.е. с набором остаточных операндов, не участвующих в замене). Тогда x4 заменяется на "унисборка(f суффикс(t p))".

   Символ f предполагается далее ассоциативным и коммутативным.

2. x4 имеет вид "запись(f унисборка(f вычеркивание(набороперандов(A)набор(подтерм(B_1) ... подтерм(B_n)))$C_1 \dots C_m$))". Каждое B_i - программное выражение для операнда вхождения, задаваемого программным выражением A. Здесь рассматриваются следующие подслучаи:

   (a) $C_1, \dots, C_m$ - подмножество списка "подтерм(B_1)", ... , "подтерм(B_n)", причем разность этих списков состоит из единственного выражения "подтерм(D)". Тогда x4 заменяется на "исключениеоперанда(A D)".

   (b) $m = 1$, $n = 2$. Тогда x4 заменяется на "склейкаоперандов(A B_1 B_2 C_1)".

   (c) $m = n$; единственное выражение C_i, не равное никакому "подтерм(B_j)", есть C, а единственное выражение "подтерм(B_j)", не равное никакому C_i, есть "подтерм(B)". Тогда x4 заменяется на "заменаоперанда(A B C)".

3. x4 имеет вид "запись(f унисборка(f вычеркивание(набороперандов(A)набор(B_1 ... B_n)))$C_1 \dots C_m$)". Оно заменяется на "унисборка(f конкатенация (набор(C_1 ... C_m)вычеркивание (набороперандов(A)набор($B_1 \dots B_n$))))".

4. x4 имеет вид "запись(f унисборка(f t)p)" либо вид "запись(f p унисборка(f t))". Соответственно, оно заменяется на "унисборка(f суффикс(t p))" либо "унисборка(f префикс(p t))".

5. x4 имеет вид "запись(f унисборка(f t)$p_1 \dots p_n$)"; $n > 1$. Оно заменяется на "сборка(f конкатенация(t набор($p_1 \dots p_n$)))".

6. x4 имеет вид "запись(f $A_1 \dots A_n$)", где хотя бы один из операндов A_i имеет вид "унисборка(t)". Находится список $t_1, \dots, t_m$ всех таких t, а также набор $p_1, \dots, p_k$ остальных операндов A_i. x4 заменяется на "унисборка(f конкатенация($t_1 \dots t_m$ набор($p_1 \dots p_k$)))".

7. x4 имеет вид "сборка(f конкатенация($A_1 \dots A_{i-1}$ набор($B_1 \dots B_{j-1}$ унисборка(f C) $B_{j+1} \dots B_m$) $A_{i+1} \dots A_n$))". Тогда C выносится в список операндов A_k, т.е. x4 заменяется на "сборка(f конкатенация($A_1 \dots A_{i-1}$ набор($B_1 \dots B_{j-1} B_{j+1} \dots B_m$) $A_{i+1} \dots A_n$ C))". Учитываются вырожденные случаи - возможность $m = 1$ и возможность получения одноэлементной конкатенации; в последнем случае вместо заголовка "сборка" используется "унисборка".

8. В x4 имеется вхождение подтерма "конкатенация($A_1 \dots A_n$)", являющееся операндом для символа "сборка" либо "унисборка". Некоторое A_i имеет вид "вычеркивание(B A_j)". Тогда A_i, A_j заменяются на B.

Упрощающие преобразования других типов будут применяться к операторам компилируемой программы при завершающем редактировании.

19.13 Предварительная обработка установок на нормализацию

При компиляции теоремных термов используются информационные элементы (быстрпреобр ...), заблаговременно создаваемые процедурой "учетнормализаторов(x1 x2 x3 x4)". Входными данными здесь служат: x1 - теорема приема; x2 - заголовок приема; x3 - программный блок; x4 - описание приема. Обращения к процедуре "учетнормализаторов" разбросаны по различным фрагментам компилятора так, чтобы по мере идентификации переменных происходило своевременное пополнение списка элементов (быстрпреобр ...). Отсутствие этих элементов на момент компиляции терма приведет к игнорированию его нормализаторов.

Если среди нормализаторов приема имеется хотя бы один, использующий список посылок, причем сам прием не относится к пакетному нормализатору, то проверяется наличие информационного элемента (проверка x). Этот элемент ссылается на программную переменную x, значением которой является пара (выводимо A); A - накопитель использованных приемом посылок. Если такого элемента нет, то в компилируемую программу заносится оператор, инициализирующий накопитель (выводимо ...), и указанный элемент создается. Одновременно инициализируется накопитель (коррекцияпосылок A) и вводится связанный с ним информационный элемент (коррекцияпосылок ...). Ссылки на накопители использованных посылок будут передаваться компилятором в комментарии нормализаторов. Наличие в комментариях элемента (выводимо ...) является для многих нормализаторов необходимым условием выполнения ими своих преобразований.

Инициируется нулем индикатор отказа x5. Если ему будет присвоена единица, то произойдет отказ от компиляции приема.

Определяющие группы нормализаторов приема термы "быстрпреобр(...)" упорядочиваются по возрастанию длин, после чего начинается цикл их просмотра. x7 - текущий просматриваемый терм "быстрпреобр(...)". Переменной x8 присваивается обрабатываемый нормализаторами терм, а переменной x9 - список применяемых к нему нормализаторов приема, причем в случае пакетного нормализатора берется только его заголовок. Переменной x10 присваивается список той же длины, что и x9, образованный наборами указателей обращения к пакетным нормализаторам. Для обращений к вспомогательным задачам в списке x10 помещаются пустые наборы.

Просматриваются нормализаторы "посылки(...)" и "удалениепосылок(...)", приведенные в терме x7. По ним создаются информационные элементы (посылки ...) и (чисткапосылок ...), уточняющие список посылок, относительно которого к терму x8 будут применяться нормализаторы. Затем - переход через "ветвь 2".

Здесь проверяется отсутствие ранее созданного для обработки терма x8 информационного элемента (быстрпреобр ...). Затем инициируются накопители x11 и x12 разрядов A и B создаваемого информационного элемента (быстрпреобр x8 x9 A B

C). Напомним, что A - список наборов программных выражений для входных данных нормализаторов, за исключением первого входного данного (т.е. выражения для нормализуемого терма); B - список наборов сопровождающих обращение к нормализатору указателей "условие(...)", "вариант(...)", "лимит(...)"; C - изначально 0, а затем программное выражение для результата обработки терма x8 нормализаторами. Последнее будет определяться уже на этапе обращения к процедуре "метаперевод".

Для заполнения накопителей x11 и x12 реализуется цикл синхронного просмотра списков x9 и x11. В этом цикле x13 - текущая позиция списка x9, x14 - текущая позиция списка x11. Если на позиции x13 расположено название пакетного нормализатора, то переменной x15 присваивается набор, задающий его формат; в случае задачи берется набор ("списокпосылок", "коррекцияпосылок"), а в случае переменной для варьируемого названия пакетного нормализатора - набор ("списокпосылок").

Если x15 оказалось равно 0, причем компилируется прием вывода теорем, а на позиции x13 расположено название более чем одноместного пакетного нормализатора, отличного от символов "станд", "стандупорядочение", то второе и третье входные данные обращения суть: список комментариев и блок вывода "икс(1)". Они заносятся на позицию x14, и продолжение цикла просмотра списков x9, x11.

Иначе - находится список x16 указателей обращения к нормализатору для текущей позиции x13. Из него извлекаются, если они там есть, элементы "лимит(...)", "фильтр(...)", "условие(...)", "вариант(...)", и регистрируются на текущей позиции накопителя x12. В последнем случае рассматривается альтернативный пакетный нормализатор x18, и список x15 пополняется элементами набора, задающего формат этого нормализатора. Далее программа разветвляется.

19.13.1 Компиляция приема пакетного нормализатора

Если компилируется прием пакетного нормализатора, то находятся заголовок нормализатора x16 и набор x17, определяющий его формат.

Если нормализатор по вхождению x13 не имеет списка посылок, то проверяется, имеет ли он хотя бы список комментариев. В последнем случае на позицию x14 заносится одноэлементный набор, состоящий из программного выражения для списка комментариев нормализатора x16. Иначе - на позиции x14 сохраняется "пустоеслово".

Если же нормализатор по вхождению x13 имеет список посылок, то инициируется накопитель x18 программного выражения для списка комментариев обращения к нормализатору. Изначально в нем содержится ссылка на список комментариев того нормализатора, чей прием компилируется. Далее выполняются следующие коррекции x18:

1. В x15 содержится указатель формата "разборслучаев". Если нормализатор x16 имеет тот же заголовок, что и текущий нормализатор, то список комментариев x18 корректируется так, чтобы в нем оказался элемент "подчинено". Это позволит различать корневое обращение к нормализатору, выполняющему разбор случаев, и вспомогательные рекурсивные обращения, использующие для разбора случаев структуру данных, созданную при корневом обращении. Если же заголовок x16 отличается от заголовка текущего нормализатора, то в компилируемую программу заносится оператор, создающий структуру данных для разбора случаев. Эта структура данных передается в список комментариев x18, откуда процедурой "началоразбора" предварительно исключаются структуры данных, относящиеся к внешнему разбору случаев.

2. Текущий нормализатор имеет указатель обращения "исключкоммент(...)", определяющий отбрасывание всех начинающихся с заданных символов комментариев того нормализатора, чей прием реализуется. Тогда к x18 применяется процедура "исключкоммент", выполняющая данную расчистку.

3. Текущий нормализатор имеет указатель обращения одного из видов: "замечание(...)", "примечание(...)", "удалениезамечания(...)". Этот указатель явно задает присоединяемый либо удаляемый комментарий. Находится программное выражение для комментария, и выполняется соответствующая коррекция x18.

4. Если компилируемый прием имеет указатель (вывод ...), а также если нормализатор, к которому он относится, не является корневым и предусматривает обработку комментария (коррекцияпосылок ...), - предпринимается удаление из x18 ссылки на "старый" накопитель (выводимо ...) и вводится ссылка на новый такой накопитель. Для этого применяется процедура "переучет".

Далее вводится накопитель x19 программного выражения для списка посылок, используемого при обращении. Изначально берется ссылка на список посылок реализуемого нормализатора. Для случая нормализатора "нормили" список посылок пополняется утверждениями из контекста преобразуемого терма.

Наконец, на позицию x14 заносится пара (x19,x18) выражений для списков посылок и комментариев обращения к текущему нормализатору x13.

19.13.2 Компиляция приема, не относящегося к пакетному нормализатору

В этом случае прежде всего определяется программное выражение x16 для списка посылок обращения (контрольная точка "прием(2)"). Для этого применяется процедура "посылки". Если имелся указатель обращения "расширениепосылок", то x16 корректируется с помощью одноименной процедуры - добавляются утверждения, выводимость которых из основных посылок устанавливается по комментарию (коррекцияпосылок ...). Затем x16 регистрируется в конце накопителя входных программных выражений, размещенного на текущей позиции x14.

После отката к переходу через "ветвь 1"переменная x16 снова становится не определенной, а затем инициируется пустым словом уже в качестве накопителя программных выражений для наборов комментариев компилируемого обращения к нормализатору.

В случае обращения к нормализатору из приема пакетного анализатора происходит переприсвоение x16 выражения для списка комментариев анализатора, пополненного элементом (выводимо пустоеслово). В прочих случаях набор комментариев полагается состоящим из элемента (выводимо пустоеслово), а при наличии указателя "коррекцияпосылок" формата текущего нормализатора - также элемента (коррекцияпосылок пустоеслово).

Если компилируемый прием имел указатель "удалениеусловия(...)" либо "удалениепосылки(...)", в котором оговаривалась блокировка удаления утверждения, использованного как посылка при нормализации заданных термов, то создается либо корректируется информационный элемент (норм ...). В нем перечисляются накопители (выводимо ...), появление в которых удаляемого утверждения означает отмену удаления.

Если имеется указатель "неизвестные" формата текущего нормализатора, то в x16 заносится программное выражение для набора, либо состоящего из цели текущей задачи (неизвестные ...), либо, если такой цели нет, пустого. При наличии указателя формата "разборслучаев" в x16 включается программное выражение для комментария (разборслучаев ...), организующего разбор случаев текущим нормализатором.

Далее x16 пополняется комментариями, определяемыми указателями обращения к нормализатору "замечание(...)", "комментарии(...)".

При обращении к нормализатору с переменным заголовком из приемов вывода теоремы происходит регистрация комментария "редуцирование".

Наконец, набор программных выражений для входных данных текущего обращения пополняется термом "унисборка(конкатенация x16)", собирающим список комментариев из перечисленных выше фрагментов.

19.14 Компиляция фильтров для приема сканирования задачи

При компиляции приема сканирования задачи обработка фильтров производится процедурой "блокпроверок(x1 x2 x3 x4 x5)". Ее входные данные таковы: x1 - теорема приема; x2 - заголовок приема; x3 - описание приема; x4 - программный блок; x5 - узел приема. Процедура обрабатывает все те фильтры приема, которые не были учтены на предшествующих этапах компиляции, и вставляет их в компилируемую программу так, чтобы обеспечить возможно лучшее отсечение бесполезных действий при сканировании задачи. Выйти на начальную точку программы "блокпроверок" можно через пункт оглавления программ "Формирование фильтров приема, активизируемого при сканировании задачи".

19.14.1 Учет уровней срабатывания приема

Прежде всего, создается рабочая копия x6 описания приема, которая будет изменяться по мере обработки фильтров. Далее переменной x7 присваивается вхождение первого фрагмента программы текущего программного блока x4, а переменной x8 - терм "условие(и($F_1 \ldots F_n$))" из описания приема, перечисляющий фильтры $F_1, \ldots, F_n$. Инициализируется нулем заготовка x9 оператора, определяющего уровни срабатывания приема. После контрольной точки "прием(3)"вводится пустой накопитель x10 этих уровней. Он заполняется всеми такими значениями n, которые встречаются в подтермах фильтров, имеющих вид "уровень(... n ...)". При этом неважно, является ли подтерм корневым или расположен внутри фильтра. Значения n упорядочиваются по возрастанию, и по завершении просмотра (переход "иначе 5") переменной x9 переприсваивается оператор "уровень(...)". Затем - откат к переходу через "ветвь 4". Если список x10 был пуст и x9 осталось равно 0, то компиляция приема отменяется.

19.14.2 Компиляция префиксных фильтров приема

После контрольной точки "прием(4)" происходит инициализация списка x10 фильтров, заносимых в начало программы приема (они называются префиксными фильтрами). В список x10 заносятся: оператор "стандарт(x2)", проверяющий, что после вхождения текущего сканируемого символа идет открывающая скобка (кроме случаев, когда такой оператор заведомо не нужен); присвоенный переменной x9 опера-

тор "уровень(...)"; оператор "новый", блокирующий повторное рассмотрение символа при ненулевом отложенном уровне сканирования. Далее составляется список x11 всех фильтров приема, из которого вычеркиваются фильтры, учтенные на предшествующих этапах компиляции - они зарегистрированы в информационных элементах (комментарии ...). Предпринимается просмотр всех фильтров списка x11, не содержащих символа "результат" и не имеющих заголовка "уровень". Первые будут обрабатываться после формирования приемом результирующего (например, заменяющего) терма; второй - уже учтен при создании оператора x9.

Для текущего фильтра предпринимается цикл разборки его логической конструкции на атомарные составляющие, из которых собирается набор x13. Если в некоторый момент выясняется, что атомарная составляющая не подлежит включению ее в префиксный фильтр (здесь применяется справочник "начало"), то переход к очередному элементу списка x11. Специально блокируется обработка фильтра "цель(...)" до обработки фильтра "тип(...)". Проверка наличия той или иной цели до выяснения типа задачи может привести к обрыву решения по некорректному обращению к оператору ЛОСа - так как задачи на доказательство целей не имеют.

По завершении цикла разборки текущего фильтра, целью которого является принятие решения о занесении его в префиксные фильтры, выполняется обращение к процедуре "фильтр" и получение программного выражения x15 для конъюнкции операторов, проверяющих истинность фильтра. Конъюнктивные члены x15 добавляются к концу списка x10.

По мере обработки префиксных фильтров их позиции в списке x11 заменяются нулями. Затем терм "условие(...)" в копии x6 описания приема заменяется на скорректированный, в котором исключены уже обработанные фильтры. Наконец, список операторов x10 вставляется в начало первого фрагмента скомпилированной программы, после первого оператора этого фрагмента.

19.14.3 Определение результирующего терма

В основном, компиляция фильтров, анализирующих результирующий терм приема (заменяющий терм, выводимое утверждение и т.п.), выполняется в рамках процедуры "преобразователь", компилирующей действия по преобразованию задачи. Однако, в двух специальных случаях этот терм находится уже при обработке фильтров.

Прием замены, имеющий фильтр "длина"

Если компилируется прием замены, причем некоторый фильтр имеет вхождение символа "длина", рассматриваемое как указание на сравнение старого и нового термов (контрольная точка "прием(5)"), то находится заменяющий терм x8 теоремы приема и определяется программное выражение x9 для этого терма. Оно регистрируется для дальнейшего использования в информационном элементе (результат ...).

Прием вывода, имеющий фильтр "контрольвывода"

Если компилируется прием вывода, некоторый фильтр которого содержит символ "контрольвывода", то должна выполняться проверка того, что выводимое утверждение не содержит выражений, ранее отстуствовавших в задаче (контрольная точка "прием(6)"). Здесь определяется программное выражение x7 для выводимого утверждения, регистрируемое в информационном элементе (результат ...).

19.14.4 Основной цикл обработки фильтров

После контрольной точки "прием(7)" происходит обработка оставшихся фильтров, кроме тех, которые содержат ссылку "результат" на новый терм, вводимый приемом. Переменной x7 сначала присваивается терм "условие(...)" из копии x6 описания приема; затем находится терм x9, полученный удалением из x7 уже обработанных либо содержащих "результат" фильтров, и переменной x7 переприсваивается значение x9.

Для тех теоремных переменных, которые расположены внутри подтермов фильтров, имеющих вид "программа(...)", рассматриваются программные выражения T, определяющие их идентификацию. Если встречается T, отличное от переменной, то для него вводится вспомогательная программная переменная.

Находится список x8 фильтров, задаваемых термом x7. Его элементы переупорядочиваются так, чтобы фильтры с заголовком "конец" размещались в конце.

Последовательно просматриваются фильтры x9 списка x8. Из просмотра исключаются фильтр "контрольодз", предполагающий анализ результирующего терма приема, а также технические разделители ",". Для обработки текущего фильтра привлекается процедура "фильтр" (она будет описана далее). Определяемая ею конъюнкция операторов, реализующих фильтр, разбирается на отдельные конъюнктивные члены, и каждый из них регистрируется в компилируемой программе с помощью процедуры "вставкафильтра". Эта процедура и принимает решение о точке размещения фильтра в компилируемой программе. Краткое описание ее будет приведено далее.

19.14.5 Дополнительные фильтры для приема "связка"

После контрольной точки "прием(8)" рассматривается случай приемов, имеющих заголовок "связка" либо "существует". Такие приемы выполняют устранение группы X несущественных неизвестных задачи, усматривая либо возможность замены утверждения существования значений X, при которых истинны все содержащие их условия задачи, на некоторое новое условие, либо истинность данного утверждения существования. Переменной x8 присваивается список программных выражений для переменных списка X. Затем добавляются операторы, проверяющие следующие условия: все переменные списка X суть несущественные неизвестные; они различны; не участвующие в групповой замене условия задачи не содержат переменных X.

19.15 Процедура "фильтр"

Для компиляции отдельного фильтра введена процедура "фильтр(x1 x2 x3 x4 x5)". Это - операторное выражение ЛОСа, которому передаются следующие входные данные: x1 - теорема приема; x2 - заголовок приема; x3 - описание приема; x4 - программный блок; x5 - компилируемый подтерм фильтра. Значением выражения становится оператор, реализующий проверку истинности подтерма x5. Если такой оператор создать не удалось, выдается значение 0.

Всю фактическую работу по компиляции фильтра выполняют справочники "блокпроверок" и "оператор", к которым обращается процедура "фильтр". Логическим символом обращения является заголовок подтерма x5.

Программа справочника "блокпроверок" получает при обращении к ней те же входные данные x1 - x5, что и процедура "фильтр". Она выдает в качестве результата оператор для проверки фильтра x5. Возможны особые случаи, когда выдается

операторное выражение, используемое во внешнем операторе фильтра, либо даже вспомогательный программный блок (см. далее раздел, посвященный фильтрам с заголовком "контекст"). Обычно процедура справочника "блокпроверок" предпринимает сначала обращения к процедуре "прогрвыражение" для получения операторных выражений, задающих входные данные проверки, а затем строит с их помощью проверочный оператор.

Чтобы упростить подключение к компилятору новых типов фильтров, был введен справочник "оператор". Он используется в тех случаях, когда заголовок S подтерма x5 одновременно является заголовком выполняющей проверку процедуры ЛОСа. Справочник получает в качестве входного данного лишь символ обращения S и выдает набор указателей типов значения операндов процедуры S. По этому набору типов процедура "фильтр" сама организует необходимые обращения к оператору "прогрвыражение" и создает результирующий оператор $S(\ldots)$. Обращение к справочнику "оператор" происходит из процедуры "фильтр", если обращение к справочнику "блокпроверок" оказалось безрезультатным.

Программы справочников "блокпроверок" и "оператор" практически все несложны. Исключение здесь составляет лишь весьма большая программа справочника "блокпроверок" на символе "контекст". Ей будет посвящен далее специальный раздел.

Если фильтр x5 имеет вид "определено(P $A_1 \ldots A_n$)", то выполняющий проверку условия P оператор должен размещаться в программе после того, как будут идентифицированы все переменные термов $A_1, \ldots, A_n$. Процедура "фильтр" находит оператор x6 для проверки P; составляет список x8 программных выражений для $A_1, \ldots, A_n$, находит список x9 параметров этих выражений, и выдает в качестве результата вспомогательный терм "фикс(x6 x9)". Наличие в нем переменных списка x9 приведет к тому, что процедура "вставкафильтра" начнет выбор позиции для фильтра в программе после точки, в которой все эти переменные определятся. При этом заголовок "фикс" и переменные x9 будут отброшены, а вставлен в программу только сам оператор x6.

19.16 Процедура "вставкафильтра"

Обращение к процедуре имеет вид "вставкафильтра(x1 x2)", где x1 - программный блок; x2 - вставляемый оператор фильтра. Предполагается, что он не имеет выходных переменных.

Если x2 имеет заголовок "истина", то никаких действий не предпринимается. Если x2 имеет вид "конец(A)", то оператор фильтра A заносится в конец компилируемой программы.

В остальных случаях определяется такая программная переменная x3, что фильтр должен быть расположен в программе не раньше места, где значение этой переменной становится определенной. Сначала x3 присваивается переменная с номером 6 - первая не определенная при сканировании задачи. Если имеется информационный элемент (вставкафильтра X), причем номер переменной X больше номера значения переменной x3, то x3 переприсваивается X. Для пакетного оператора в начале программы ищется "точка разветвления", до которой располагается общая часть всех приемов пакета (например, у нормализаторов это оператор "повторение"). Если номер переменной - значения x3 не превосходит номеров переменных, встречающихся до точки разветвления, то значение x3 корректируется. Наконец, проверяется, не со-

держит ли вставляемый фильтр свободной переменной, номер которой не меньше номера значения x3. Тогда выполняется еще одна коррекция x3.

Если фильтр имеет вид "фикс($A\ B_1 \ldots B_n$)", то он заменяется на A. Указание на размещение этого фильтра после того, как определятся все свободные переменные термов $B_1, \ldots, B_n$, уже учтено при выборе x3.

Далее начинается цикл просмотра операторов программы для выбора места вставки фильтра. Фрагменты программы просматриваются в том порядке, как они расположены в программном блоке; внутри каждого фрагмента операторы перечисляются слева направо. x4 - вхождение текущего фрагмента x5 в список фрагментов; x6 - вхождение текущего оператора в фрагмент x5. Вставка будет происходить перед оператором x6. Операторы "меткаперехода" и "Обрыв" игнорируются. При обнаружении оператора перехода просмотр фрагмента обрывается. Для текущего оператора P проверяется выполнение хотя бы одного из условий:

1. Имеется свободная переменная P, номер которой не меньше номера переменной x3.

2. Фильтр имеет вид "десчисло(A)", x6 - последнее вхождение в последний фрагмент, причем все свободные переменные терма A встречались в предшествующих операторах.

3. Фильтр имеет единственную переменную x, являющуюся свободной переменной предшествующего x6 оператора, причем в этом фильтре используются только символы отношений и операций "равно", "заголовок", "входит", "не", "унисборка", "параметры", "списокпеременных", "символ".

4. Фильтр имеет вид "равно(суммавсех(... операнд(...) ...)A)", причем все его свободные переменные встречались в предшествующих операторах.

5. Фильтр имеет вид "не(равно($x\ y$))", причем оператор, предшествующий x6, имеет переменные с номерами, не меньшими номеров переменных x, y.

Если ни одно из них не выполнено, то переход к очередному вхождению x6. Иначе - продолжается анализ текущего вхождения с точки зрения целесообразности вставки перед ним фильтра x2. Заметим, что приведенные выше условия, начиная со второго, выделяли особые случаи "простых" фильтров, размещаемых возможно ближе к началу.

Далее идет цепь проверок различных условий, каждое из которых означает нецелесообразность вставки фильтра перед текущей позицией x6. Выполнение любого из них влечет откат для перехода к следующей текущей позиции x6. В особых случаях вместо этого предпринимается сдвиг текущей позиции влево и продолжение обработки цепочки условий. Перечислим эти условия:

1. Фильтр содержит подтерм "числзначение(A)", причем после x6 расположен оператор "десчисло(A)". Очевидно, попытка получения десятичного числа по выражению A до проверки того, что это выражение является десятичной записью числа, нежелательна.

2. После x6 имеется вхождение оператора "Текпосылка" либо "текпосылка". Эти операторы применяются в пакетных анализаторах для перечисления текущих вхождений, и отрезок до них обычно относится сразу ко многим приемам.

3. После x6 имеется оператор "символ(x a)", где a - логический символ либо символ переменной, причем фильтр содержит подтерм "операнд(x t)". Отсечение по заголовку подтерма x дешевле, чем проверка условия, затрагивающего операнды этого подтерма.

4. После x6 имеется оператор "равныетермы(...)", причем в фильтре используется подтерм "десчисло(...)" или "констдробь(...)".

5. После x6 имеется оператор "новый".

6. После x6 имеется оператор "повторение", либо "контрольбуфера", либо дизъюнкция "или($A_1 \dots A_n$)", у которой A_1 - оператор присвоения, а сразу вслед за дизъюнкцией расположено некоторое A_i. Все эти случаи относятся к размещению "точки разветвления" пакетного оператора после текущей позиции x6.

7. Либо существует общая свободная переменная оператора фильтра и текущего оператора, не определенная до реализации этого оператора и имеющая номер более 5, либо существует такая свободная переменная фильтра, которая изменяется идущим после x6 оператором "замена(...)". В последнем случае исключение делается для изменения переменной x2 приемом пакетного нормализатора (коррекция списка посылок после срабатывания приема). В обоих случаях номер переменной x3 увеличивается так, чтобы он был больше номеров переменных текущего оператора.

8. Фильтр имеет операторное выражение, выделяющее корневой операнд либо подтерм с заданным номером от начала или от конца - например, "второйоперанд(x)", причем после x6 располагается оператор "символ(x ...)" или "заголовок(x ...)". Без уточнения того, какой символ расположен на вхождении x, нельзя гарантировать существование операнда с указанным в фильтре номером.

9. Если фильтр имеет вид "десчисло(A)", причем некоторый предшествующий текущему оператору оператор x8 имеет вхождение подтерма "числзначение(A)" либо "числзначение(левыйкрай(A))", то предпринимается откат текущей позиции x6 к позиции оператора x8. После этого - не возвращение к циклу просмотра вхождений x6, а переход через "ветвь 1" для продолжения обработки цепочки условий.

10. Фильтр имеет вхождение подтерма "цель(x1 ...)", причем некоторый расположенный после текущего (включая его) оператор имеет вхождение подтерма "тип(x1 ...)", а предшествующие текущему операторы не имели таких подтермов. Здесь x1 - переменная, обозначающая текущую задачу приема. До установления ее типа нельзя обращаться к проверке наличия той или иной цели, т.к. задача на доказательство целей не имеет, и интерпретатор ЛОСа в указанном случае будет выдавать сообщение о некорректном использовании оператора.

11. Если оператор, предшествующий текущему, имеет заголовок "уровеньобращения", то происходит откат текущей позиции x6 к предыдущей - чтобы фильтр не портил установки на уровень обращения к вспомогательной задаче. Далее - продолжение обработки цепочки условий.

12. Если фильтр имеет вхождение подтерма "унисборка(A t)", причем некоторый расположенный после х6 оператор P проверяет непустоту списка t, то он выбирается в качестве текущего, и далее - продолжение обработки цепочки условий. Если P был расположен в конце программы, то фильтр добавляется после него.

После прохождения через цепочку проверок решение о вставке фильтра перед позицией х6 считается принятым. Далее просматриваются условные выражения "вариант(A t_1 t_2)", расположенные после х6, у которых A совпадает с фильтром или его отрицанием. Такие условные выражения заменяются на соответствующий подтерм t_1 либо t_2.

Чтобы связанные переменные фильтра были отличны от ранее определенных переменных, предпринимается его коррекция с помощью процедуры "нормоператора". Проверяется отсутствие уже имеющегося ранее позиции х6 оператора, совпадающего с оператором фильтра. Затем предпринимается вставка фильтра непосредственно перед х6.

19.17 Компиляция фильтров и операторных выражений со связанными переменными

Как уже отмечалось выше, в языке фильтров ГЕНОЛОГа кванторы и описатели вводятся без явного указания связанных переменных.

Например, квантор существования оформляется в виде фильтра "контекст($A_1 \ldots A_n$)", у которого связанными считаются все переменные, не идентифицированные вне этого фильтра. Строго говоря, данная конструкция представляет собой нечто большее, чем просто квантор существования - она может содержать в списке $A_1, \ldots, A_n$ указатели идентификации и другие технические элементы, доопределяющие семантику фильтра и способ его компиляции. Специального аналога квантора общности для фильтров нет - он выражается через отрицание квантора существования.

Кроме фильтра "контекст", принцип введения связанных переменных "по умолчанию" применяется и для встречающихся в фильтрах операторных выражений. Перечислим основные типы таких выражений:

1. "число($A_1 \ldots A_n$)" - количество ситуаций, удовлетворяющих условиям $A_1, \ldots, A_n$;

2. "перечисление($A_1 \ldots A_n$ t)" - набор значений операторного выражения t для ситуаций, в которых истинны $A_1, \ldots, A_n$, с исключением повторных вхождений;

3. "выписка($A_1 \ldots A_n$ t)" - то же, что предыдущее, но без исключения повторных вхождений;

4. "суммавсех($A_1 \ldots A_n$ t)" сумма числовых значений операторного выражения t для ситуаций, в которых истинны $A_1, \ldots, A_n$;

5. "нод($A_1 \ldots A_n$ t)" - наибольший общий делитель значений операторного выражения t для ситуаций, в которых истинны $A_1, \ldots, A_n$;

6. "нок($A_1 \ldots A_n$ t)" - наименьшее общее кратное значений операторного выражения t для ситуаций, в которых истинны $A_1, \ldots, A_n$;

7. "максимум($A_1 \dots A_n$ t)" - наибольшее значение операторного выражения t для ситуаций, в которых истинны $A_1, \dots, A_n$;

8. "минимум($A_1 \dots A_n$ t)" - наименьшее значение операторного выражения t для ситуаций, в которых истинны $A_1, \dots, A_n$;

Здесь, как и выше, некоторые из термов A_i могут представлять собой указатели. Во всех перечисленных случаях для компиляции применяется программа справочника "блокпроверок" на символе "контекст". Если заголовок операторного выражения либо фильтра, имеющих связанные переменные, отличен от символа "контекст", то происходит переадресация обращения программе символа "контекст". Так как заголовок компилируемого терма при такой переадресации сохраняется, то программа "контекст" сможет впоследствии учесть специфику конкретного обращения к ней. Найти начальную точку данной программы можно через раздел "Формирование фильтров "контекст(...)" " оглавления программ.

В начале программы определены следующие значения: x1 - теорема приема; x2 - заголовок приема; x3 - описание приема; x4 - программный блок; x5 - компилируемый терм $F(A_1 \dots A_n)$.

Переменной x6 присваивается набор корневых операндов $A_1, \dots, A_n$. Если компилируется программное выражение, то из данного набора исключается последний элемент A_n - он играет особую роль и будет учитываться впоследствии.

Создаются списки x7 и x8, несущие информацию об идентифицирующих термах A_i, имеющих, соответственно, вид "вид(t_1 t_2)" либо "подтерм(t)". x7 состоит из пар термов (t_1, t_2); x8 - из термов t.

Переменной x9 присваивается список заголовков тех информационных элементов, которые будут передаваться из текущего программного блока x4 во вспомогательный программный блок x10, создаваемый для компиляции терма x5. При компиляции программного выражения будет создаваться оператор, реализующий цикл обработки накопителя (текущей суммы, текущего списка, текущего максимума или минимума и т.п.). В этих случаях первая неопределенная программная переменная блока x10 берется на единицу большей - чтобы пропустить переменную, выбранную в качестве накопителя. Кроме элементов с заголовками из x9, в блок x10 переносятся все элементы (быстрпреобр ...) и вводится ряд новых элементов, позволяющих компилятору учитывать специфику данного программного блока. Например, элемент "и" означает, что программа должна состоять из единственного фрагмента - это естественно потребовать, так как нужно будет брать конъюнкцию ее операторов. Накопитель программы блока x10 состоит из единственного пустого фрагмента. Чтобы оператор "вставкафильтра" мог регистрировать в программе блока x10 новые операторы, создается информационный элемент (вставкафильтра X), указывающий первую не определенную переменную X для начала этой программы.

После контрольной точки "прием(2)" начинается цикл обработки идентифицирующих термов, размещенных в списке $A_1, \dots, A_n$ до первого из специальных идентифицирующих термов, имеющих заголовок "вид" либо "подтерм". Обычные идентифицирующие термы, расположенные после термов "вид", "подтерм", останутся необработанными, и будет выдан отказ от компиляции приема. Текущий просматриваемый терм A_i присваивается переменной x12.

Подлежащие обработке идентифицирующие термы распознаются по наличию не идентифицированной переменной, не расположенной внутри фильтра "контекст(...)" или операторного выражения со связанными переменными. Для обработки их применяется процедура "оператор". Она сначала обращается к справочнику "значение",

а затем, как и в рассмотренной выше компиляции "обычных" фильтров, применяет справочник "оператор". Отличие от компиляции фильтров состоит в наличии у обрабатываемого терма выходных переменных. Справочник "значение" аналогичен рассматривавшемуся при компиляции фильтров справочнику "блокпроверок", с поправкой на наличие выходных переменных.

При успешной обработке текущего идентифицирующего терма A_i на его место в списке x6 помещается 0. По окончании цикла обработки идентифицирующих термов - откат к переходу через "ветвь 1".

Здесь происходит исключение из списка x6 всех нулей, возникших на местах откомпилированных идентифицирующих термов, а также термов "вид(...)" и "подтерм(...)", которые уже учтены в списках x7 и x8.

После контрольной точки "прием(3)" начинается подготовка обработки в программном блоке x10 идентифицирующих термов "вид(...)", "подтерм(...)". Рассматривается набор x7 пар (t_1 t_2), созданных по термам "вид(t_1 t_2)". По нему составляется список x11 программных выражений для корневых вхождений термов t_1, которые будут идентифицироваться с t_2. Если для некоторого t_1 подходящее программное выражение для вхождения не усматривается, но зато имеется программное выражение T, определяющее набор корневых операндов этого вхождения (в формате термов), то в список x11 на соответствующую позицию заносится 0, а в блоке x10 создается информационный элемент (набороперандов V T); V - корневое вхождение t_2. После заполнения списка x11 - переход через "иначе 1".

После контрольной точки "прием(4)" реализуется цикл определения по ненулевым элементам списка x11 установок на идентификацию. Если идентификация для t_2 одношаговая (например, t_2 - переменная, либо терм, для которого идентификация возможна с помощью справочника "вид"), то в программу блока x10 сразу заносятся выполняющие идентификацию операторы, а соответствующий разряд списка x11 заменяется на 0. Иначе определяется установка на идентификацию (операнд ...), которая регистрируется на текущей позиции x13 списка x11 вместо ранее занимавшего ее программного выражения. При этом в программу блока x10 заносится оператор, анализирующий заголовок идентифицирующего терма, а также оператор, вводящий используемую в установке (операнд ...) вспомогательную переменную для корневого вхождения. По окончании просмотра списка x11 - переход через "иначе 1".

Из списка x11 исключаются нулевые элементы, и после контрольной точки "прием(5)" реализуется цикл пополнения его установками на идентификацию для идентифицирующих термов "подтерм(t)". Переменной x12 присваивается текущий терм t списка x8. Чтобы информационный элемент (корень ...) внешнего программного блока x4, формально перенесенный в блок x10, не вносил возможных искажений в компиляцию, он удаляется. Далее, в зависимости от типа терма t, рассматриваются следующие случаи:

1. t имеет вхождение логического символа "теквхожд". Данное вхождение рассматривается как ссылка на корневое вхождение v идентифицируемого в приеме терма (например, заменяемого терма в случае приема замены). Если оно размещено внутри конструкции "внешсимвол(s теквхожд)", то в качестве точки привязки при идентификации берется не вхождение v, а вхождение той внешней операции s, операндом которой оно является. В случае, когда внешняя операция отсутствует или имеет отличный от s заголовок, сохраняется вхождение v. По выбранной точке привязки V создается установка на идентификацию (операнд V ...), регистрируемая в списке x11. Создается также информацион-

ный элемент (подтерм ...) программного блока x10, ссылающийся на корневое вхождение терма t.

2. t имеет вхождение выражения "теквхожд(A)", где A - либо переменная, либо указатель "фикс(...)" вхождения в теорему приема. В обоих случаях анализируется окрестность вхождения, идентифицированного с A. Для этого, как и выше, создаются установка (операнд ...) и информационный элемент (подтерм ...).

По окончании цикла - откат к переходу через "ветвь 1". Здесь начинается цикл учета указателей идентификации, имеющихся в списке x6. Напомним, что этот список образован не обработанными на текущий момент корневыми операндами A_i компилируемого фильтра. После контрольной точки "прием(9)" переменной x12 присваивается список всех термов, внутри которых выделены вхождения, упоминаемые в установках списка x11.

После контрольной точки "прием(10)" предпринимается учет указателей приема "отображение(...)", ссылающихся на функциональные переменные термов списка x12 - для них в программный блок x10 заносятся информационные элементы (отображение ...). Затем - переход через "ветвь 2", где начинается рассмотрение указателей идентификации x13 списка x6: Предусмотрено использование указателей с заголовками: "отображение", "единица", "отрицание", "заменазнака", "знаксуммы", "операнд", "циклупорядочение", "множество", "дробь", "набороперандов", "вариант". По ним создаются соответствующие информационные элементы блока x10, а сами указатели исключаются из списка x6. После обработки указателей - откат к переходу через "ветвь 1".

Здесь происходит пополнение списка x11 установок на идентификацию за счет фильтров и идентифицирующих термов "усм(...)", содержащихся в списке x6. Далее - переход через "ветвь 1". Переменной x12 присваивается тот из идентифицируемых термов, в котором расположено служебное слово "теквхожд"; если такого терма нет, то вместо него берется однобуквенный терм "0". Корневое вхождение терма x12 будет передаваться процедуре "идентификатор" в качестве первого операнда.

Переменной x13 присваиваются все оставшиеся термы списка x6, имеющие служебный характер. Результат исключения их из x6 образован фильтрами; он присваивается переменной x14. Наконец, вводится набор x15, полученный добавлением к списку x13 терма "условие(K)" для конъюнкции K фильтров из x14. Этот набор будет передаваться процедуре "идентификатор" в качестве заменителя описания приема. Из основного описания приема в него переносятся элементы с заголовками "символ", "комплексное", "частнпроизв", "кортежпеременных".

Если компилируется выражение вида "идентификатор(...)", возникающее при рассмотрении указателя приема "контекст(...)", то создается информационный элемент (копия ...) основного программного блока x4, ссылающийся на идентифицируемые части корневых операндов "вид(...)", "подтерм(...)" данного выражения.

Среди фильтров списка x14 выбираются фильтры x16 с заголовком "стандарт". Предпринимается попытка компиляции их относительно программного блока x10. Опережающая обработка таких фильтров нужна для предотвращения некорректных обращений к операторам, предполагающим наличие вхождения символа с идущей после него открывающей скобкой. Обработанные фильтры исключаются из списка x14.

Наконец, после контрольной точки "прием(17)" происходит обращение к процедуре "идентификатор" для обработки в программном блоке x10 установок на иденти-

фикацию, накопленных в списке x11. Таким образом, оказываются учтены идентифицирующие термы "вид(...)", "подтерм(...)" и "усм(...)". После этого компилируются оставшиеся фильтры списка x14. Далее - переход через "иначе 1".

Здесь компиляция разветвляется - в зависимости от заголовка обрабатываемого терма x5. Как уже замечалось выше, процедура справочника "блокпроверок" на логическом символе "контекст" используется для компиляции либо фильтра, либо операторного выражения специального вида. После приведенной выше общей части далее учитывается конкретная специфика компилируемого терма.

1. x5 имеет вид "и($A_1 \dots A_n$)". Такой терм x5 вводится при компиляции идентифицирующего терма вида "или(...)". Заголовок "и" здесь играет роль служебного символа. Результатом обработки является сам программный блок x10. Эти программные блоки, полученные для различных дизъюнктивных членов, будут далее использоваться программой справочника "значение" на символе "или" при сборке результирующего перечисляющего оператора. Соответственно, в данном случае выдается ответ x10.

2. x5 имеет вид "идентификатор($A_1 \dots A_n$)". Такой терм x5 вводится при обработке указателя приема "контекст(...)", обеспечивающего дополнительную идентификацию. Все переменные, идентифицированные внутри вспомогательного блока x10, будут далее использоваться в основном блоке x4. Соответственно, программа блока x10 присоединяется к программе блока x4, и все обеспечивающие идентификацию информационные элементы блока x10 передаются блоку x4. В качестве ответа выдается единица, сигнализирующая об успешном завершении компиляции.

3. x5 имеет вид "примеч($A_1 \dots A_n\ B$)". Такой терм представляет собой указатель приема, определяющий ввод серии комментариев B для значений переменных, порождаемых условиями $A_1, \dots, A_n$. Строится оператор x19, обеспечивающий на текущем шаге перечисления наборов значений переменных ввод необходимого комментария задачи; на основе программы блока x10 и оператора x19 создается кванторная импликация x21, реализующая цикл перечисления, и эта импликация регистрируется в программе блока x4. В качестве ответа выдается единица.

4. Остальные случаи, когда x5 не является фильтром "контекст(...)", сгруппированы в общую ветвь. Она начинается с создания по вспомогательному программному блоку x10 оператора x16 вида "существует($x_1 \dots x_m\ B$)". Здесь B - конъюнкция операторов программы блока x10; $x_1, \dots, x_m$ - все новые программные переменные, использованные в B. Этот оператор - сугубо промежуточный; в зависимости от заголовка терма x5, по нему будет строиться итоговое программное выражение x17. Сначала переменной x17 присваивается 0. Затем рассматриваются следующие подслучаи:

   (a) x5 имеет вид "число($A_1 \dots A_n$)". Тогда x17 переприсваивается выражение "суммавсех($x_1 \dots x_n\ B\ 1$)".

   (b) x5 имеет вид "нок($A_1 \dots A_n\ C$)" либо "нод($A_1 \dots A_n\ C$)". Переменной x18 присваивается программное выражение для численного значения C. Наименьшее общее кратное либо наибольший общий делитель вычисляются в цикле, с использованием накопителя N, инициализируемого нулем. Роль

этого накопителя играет первая неопределенная программная переменная блока x4. Оператор x19 обеспечивает пересчет накопителя на очередном шаге цикла: если накопитель был равен 0, то ему присваивается текущее значение C, иначе - вычисляется нок либо нод. В случае нахождения наибольшего общего делителя после оператора x19 вставляется оператор, обеспечивающий замену нулевого значения N на единицу.

(c) x5 имеет вид "максимум($A_1 \dots A_n$ C)" либо "минимум($A_1 \dots A_n$ C)". Случай аналогичен предыдущему, но в цикле вычисляются максимум либо минимум.

(d) В остальных случаях x5 имеет вид $F(A_1 \dots A_n$ $C)$ для некоторого описателя F. Операторное выражение, вычисляющее значение x5, строится с помощью того же самого символа описателя. Если C никак не помечено, то его значение вычисляется в формате логического символа. Кроме того, оно может быть помещено под служебным символом "терм" либо "транскоммент" (для компиляции комментария).

5. x5 - фильтр "контекст(...)".

   Если некоторое A_i имело вид "буфер(C D)", то оно сохраняется в списке x14 и не рассматривается в указанном выше цикле синтеза операторов фильтра по x14. Указатель "буфер(...)" означает, что по мере перечисления значений внутренних переменных компилируемого фильтра x5 должны вычисляться значения операторного выражения D. Эти значения передаются в накопитель N комментария (C N), извлекаемого из контекста срабатывания приема. Операторы, обеспечивающие указанные действия, включаются в консеквент кванторной импликации x25, перечисляющей допустимые значения переменных согласно программе блока x10. Если хотя бы однажды значение D было вычислено, то индикатор i существования требуемых значений переменных изменяется на единицу. Оператор x26 для проверки фильтра x5 после этого строится как "существует(i и(равно(i 0)x25 равно(i 1)))".

   В остальных случаях оператор фильтра строится путем навешивания квантора существования на конъюнкцию операторов программы блока x10.

19.18 Компиляция преобразующей части приема сканирования задачи

Для создания завершающей части программы приема сканирования задачи, осуществляющей основные преобразования приема, служит процедура "преобразователь(x1 x2 x3 x4 x5)". Ей передаются входные данные: x1 - теорема приема; x2 - заголовок приема; x3 - описание приема; x4 - программный блок; x5 - узел приема. Выйти на начало программы этой процедуры можно через пункт "Формирование преобразующей части приема, активизируемого при сканировании задачи" оглавления программ.

19.18.1 Предварительный учет указателей приема

На первом этапе происходит просмотр и обработка тех указателей приема, которые определяют действия, выполняемые до основного преобразования приема. Рассматриваются следующие указатели:

1. Указатель приема "контроль(P лимит(N) K)". Должна вводиться установка на откат к текущему состоянию задачи, если через N шагов работы интерпретатора ЛОСа не будет выполнено условие P. K - набор термов "замечание(...)", определяющих комментарии A, которыми сопровождается сохраняемая для отката версия текущего состояния задачи. Этот набор может быть пустым. Заметим, что данный указатель используется в решателе крайне редко.

   Чтобы организовать слежение за выполнением условия P, прием будет вводить или пополнять ранее введенный комментарий к посылкам задачи (контроль A_1 A_2). Здесь A_1 - число, указывающее содержимое счетчика шагов, по достижении которого требуется перейти к анализу установок на откат, хранящихся в наборе A_2. Каждая установка в наборе A_2 представляет собой набор ($B_1 \ldots B_5$). B_1 - номер шага работы интерпретатора, по достижении которого следует предпринять анализ выполнения условия P, определяемого данной установкой на откат. B_2 - номер шага работы интерпретатора, на котором была введена установка; B_3 - логический символ, по которому следует предпринять обращение к справочнику "контроль", выполняющему проверку этого условия (откат происходит при его нарушении); B_4 - тройка (номер узла того приема, который ввел установку - заголовок приема справочника "контроль", обеспечивающего проверку условия - набор дополнительных входных данных). B_5 - исходное состояние текущей задачи, к которому выполняется откат.

   Как видно из описания структуры установки, проверка условия P обеспечивается специальным приемом справочника "контроль". Этот сопровождающий прием будет создаваться рассматриваемой ветвью компилятора одновременно с основным приемом. Справочник "контроль" имеет следующие входные данные: x1-x5 - такие же, как в сканировании задачи, причем текущее вхождение не выделено, т.е. x2-x4 равны 0; x6 - номер узла основного приема; x7 - заголовок приема справочника "контроль"; x8 - набор дополнительных входных данных. Текущий символ обращения к справочнику - тот символ, за которым закреплен основной прием. Входные данные x6, x7 позволяют программе справочника отсекать "чужие" приемы при проверке условия P. Необходимая для отсечения информация содержится в заголовке приема справочника, имеющем вид "контроль(C_1 C_2)", где C_1 - заголовок основного приема; C_2 - номер указателя "контроль(...)" этого приема, по которому был скомпилирован прием справочника.

   Обработка указателя, присвоенного переменной x7, начинается после контрольной точки "прием(65)". Переменная x6 имеет своим значением номер текущего указателя "контроль(...)". Переменной x8 присваивается фильтр P; x9 - число N; x10 - список термов "замечание(...)"; x11 - список свободных переменных фильтра P.

   После контрольной точки "прием(66)" вводится накопитель x12 программных выражений для дополнительных входных данных, передаваемых справочнику. Эти входные данные суть результаты идентификации свободных переменных фильтра P. Одновременно вводится накопитель x13 информационных элементов, задающих значения свободных переменных фильтра P как элементы набора дополнительных данных, передаваемого программе справочника. Набор x13 будет включен во вспомогательный программный блок, относительно которого компилируется указанная программа. По окончании заполнения накопителей x12, x13 - откат к переходу через "ветвь 3".

Для обработки термов "замечание(...)" вводятся накопители x14, x15. Первый заполняется программными выражениями для безусловных комментариев; второй - программными выражениями для наборов комментариев, вводимых в зависимости от контекста. Каждый такой набор - либо одноэлементный, либо пустой. По окончании цикла обработки комментариев - переход через "иначе 1".

Создается программное выражение x16 для общего списка комментариев, передаваемых сохраняемой задаче. Создается программное выражение x20 для заголовка приема справочника "контроль", и далее - программное выражение x21 для оператора "установка(...)", обеспечивающего ввод установки на откат. Этот оператор регистрируется в компилируемой программе.

Наконец, предпринимается компиляция программы справочника "контроль", для чего используется вспомогательный программный блок x22. Собственно синтез программы обеспечивается процедурой "фильтр", обрабатывающей условие P. Здесь же осуществляется запись этой программы в программном блоке.

2. Указатель приема "задачи(N P)". Этот указатель можно использовать при отладке приема. Он обеспечивает сохранение ссылок на те задачи, в которых прием сработал так, что было выполнено условие P. Переменной x9 присваивается программное выражение для оператора "учетзадачи(...)", сохраняющего ссылку на прием в комментарии (учетзадачи ...) к посылкам исходной задачи. Чтобы отбирались только ситуации, в которых выполнено условие P, находится оператор x11 проверки данного условия, и в компилируемую программу заносится дизъюнкция отрицания x11 с x9.

3. Указатель приема "биключ(...)" (контрольная точка "прием(2)"). Он определяет проверку отсутствия заданного комментария задачи непосредственно перед выполнением преобразований приема. Если комментария не было, то происходит автоматическая его регистрация.

   Переменной x8 присваивается набор программных выражений для разрядов комментария, следующих после его заголовка; переменной x9 - набор, полученный добавлением к началу набора x8 переменной "x1" и заголовка комментария. Таким образом, получается список программных выражений для входных данных операторов, проверяющих отсутствие комментария либо вводящих его. Если по информации, имеющейся в фильтрах приема, удается определить тип текущей задачи, то указанные операторы адресуют комментарий либо списку общих комментариев задачи, либо списку комментариев к посылкам. В противном случае обработка комментария осуществляется условным оператором "альтернатива(...)", предварительно анализирующим тип текущей задачи.

4. Указатель приема "примечпосылки(...)" (контрольная точка "прием(3)"). Происходит ввод комментария к посылке либо условию задачи, идентифицированным с заданным антецедентом теоремы приема.

   Переменной x9 присваивается набор программных выражений для разрядов комментария, следующих после его заголовка; переменной x11 - вхождение рассматриваемого антецедента теоремы приема. Если усматривается, что речь идет о посылке задачи, переменной x12 присваивается 0, иначе - 1. Если точка

привязки расположена в выделенном антецеденте, то сразу создается оператор, регистрирующий комментарий к текущему терму задачи; его вхождение в задачу известно - это "x3". Аналогичным образом строится оператор, если вхождение в задачу терма, идентифицированного с рассматриваемым антецедентом, уже определялось ранее и было сохранено в информационном элементе (вхождениевзадачу ...). В остальных случаях для поиска вхождения применяется оператор "посылка(...)" (при x12 = 0) либо "вхождениевзадачу(...
)".

5. Указатель приема "примечание(...)" (контрольная точка "прием(4)"). Случай аналогичен предыдущему, но вводится комментарий к новому либо преобразуемому терму задачи. Фактически здесь рассматривается только подслучай приема замены, причем в указателе отсутствует символ "результат". Прочие подслучаи будут обрабатываться после компиляции основного преобразования приема. Создается оператор, регистрирующий комментарий в списке комментариев к преобразуемому терму задачи. Отдельно рассматриваются случаи безусловного и условного ввода комментария.

6. Указатель приема "удалениезамечания(A B)" (контрольная точка "прием(5)"). Перед выполнением преобразования приема, в случае истинности фильтров B, происходит удаление комментариев, определяемых идентифицирующим термом A. Фильтры могут отсутствовать; A имеет вид "комментарий(...)" либо "комментарийпосылки(...)".

   Для обработки указателя создается вспомогательный программный блок x8. Сначала относительно x8 компилируется идентифицирующий терм A, затем - фильтры списка B. После этого создается кванторная импликация, антецеденты которой суть все операторы программы блока x8, а консеквент - оператор, исключающий комментарий.

7. Указатель приема "новаяпосылка(x $P(x)$ v)" либо "новоеусловие(x $P(x)$ v)" (контрольная точка "прием(6)"). Должно быть выполнено уменьшение весов посылок и условий x текущей задачи, удовлетворяющих фильтру $P(x)$, до величины v. Перед x возможно размещение терма "условие(Q)", указывающего условие Q на текущий контекст, при выполнении которого выполняется данное действие.

   Переменной x7 присваивается x; вводится вспомогательный программный блок x8, соответствующий текущему шагу просмотра посылок либо условий. В нем переменная x является идентифицированной. Находится результат x9 компиляции фильтра $P(x)$ относительно блока x8. Строится вспомогательный оператор x10, обеспечивающий синхронный просмотр списка термов задачи и списка их весов. По нему и оператору фильтра x9 создается оператор x11, осуществляющий просмотр допустимых термов задачи и изменение их весов.

   Если имеется фильтр Q, то находится результат x13 компиляции его, после чего оператор x11 заменяется на дизъюнкцию отрицания x13 и старой версии x11. Затем предпринимается регистрация x11 в программе приема.

8. Указатель приема "замечание(...)" либо "комментариипосылок(...)" (контрольная точка "прием(7)"). Соответственно, должен вводиться комментарий к задаче либо к ее списку посылок. В данном пункте рассматривается лишь подслучай, когда указатель не содержит символа "результат", причем либо прием

не имеет заголовка "вывод", либо фильтры приема также не содержат символа "результат", а указатель не ссылается на вхождение в консеквент теоремы приема.

Действия аналогичны случаю указателя "примечание".

9. Указатель приема "изменение(A B)" (контрольная точка "прием(8)"). Здесь A - идентифицирующий терм "комментарий(...)", "комментарийпосылки(...)" либо "примечание(...)", с помощью которого происходит перечисление требуемых комментариев. B - набор термов "замена(i T)", где i - номер изменяемого разряда комментария (заголовок комментария имеет номер 0); T - операторное выражение, значением которого является заменяющий объект. Кроме того, в B могут содержаться фильтры, ограничивающие множество обрабатываемых комментариев.

   Для обработки текущего шага просмотра комментариев вводится вспомогательный программный блок x9. Сначала относительно x9 компилируется идентифицирующий терм A, затем - содержащиеся в B фильтры. Переменной x13 присваивается набор программных выражений для заменяющих объектов T; переменной x15 - список операторов, выполняющих изменение разрядов. Наконец, создается кванторная импликация x17 для всего цикла просмотра комментариев.

10. Указатель приема "посылка(A B C D)" (контрольная точка "прием(85)"). Здесь A - терм вида "условие(Q)", определяющий фильтр Q, при истинности которого будет вводиться дополнительная посылка C текущей задачи. B - список утверждений в "теоремных" переменных, истинность которых должна быть установлена перед вводом посылки; D - набор термов "примечание(...)", определяющих сопровождающие посылку комментарии.

    Переменной x7 присваивается список корневых операндов указателя. Из него выбираются: список x8, образованный термом A, если он есть, и пустой в противном случае, а также список x9, образованный термами "примечание(...)". Оба эти списка исключаются из x7. Теперь последний элемент списка x7 - утверждение C. Оно присваивается переменной x10 и также исключается из x7. В результате x7 будет состоять из проверяемых перед занесением посылки утверждений "теоремного" уровня. Находятся программное выражение x11 для C, а также список x12 программных выражений для сопровождающих C комментариев. Затем - переход через "иначе 4".

    Если списки x7 и x8 пусты, то сразу вводится оператор, присваивающий текущей неиспользуемой программной переменной набор (вывод ...) и регистрирующий эту переменную в информационном элементе (ключвывода ...). Набор (вывод ...) будет впоследствии передаваться процедуре, реализующей основное преобразование приема. Если хотя бы один из списков x7, x8 непуст, то переход через "иначе 1". Здесь также осуществляется создание и регистрация набора (вывод ...), но позиции его пока "пустые". Вводится накопитель x14 операторов, проверяющих истинность условий ввода посылки. Если имеется фильтр Q, то сначала компилируется он. Затем рассматриваются элементы списка x7. Для компиляции обработки их проверочными операторами создается вспомогательный программный блок x17. Информация об использованных при проверке утверждениях передается в набор (вывод ...) и впоследст-

вии будет учтена в комментариях к новой посылке C. Обращения к проверочным операторам заносятся под квантор существования, чтобы связать их выходные переменные, и регистрируются в накопителе x14. Наконец, создается набор x15 операторов, заполняющих пустые позиции набора (вывод ...). Чтобы эти операторы выполнялись при истинности фильтров списка x14, строится дизъюнкция отрицаний фильтров, после которых размещена конъюнкция операторов.

11. Указатель приема "вывод(A B C)" (контрольная точка "прием(57)"). В дополнение к основному преобразованию приема осуществляется вывод следствия B, используемого для сопровождения по о.д.з. A - терм "условие(Q)", определяющий условие, при котором происходит вывод. C - терм "эквивалентно($n_1 \dots n_k$)", перечисляющий номера n_i антецедентов теоремы, являющихся следствиями утверждения B и остальных антецедентов теоремы. Посылки, с которыми идентифицированы такие антецеденты, после применения приема удаляются, если только они не нужны для сопровождения по о.д.з. Термы A, C могут отсутствовать.

    Сначала переменной x7 присваивается список всех корневых операндов указателя. Если имеется фильтр Q, то переменной x8 присваивается программное выражение для проверяющего оператора, а терм A из списка x7 удаляется. Затем - переход через "ветвь 2", где переменной x9 присваивается программное выражение для новой посылки B. Переменной x11 присваивается 0; если имеется терм C, то ей переприсваивается программное выражение для набора посылок, идентифицированных с указанными в C антецедентами. Переменной x12 присваивается набор программных выражений для наборов посылок, использованных при срабатывании приема. Находится программная переменная для накопителя (коррекцияпосылок ...); если этого накопителя в программе приема еще не было, то он вводится. Далее создается оператор x15 - обращение к процедуре "учетвывода", которая будет регистрировать в накопителе (коррекцияпосылок ...) информацию о выводимости антецедентов. Эта информация впоследствии будет учтена процедурой, выполняющей основное преобразование приема. При наличии фильтра Q в компилируемой программе регистрируется не сам оператор x15, а дизъюнкция "или(не(x8)x15)".

12. Указатель приема "вводтерма(i)" (контрольная точка "прием(39)"). i - й антецедент теоремы приема имеет вид "равно(A x)" и используется для ввода обозначения x известного выражения A. При выполнении приема определяемое данным антецедентом равенство заносится в список посылок задачи и снабжается комментарием "ориентацияравенства", блокирующим перестановку частей.

    Находятся программные выражения x11, x12 для x, A; по ним строится программное выражение x13 для равенства. Находится также программное выражение x14 для x в формате переменной. В компилируемую программу заносятся операторы, выполняющие регистрацию новой посылки x13, снабжение ее коментарием "ориентацияравенства", а также ввод общего комментария задачи (вспомпараметр x).

13. Указатель приема "вспомнеизв($x_1 \dots x_n$)" (контрольная точка "прием(59)"). Вводятся новые переменные $x_1, \dots, x_n$, которые становятся новыми несущест-

венными неизвестными задачи на описание (т.е. регистируются в цели (параметры ...)). Переменной x7 присваивается список программных выражений для этих переменных, и в компилируемую программу вводятся операторы "вспомнеизв", регистрирующие переменные в целях (неизвестные ...), (параметры ...).

14. Указатели "удалениепосылки(A B C)", "удалениеусловия(A B C)" (контрольная точка "прием(10)"). Если существует посылка (условие) C текущей задачи, причем выполнены дополнительные ограничения, определяемые термом A вида "условие(Q)", и C не было использовано при нормализации надтермов термов $T_1, \ldots, T_n$, перечисленных в терме B вида "нормализатор($T_1 \ldots T_n$)", то после выполнения основного действия приема проверяется, используется ли C для сопровождения по о.д.з., и если не используется, то удаляется. Термы A, B могут отсутствовать.

    Если уже имелся информационный элемент (удалениепосылки x) либо, соответственно, (удалениеусловия x), у которого x - программная переменная, используемая как накопитель списка удаляемых посылок (условий), то переменной x7 присваивается x. Иначе - такой элемент вводится, после чего значение x7 меняется указанным образом. Затем - переход через "ветвь 2".

    Здесь компиляция разветвляется. Сначала рассматривается случай, когда терм A имеется, и обработка его фильтра предполагает использование проверочных операторов. Вводится вспомогательный программный блок x9, относящийся к текущему шагу цикла просмотра посылок (условий). Если C есть простая либо функциональная переменная, то переход через "иначе 2", где терм C будет определен с помощью процедуры "прогрвыражение". Иначе - он идентифицируется с помощью обращения к процедуре "идентификатор". В остальном оба случая аналогичны, поэтому рассмотрим здесь лишь второй из них. Переменной x11 присваивается связывающая приставка для кванторной импликации, которая будет выполнять цикл просмотра посылок (условий). Далее - переход через "ветвь 3". Здесь создается еще один вспомогательный программный блок x12 для компиляции фильтра Q. Ему передается информационный элемент (внешконтроль y), где y - накопитель использованных при проверке фильтра Q утверждений. Затем определяется результат x14 компиляции фильтра Q и строится кванторная импликация x15, реализующая цикл просмотра и отбора удаляемых посылок (условий). Отбираемые утверждения регистрируются в накопителе x7. Если утверждение было использовано для установления истинности Q, то оно не отбирается. Наконец, создается кванторная импликация x16, в которой инициализируется накопитель y, выполняется обработка Q, и далее реализуется оператор x15. Она и заносится в компилируемую программу.

    Случай, когда терм A либо отсутствует, либо его обработка не требует проверочных операторов, аналогичен, но проще - не нужен накопитель y. Заметим, что учет терма B и собственно удаление посылок (условий) будут выполняться впоследствии - здесь лишь заполняется накопитель x7.

19.18.2 Компиляция основного действия приема

После контрольной точки "прием(11)" начинается компиляция основного преобразования приема. В зависимости от заголовка приема, она выполняется различными

ветвями программы.

Прием замены

Подслучай рассматривается после контрольной точки "прием(12)". Переменной x6 присваивается вхождение консеквента теоремы приема (если теорема не является кванторной импликацией, то - вхождение корня теоремы). Проверяется, что по данному вхождению расположено равенство либо эквивалентность. Переменной x7 присваивается заменяющий терм теоремы. Инициализируется нулем накопитель x8 программного выражения для заменяющего терма. Переменной x9 присваивается программное выражение для вхождения корня заменяемого терма. Далее для компиляции заменяющего терма рассматриваются следующие подслучаи:

1. Имеется информационный элемент (результат x), содержащий программную переменную x для заменяющего терма, который уже был ранее откомпилирован. Тогда x8 заменяется на терм x (контрольная точка "прием(14)").

2. Имеется информационный элемент (набор A B), у которого A - вхождение в заменяемую часть теоремы коммутативно-ассоциативной операции, расформируемой при выполнении приема с указателем "набор(первыйтерм)". B - программное выражение для вхождения A. По вхождению A расположен терм вида $f(ab)$; заменяющий терм x7 имеет вид $g(t(a)t(b))$. Здесь a, b - переменные. После контрольной точки "прием(21)" переменной x11 присваивается символ коммутативно- ассоциативной операции g; программной переменной x12 - теоремная переменная a; переменной x14 - терм $t(a)$. Создается вспомогательный программный блок x15, относящийся к шагу цикла просмотра операндов идентифицированного с A расформируемого вхождения. Для этого шага определяется программное выражение x16, значением которого служит текущий терм $t(a)$. Далее строится выражение x18 для всего списка термов $t(a)$, возникающих в цикле. Если при компиляции была введена переменная-модификатор, идентифицируемая с остатком операндов заменяемого терма, то x18 пополняется этими операндами. Наконец, x8 заменяется на выражение "сборка(x11 x18)". Дополнительно учитываются указатели нормализации для x7.

3. Заменяемый и заменяющий термы имеют своим заголовком одну и ту же коммутативно-ассоциативную операцию f (контрольная точка "прием(22)"). Если заменяющий терм получается добавлением к операндам заменяемого терма единственного нового операнда A, то находится программное выражение x15 для A. Список корневых операндов заменяемого терма равен "набороперандов(x9)"; к нему присоединяется x15, и строится программное выражение x16 для заменяющего терма. Если заменяющий терм отличается от заменяемого только единственным операндом, то корректируются x9 и x7: новые версии заменяемого и заменяющего терма связываются только с этим операндом. В этом случае x8 пока не определяется; после коррекции предпринимается откат для продолжения сравнения заменяемого и заменяющего термов.

4. Общий случай (контрольная точка "прием(24)"). Здесь программное выражение для заменяющего терма находится с помощью процедуры "метаперевод".

После получения программного выражения x8 для заменяющего терма предпринимается цепочка коррекций этого выражения:

1. Учитывается информационный элемент (корни ...), хранящий программное выражение для остатка корневых операндов при идентификации коммутативно-ассоциативной операции с указателем "развертка" (см. фрагмент, содержащий контрольную точку "прием(24)").

2. Учитывается информационный элемент (заменагруппы ...), хранящий программные выражения для остаточных наборов слева и справа от заменяемой части - если корневая операция ассоциативна, но не коммутативна.

3. Учитывается указатель приема "кванторнаясвертка" (контрольная точка "прием(26)"). В этом случае квантор - заголовок заменяемого терма - идентифицируется с возможными преобразованиями перехода к противоположному квантору и группировки части подкванторных утверждений под вспомогательный квантор.

   Переменной x10 присваивается вхождение заменяемого квантора. Если этот квантор является квантором общности, то x8 переприсваивается выражение для отрицания заменяющего утверждения теоремы. Таким образом, x8 становится отриентированным на случай идентификации x10 с квантором существования. Впрочем, это лишь промежуточное выражение - настоящий заменяющий терм определяется далее с помощью обращения к процедуре "кванторнаясвертка(...)", учитывающей указанные выше преобразования.

4. Учитывается указатель приема "дробь" (контрольная точка "прием(27)"). В этом случае при идентификации возможна перестановка операндов заменяемого терма, и такая же перестановка делается для операндов заменяющего терма.

   По информационному элементу (корень ...) определяется программное выражение для вхождения, идентифицированного с заменяемой частью теоремы; по информационному элементу (операнд ...) - программное выражение для вхождения, идентифицированного с первым операндом заменяющей части. Эта информация позволяет распознать наличие перестановки операндов; соответственно корректируется x8.

После контрольной точки "прием(89)" происходит учет указателя приема "примеч(A K)". Здесь должна вводиться серия комментариев задачи, возникающих при перечислении значений переменных согласно списку A идентифицирующих термов, фильтров и указателей. Терм K вида "замечание(...)" определяет текущий комментарий. Для создания операторов, реализующих эти действия, используется обращение к справочнику "блокпроверок" на логическом символе "контекст" (см. выше).

После контрольной точки "прием(83)" происходит учет указателя "новыенеизвестные(X)". X - список новых неизвестных текущей задачи, вводимых приемом.

Вводятся два накопителя: x11 - для отдельных переменных, идентифицированных с элементами списка X; x12 - для групп таких переменных. Создается программное выражение x15 для итогового списка новых неизвестных; регистрация их в задаче выполняется оператором "учетнеизвестных(...)".

После контрольной точки "прием(80)" проверяется наличие фильтров приема (а также указателей "задачи(...)"), содержащих логический символ "результат". Если они есть, то вводится программная переменная x10 для заменяющего терма x8, и создается информационный элемент (результат x10). Он позволит переходить от служебного символа "результат" к выражению для заменяющего терма.

После контрольной точки "прием(28)" происходит обработка серии фильтров приема, содержащих символ "результат" (ранее они не рассматривались). Действия здесь совершенно аналогичны тем, которые осуществлялись в основном цикле обработки фильтров, однако служебный символ "результат" на этом этапе уже доступен для компиляции.

Аналогичным образом, после контрольной точки "прием(81)" обрабатываются содержащие символ "результат" указатели приема "задачи(...)".

После контрольной точки "прием(78)" рассматриваются указатели приема "актив(N p)". Должно выполняться уменьшение до уровня p весов посылок и условий, идентифицированных с антецедентами, номера которых образуют список N. Программные выражения для таких посылок и условий находятся через информационные элементы (антецедент ...) либо непосредственно с помощью процедуры "метаперевод". Далее в компилируемую программу заносится оператор "актив(...)", локализующий в текущей задаче посылку либо условие и уменьшающий ее вес.

После контрольной точки "прием(29)" начинается формирование программного выражения для набора R информационных элементов, передаваемых оператору "заменавхождения" (т.е. входной переменной x6 этого оператора). Сначала определяется список x10 программных выражений для групп утверждений, использованных приемом при обосновании корректности замены. Инициализируется пустым словом накопитель x11 программных выражений для информационных элементов набора R (в нижеследующем перечислении, говоря о перенесении в список x11 различных информационных элементов, подразумеваем программные выражения для этих элементов). Заполнение накопителя x11 выполняется следующим образом:

1. Создаются информационные элементы для ссылки на прием и для ссылки на использованные приемом утверждения контекста замены (контрольная точка "прием(30)").

2. Создается информационный элемент (примечание ...), в котором перечисляются комментарии, сопровождающие изменяемый терм задачи (контрольная точка "прием(31)"). Напомним, что накопитель таких комментариев для указателей приема "примечание(...)", не содержащих логического символа "результат", уже был введен на предшествующих этапах компиляции, и зарегистрирован в информационном элементе программного блока (примечание ...). Здесь выполняется компиляция оставшихся указателей "примечание(...)" и передача объединенного списка накопителю x11.

3. Указатель приема "нормализация" передается в список x11 непосредственно (контрольная точка "прием(9)"). Он включает механизмы дополнительной стандартизации надтермов заменяемого терма, предусмотренные в процедуре "заменавхождения".

4. Указатель приема "обозначения(...)", перечисляющий введенные приемом переменные, требующие регистрации в цели (обозначения ...), порождает одноименный информационный накопителя x11 (контрольная точка "прием(18)").

5. Указатель приема "установка(A B)", определяющий пополнение цели (A ...) текущей задачи элементом B, порождает одноименный информационный элемент для x11 (контрольная точка "прием(15)").

6. Указатели приема "норм", "вычерк", "внешвывод" передаются в список x11; два последних символа при этом изменяются на символы "свертка" и "новая-посылка".

7. Указатель приема "титр(...)", обеспечивающий создание поясняющих срабатывания приема текстов, компилируется процедурой "транститр" в информационный элемент списка x11 (контрольная точка "прием(17)").

8. Указатель приема "выводусловия(...)", необходимый для создания сопровождающего замену дополнительного условия текущей задачи на описание, порождает одноименный информационный элемент для x11 (контрольная точка "прием(76)").

9. Указатели приема "сопровождение", "стандследствие" передаются в список x11 непосредственно (контрольные точки "прием(62)", "прием(16)").

10. Указатель приема "контрольнормализации(...)", определяющий исключение из буфера результатов обращений к заданному нормализатору всех результатов, содержащих заданный подтерм, порождает одноименный информационный элемент для x11.

11. Происходит передача в x11 накопителей (удалениепосылки ...), (удалениеусловия ...), ранее созданных в программном блоке (контрольная точка "прием(38)").

12. Информационный элемент программного блока (норм ...), необходимый для блокировки удаления посылок, использованных приемом при нормализации термов, порождает одноименный элемент для x11.

13. Извлекаемый из информационного элемента (коррекцияпосылок A) накопитель A, также имеющий вид (коррекцияпосылок ...), передается в список x11 (контрольная точка "прием(40)").

14. Извлекаемый из информационного элемент (ключвывода ...) накопитель (вывод ...), определяющий вывод сопровождающего утверждения, переносится в список x11 (контрольная точка "прием(86)").

15. Если вхождение x9 заменяемого подтерма в процессе компиляции было изменено по сравнению с вхождением, указанным в информационном элементе (корень ...), то последний передается в список x11 (контрольная точка "прием(91)").

16. Далее находится программное выражение для списка информационных элементов, перечисленных в x11, которое переприсваивается переменной x11. После этого пополнение данного списка продолжается в новом формате: каждый раз вводится дополнительный операнд терма x11 вида "набор(...)".

17. Указатели приема "замечание(...)" и "комментариипосылок(...)", не обработанные ранее из-за того, что они содержали символ "результат", порождают одноименные информационные элементы для накопителя x11 (контрольная точка "прием(41)").

18. Указатель приема "Замечание(...)", используемый для передачи комментариев вспомогательной задаче, вводимой оператором "попытказамены", порождает одноименный информационный элемент.

После того, как определено x11, учитывается указатель приема "стоп" (контрольная точка "прием(25)". По нему вводится оператор "трассировка(стоп 0)", обеспечивающий выход в отладчик ЛОСа перед применением приема.

Собственно формирование обращения к оператору, выполняющему замену, начинается с контрольной точки "прием(42)". Переменным x12 и x13 присваиваются программные переменные для второй и третьей компонент координаты вхождения заменяемого терма в задачу. Обычно это переменные "x3" и "x4". Однако, в ряде случаев точка привязки приема замены может быть выбрана не в заменяемой части теоремы, а в ее антецеденте. Тогда информационный элемент (переходник ...) сохраняет вторую и третью компоненты координат вхождения заменяемого терма, и они переприсваиваются переменным x12, x13. Затем - переход через "ветвь 1".

Сначала рассматривается один особый случай - наличие у приема указателя "дизъюнктчлен(...)" (контрольная точка "прием(47)"). Напомним, что этот указатель выделяет антецеденты вида $A = B$, помеченные также указателем "идентификатор". При обработке антецедента находится множество дизъюнктивных членов идентифицирующей части A, которые последовательно идентифицируются с идентифицируемой частью B. Прием осуществляет эквивалентную замену, причем заменяющий терм строится как дизъюнкция заменяющих термов, созданных согласно теореме приема для отдельных подслучаев указанного перечисления вариантов идентификации. В процессе компиляции по указателю был создан информационный элемент программного блока (дизъюнктчлен A_1 A_2 A_3 A_4), у которого A_1 - набор вхождений выделенных антецедентов; A_2 - программная переменная для накопителя дизъюнктивных членов заменяющего терма; A_3 - метка отката при обрыве идентификации; A_4 - набор меток отката для продолжения перечислений идентифицируемых дизъюнктивных членов, имеющий ту же длину, что и набор A_1.

Информационный элемент (дизъюнктчлен ...) присваивается переменной x14. Переменной x15 присваивается номер последней из меток отката, упомянутых в наборе A_4. После определения версии заменяющего терма для текущего подслучая и регистрации ее в накопителе A_2 продолжение перечисления будет обеспечиваться откатом по метке x15. Фрагмент программы P, выполняющий действия по завершении перечисления, сохранен в информационном элементе (блокпрограммы 1 P). Переменной x17 присваивается программное выражение для итогового заменяющего терма - дизъюнкции утверждений накопителя A_2, после чего к концу фрагмента P добавляется оператор "заменавхождения", выполняющий замену (либо оператор "попытказамены", относящий ее к вспомогательной задаче), а также оператор "пересмотр". Заметим, что содержимые набора x11 для всех подслучаев объединяются в специальном накопителе (программная переменная, номер которой на единицу больше номера переменной A_2, и итоговый список передается процедуре, выполняющей замену.

Наконец, в "общем" случае строится оператор x14, выполняющий замену. Если прием имеет указатель "попытказамены", то замена будет выполняться не в текущей задаче, а в ее вспомогательной копии. Для этого используется оператор "попытказамены(...)". Если вспомогательная задача решается, то ответ на нее передается исходной задаче через комментарий (ответ ...). В противном случае оператор "попытказамены(...)" оказывается ложным, и процесс решения текущей задачи продолжается. При отсутствии указателя "попытказамены" создается обращение к оператору "заменавхождения(...)". Сразу заметим, что использование указателя "попытказамены" в решателе - исключительная редкость. При создании операто-

ра x14 учитывается также указатель "корень", используемый в ситуациях, когда заменяемая часть теоремы вырождена и представляет собой обычную или функциональную переменную, идентифицируемую с условием текущей задачи на преобразование.

Если программное выражение для заменяющего терма имеет вид "заменаоперанда($v\ w\ t$)", причем v - выражение для заменяемого вхождения, то x14 модифицируется так, чтобы новым вхождением заменяемого терма было w, а заменяющим термом - t. Затем - откат к переходу через "ветвь 1".

После контрольной точки "прием(43)" обрабатывается, если он есть, фильтр "контрольодз". Этот фильтр требует, чтобы все подвыражения заменяющего терма имели истинные в контексте замены ограничения на о.д.з. Для проверки в программу вставляется обращение к оператору "контрольодз(...)".

После контрольной точки "прием(44)" учитывается указатель приема "заменавхождений". Этот указатель требует проведение одновременной замены в задаче всех вхождений заменяемого терма на заменяющий. Фактически замена будет затрагивать лишь те вхождения, в контексте которых имеются все утверждения, использованные для обоснования корректности данного перехода. Для учета указателя обращение к оператору "заменавхождения" заменяется обращением к оператору "заменавхождений".

Теперь оператор x14 заносится в программу. Если имелся указатель приема "внимание", определяющий уменьшение до 0 весов всех условий и посылок, содержащих заданный подтерм, то после x14 размещается обращение к оператору "внимание(...)" (контрольная точка "прием(45)"). Компиляцию завершает добавление оператора "пересмотр".

Прием вывода

Подслучай рассматривается после контрольной точки "прием(48)". Прием имеет заголовок "вывод" (добавление новой посылки) либо "выводусловия" (добавление нового условия). Прежде всего, находится программное выражение x6 для добавляемого утверждения. Однако, фактически ссылки на новое утверждение будут делаться не через x6, а через инициализируемую далее нулем переменную x7. Если уже имелся информационный элемент (результат T), то далее программное выражение x6 игнорируется, а x7 присваивается выражение T. Если прием имеет заголовок "выводусловия", то x7 переприсваивается x6. Если же прием имеет заголовок "вывод", то для x6 предварительно вводится вспомогательное обозначение - новая программная переменная, которая и переприсваивается переменной x7.

Если информационного элемента (результат ...) еще не было, то он вводится. После этого реализуется цикл обработки оставшихся фильтров приема, содержащих символ "результат". По окончании цикла - откат к переходу через "ветвь 3".

После контрольной точки "прием(88)" переменной x8 присваивается список программных выражений для информационных элементов (выводимо A), перечисляющих подгруппы A утверждений, использованных приемом. В этот же набор включаются выражения для элементов (задача ...), ссылающихся на использованные приемом вспомогательные задачи. Переменной x9 присваивается дополняющий x8 набор - пустой либо одноэлементный, содержащий программное выражение для элемента (выводимо ...).

После контрольной точки "прием(49)" выполняется учет указателей приема "примечание(...)". Вводится накопитель x10 программных выражений для безусловных

комментариев; условные комментарии регистрируются в накопителе, на который ссылается информационный элемент программного блока (примечание ...). По завершении цикла заполнения этих накопителей - переход через "ветвь 2", где переменной x11 присваивается программное выражение для объединенного списка комментариев. К списку x8 добавляется элемент (примечание x11), который впоследствии будет передан реализующей вывод процедуре. Затем - откат к переходу через "ветвь 1".

Вводится накопитель x11 информационных элементов, передаваемых реализующему вывод оператору. В него включаются списки x8, x9, и заносится ссылка (прием ...) на текущий прием. Далее выполняются следующие действия, связанные с учетом указателей приема:

1. Учет указателя "прообраз(i посылки($A_1 \dots A_n$) вывод($B_1 \dots B_m$))" (контрольная точка "прием(50)"). Прием регистрирует в текущей задаче информацию о том, что идентифицированное с i-м антецедентом утверждение задачи является следствием консеквента, прочих использованных при выводе утверждений, а также дополнительных следствий $B_1, \dots, B_m$, выводимых после установления истинности с помощью проверочных операторов утверждений $A_1, \dots, A_n$.

   Переменной x15 присваивается вхождение рассматриваемого антецедента в теорему приема; переменной x17 - программное выражение для этого антецедента. Далее в x11 заносится программное выражение для элемента (прообраз C D). C - посылка, идентифицированная с рассматриваемым антецедентом. Если D не равно 0, то оператор "вывод" должен будет ввести комментарий (прообраз C R), где R - объединение утверждений списка D, остальных использованных посылок и нового утверждения; C является следствием R. При построении программного выражения для элемента (прообраз ...) сначала отдельно рассматривается простейший случай $m = n = 0$. Затем - переход через "иначе 3", где переменной x18 присваивается выражение для заготовки элемента (прообраз C D). Если $n = 0$, то D полагается равным символу "пустоеслово", иначе - 0. Если $n > 0$, то компилируется оператор, проверяющий истинность всех $A_1, \dots A_n$, и в случае положительного результата изменяющий D на список использованных для проверки утверждений. Если $m > 0$, то компилируются операторы, осуществляющие вывод дополнительных следствий $B_1, \dots, B_m$, проверяя отличие D от 0, а также оператор, регистрирующий эти следствия в списке D. Затем программное выражение x18 заносится в список x11.

2. Учет указателей "вспомпараметр(...)", "новыйсимвол(...)", "вспомнеизвестная(...)" (контрольная точка "прием(70)").

   Указатель "вспомпараметр(x V)" позволяет вводить новый вспомогательный параметр x задачи на исследование, имеющей цель "известно". Утверждения с x, вхождения которых в теорему задаются указателями списка V, переносятся в список посылок внешней задачи на описание.

   Указатель "новыйсимвол(x V)" аналогичен предыдущему, но новая переменная x становится неизвестной задачи на исследование.

   Указатель "вспомнеизвестная(x)" означает, что вводимая приемом новая переменная x становится вспомогательной неизвестной как текущей задачи на исследование, имеющей цель "известно", так и внешней задачи на описание.

Вспомогательные параметры задачи считаются известными и регистрируются в комментарии (вспомпараметр ...). Включение их в ответ внешней задачи на описание блокируется. Вспомогательные неизвестные регистрируются в комментарии (вспомнеизвестная ...). Найденные значения их не включаются в ответ внешней задачи на описание.

Во всех перечисленных случаях переменной x17 присваивается программное выражение для x. Далее в накопитель x11 заносится программное выражение для набора (Q x T) либо (случай "вспомнеизвестная") набора (Q x). Здесь Q - заголовок указателя; T - терм "набор(...)", перечисляющий утверждения, адресуемые списку посылок внешней задачи.

3. Учет указателей, задающих новые элементы чертежа (контрольная точка "прием(23)"). В x11 заносится программное выражение для набора (геомредактор набор($A_1 \dots A_n$)), где A_i - программные выражения, определяющие данные указатели.

4. Учет указателя "титр(...)", определяющего выдачу сопровождающих срабатывание приема пояснений. В x11 заносится информационный элемент, определяемый процедурой "транститр".

5. Указатель приема "внешзнак", блокирующий уменьшение веса условия задачи на преобразование, переносится в набор x11.

6. Указатель приема "стоп" инициирует ввод в компилируемую программу оператора "трассировка(стоп 0)" для выхода в отладчик ЛОСа.

7. Учет указателя "коррекцияодз" (контрольная точка "прием(51)"). Точка привязки приема определяется указателем "контрольвывода(A)", а выводимое утверждение нужно для восстановления сопровождения терма A по о.д.з. Чтобы зарегистрировать его конъюнктивные члены в комментариях "сопровождение", в компилируемую программу вставляется обращение к процедуре "коррекцияодз(...)".

8. Учет информационных элементов программного блока (удалениепосылки ...), (удалениеусловия ...). По ним создаются одноименные элементы набора x11.

9. Учет указателя приема "новыенеизвестные" (контрольная точка "прием(69)"). Новые переменные регистрируются в списке неизвестных задачи с помощью добавляемого к компилируемой программе оператора "учетнеизвестных(...)".

10. Учет указателя приема "примеч" (контрольная точка "прием(92)"). Так же, как это делалось для приемов замены.

11. Учет указателей "замечание(...)", "комментариипосылок(...)", не содержащих символа "результат". В компилируемую программу вводятся операторы, создающие соответствующие комментарии.

Если в компилируемую программу уже были введены вспомогательные операторы "вывод(...)", изменяющие задачу, то к набору x11 присоединяется логический символ "выход", необходимый для предотвращения отката в случае отказа от

выполнения основного вывода приема. Тогда, по достижении завершающего программу приема оператора "пересмотр", произойдет повторный анализ текущей ситуации.

Наконец, программной переменной x12 присваивается выполняющий основное действие приема оператор "вывод(...)" либо "выводусловия(...)". После контрольной точки "прием(33)" он заносится в компилируемую программу.

После контрольной точки "прием(52)" рассматривается указатель приема "внимание" - по нему создается оператор, обнуливающий веса посылок и условий, содержащих заданный терм.

После контрольной точки "прием(63)" рассматривается указатель приема "найдено(X)" - по нему создается оператор, исключающий переменные списка X из числа неизвестных текущей задачи.

После контрольной точки "прием(77)" рассматривается указатель приема "актив" - по нему создается оператор, уменьшающий вес заданного утверждения задачи до заданной величины.

Наконец, программа завершается добавлением оператора "контроль" (если в x11 был занесен символ "выход", то задача могла остаться неизмененной, и тогда откат к началу цикла сканирования не нужен) либо "пересмотр".

Приемы "подборзначений", "существует"

Подслучай рассматривается после контрольной точки "прием(53)". Напомним, что прием "подборзначений" предпринимает попытку неэквивалентной замены условия или группы условий задачи на описание, имеющей цель "пример". Эти условия идентифицируются с консеквентом теоремы и заменяется на группу антецедентов, выделенных указателями "подборзначений(...)". Фактически замена выполняется во вспомогательной задаче; если ее решить не удается, то продолжается решение текущей задачи. Прием "существует" представляет собой версию приема "подборзначений", в которой замене подлежит группа всех условий, содержащих заданные несущественные неизвестные. Консеквент теоремы приема здесь имеет вид квантора существования по переменным, идентифицируемым с такими неизвестными.

Переменной x6 присваивается список антецедентов теоремы приема; переменной x7 - подмножество антецедентов, выделенных указателями "подборзначений(...)". Находится программное выражение x8 для конъюнкции утверждений списка x7. Инициализируется накопитель x13 программных выражений для информационных элементов, передаваемых процедуре, выполняющей основное действие приема. В него помещаются: ссылка на прием и информация об использованных приемом посылках; информация для выдачи поясняющего срабатывание приема текста; информация о комментариях, передаваемых вспомогательной задаче. Если заменяется единственное условие, применяется процедура "попыткаспуска(...)", иначе - процедура "Попыткаспуска(...)".

Приемы групповой замены и прием "параметризация"

Подслучай рассматривается после контрольной точки "прием(54)". Приемы групповой замены имеют заголовки "заменатермов", "заменаусловия" и "связка". Соответственно, происходят замены следующих типов: замена группы посылок; замена группы условий; замена всех условий, содержащих заданные несущественные неизвестные, сопровождаемая исключением этих неизвестных. В первых двух случаях

преобразования эквивалентные; во втором - заменяющее утверждение эквивалентно существованию значений исключаемых неизвестных, при которых заменяемые условия истинны.

Прием "параметризация" основан на эквивалентности $\forall_x(A_1\ \$\ldots\ \&\ A_n \rightarrow (B \leftrightarrow \exists_y(B_1\ \&\ldots\ \&\ B_m)))$, дающей параметрическое описание для одного либо нескольких условий B задачи на описание. Он вводит вспомогательную задачу, получаемую заменой данных условий на конъюнктивные члены $B_1,\ldots,B_m$) параметрического описания и присоединением параметров описания y к списку неизвестных. По окончании решения вспомогательной задачи на ее ответ навешивается квантор существования по параметрам, и выдается результат упрощения полученного утверждения. Заметим, что прием "связка" имеет теорему такого же вида, но у него квантор существования расположен в левой части эквивалентности, y - список исключаемых несущественных неизвестных, а $B_1,\ldots,B_m$ - все условия задачи, их содержащие. В случае приемов групповой замены основное действие реализуется процедурой "заменагруппы"; в случае приема "параметризация" - процедурой "попыткапараметризации".

Переменной x6 присваивается вхождение консеквента теоремы - он имеет вид эквивалентности. Переменной x7 присваивается заменяющий терм этой эквивалентности. Если прием имеет указатель "новыенеизвестные" (контрольная точка "прием(93)"), то для регистрации перечисляемых в нем переменных как новых неизвестных текущей задачи в компилируемую программу вводится обращение к оператору "учетнеизвестных". Инициализируется накопитель x11 программных выражений для информационных элементов, которые будут передаваться процедуре, реализующей основное действие приема. В него заносятся ссылка на прием и информация об утверждениях, использованных при срабатывании приема. Далее x11 пополняется в зависимости от следующих указателей приема:

1. Указатель "примечание" порождает одноименный информационный элемент в x11, определяющий комментарии, которыми снабжаются заменяющие утверждения;

2. Указатель "удалениеусловия" передает в x11 информацию о сопровождающем основное действие удалении условий;

3. Указатель "новыйсимвол" передает в x11 информацию о новых переменных, вводимых приемом и требующих учета во внешней задаче на описание (в этом случае текущая задача - на исследование);

4. Указатель "титр" передает в x11 информацию о поясняющем срабатывание приема тексте;

5. Информационный элемент программного блока "ключвывода" передает в x11 информацию о выводе сопровождающих утверждений;

По окончании этих действий (контрольная точка "прием(84)") компиляция разветвляется. Здесь рассматриваются два случая:

1. Случай приема групповй замены. Находится программное выражение x12 для заменяющего терма. Если прием имел указатель "примеч(...)", то он обрабатывается так же, как и выше, порождая операторы, вводящие необходимые комментарии. Переменной x13 присваивается программное выражение для набора

заменяемых термов задачи. Если программное выражение x12 для заменяющего терма содержит оператор "облвхожд(...)" (при задании списков посылок пакетных операторов или вспомогательных задач), то он преобразуется в оператор "Облвхожд(...)" - для исключения из посылок заменяемых утверждений. Такая же замена предпринимается для остальных вхождений оператора "облвхожд" в компилируемую программу. Лишь после этого добавляются завершающие программу приема операторы "заменагруппы(...)" и "пересмотр".

2. Случай приема параметризации. Проверяется, что заменяющая часть теоремы приема имеет своим заголовком квантор существования. Программной переменной x12 присваивается связывающая приставка этого квантора (т.е. список параметров). В компилируемую программу вставляются операторы, выбирающие для теоремных переменных списка x12 новые переменные, не использованные в контексте срабатывания приема. Программные переменные для этих переменных накапливаются в списке x14. Переменной x15 присваивается набор программных выражений для утверждений, образующих параметрическое описание. Переменной x16 присваивается набор программных выражений для заменяемых условий. Наконец, переменной x17 присваивается обращение к процедуре "попыткапараметризации". Это обращение и оператор "ответ" заносятся в компилируемую программу.

Прием ввода либо исключения комментариев

Подслучай рассматривается после контрольной точки "прием(55)". Прием имеет заголовок "замечание". Компиляция действий с комментариями к данному моменту уже выполнена - она вынесена в предварительный цикл обработки указателей приема. Поэтому обрабатываются лишь остаточные указатели "найдено", "актив", "пересмотр". Эти действия аналогичны рассмотренным выше. При отсутвии указателей "новоеусловие", "новаяпосылка", "внимание", "пересмотр" программу приема завершает не оператор "пересмотр", а оператор "продолжение", так как изменены были только комментарии.

Прием "замещениеусловий"

Подслучай рассматривается после контрольной точки "прием(56)". Теорема приема имеет вид "замещениеусловий(A)" и определяет вид A посылки текущей задачи на исследование. Действие приема состоит в добавлении этой посылки к списку условий внешней задачи на описание. При этом предпринимается попытка усмотреть те "старые" условия задачи на описание, которые после добавления A становятся избыточными - как следствия A, прочих условий и некоторых сопровождающих A посылок, тоже переносимых в условия внешней задачи. Избыточные условия удаляются. Все эти преобразования выполняются процедурой "замещениеусловий".

В компилируемую программу заносятся операторы: проверяющий, что идентифицированная с A посылка еще не рассматривалась данным приемом; вводящий комментарий для блокировки повторного рассмотрения приемом этой посылки; проверяющий, что посылка не входит в список условий внешней задачи; реализующий обращение к процедуре "замещениеусловий". Если прием имеет указатель "обрывзадачи", то в безусловном случае далее помещается оператор "ответ", инициирующий возвращение к внешней задаче, а в условном - программа для проверки содержаще-

гося в указателе фильтра, и далее - оператор "ответ". При отстутствии указателя "обрывзадачи" программа приема завершается оператором "продолжение".

Прием "обозначение"

Подслучай рассматривается после контрольной точки "прием(58)". Теорема приема имеет вид "равно(x A)". Прием вводит вспомогательную неизвестную x для выражения A, усмотренного в условии задачи на описание. Все вхождения этого выражения в условия заменяются на x; вводится комментарий (вспомпараметр x), и заносится новое условие $x = A$. Оно сопровождается комментариями, блокирующими обратную замену A на x. Приемы такого типа применяются при решении задач, подготавливающих компилятору ГЕНОЛОГа схему вычислений. Компилятор вставляет в программу приема обращение к процедуре "обозначение", выполняющей перечисленные выше действия, после которого размещает оператор "пересмотр".

Прием "результат"

Подслучай рассматривается после контрольной точки "прием(61)". Теорема приема имеет вид "результат(A)". Если усматривается, что условие текущей задачи на преобразование имеет вид A, причем истинны фильтры приема, то происходит немедленная выдача условия задачи в качестве ответа. Так как все проверки к данному моменту компиляции уже обеспечены, то остается только ввести завершающий оператор "ответ(...)".

Прием "свертка"

Подслучай рассматривается после контрольной точки "прием(32)". Теорема приема - кванторная импликация, консеквент которой сам является кванторной импликацией K. Предпринимается попытка решить задачу на доказательство путем идентификации ее условия с консеквентом K, а части посылок - с антецедентами K. Приемы такого типа используются при доказательстве по индукции. К текущему моменту компиляции уже проведены идентификация и выполнены все проверки антецедентов теоремы. Таким образом, истинность доказываемого утверждения приемом уже установлена, и перед выдачей ответа "истина" необходимо учесть информацию об использованных посылках. Это обеспечивается обращением к процедуре "учетсвертки". Предварительно заполняется накопитель x7 программных выражений для ссылки на прием и информационных элементов (выводимо ...).

Прием "выразимо"

Подслучай рассматривается после контрольной точки "прием(82)". Прием используется в задачах на описание, составляющих схему вычислений для последующей обработки ее компилятором ГЕНОЛОГа. Он усматривает выразимость некоторых переменных через исходные данные задачи такими стандартными средствами, которые, в принципе, позволяют вычислить эти переменные (например, построить таблицу значений функции по дифференциальному уравнению и начальным значениям). Информация о выразимости регистрируется в комментариях задачи процедурой "выразимо(...)". Если установлена выразимость в указанном смысле всех неизвестных задачи, вводится цель "программа", определяющая этап технической обработки ус-

ловий перед компиляцией. Так, в указанном выше примере, будет выполнен переход от дифференциальных уравнений к соответствующим конечно-разностным.

Вводятся накопители x10, x11 единичных переменных и списков переменных, выделенных в приеме указателем "выразимо". После их заполнения создается обращение к процедуре "выразимо".

19.19 Завершающая обработка программы

Завершающая обработка программы выполняется двумя последовательно применяемыми процедурами - "вставкафрагментов" и "завершениепрограммы". Обращение к первой из них имеет вид "вставкафрагментов(x1)", где x1 - программный блок. Здесь происходит извлечение из информационных элементов (блокпрограммы ...) фрагментов программы, созданных в процессе компиляции, но пока не зарегистрированных в накопителе программ блока x1, и занесение этих фрагментов в данный накопитель. Обращение ко второй имеет вид "завершениепрограммы(x1 x2 x3 x4 x5)". Здесь x1 - пара (K F), содержащая набор F фрагментов программы приема, и некоторый набор K сопровождающих комментариев (такую пару называем на этапе редактирования программы программным модулем); x2 - логический символ, за которым закреплен компилируемый прием; x3 - номер узла теоремы этого приема; x4 - заголовок приема; x5 - описание приема. Предпринимается пополнение фрагментов программы и завершающее их редактирование. Последнее заключается в последовательных эквивалентных упрощающих преобразованиях программы. Заметим, что в действительности процедура "завершениепрограммы" почти сразу обращается к описываемой ниже процедуре "редакцияпрограммы", которая и выполняет основную часть работы.

19.19.1 Процедура "вставкафрагментов"

Происходит просмотр списка фрагментов программы в программном блоке x1; x2 - текущая позиция этого списка. Для текущего фрагмента x3 выполняется просмотр его операторов; x4 - текущая позиция оператора x5. Если оператор имеет вид "1(A n)" либо "2(n)", то он представляет собой ссылку на фрагмент программы P, сохраненный в информационном элементе (блокпрограммы n P) программного блока x1. Фрагмент P заносится в конец просматриваемого списка фрагментов программного блока x1; текущий оператор заменяется в первом случае на два оператора - A, "иначе(k)", а во втором случае - на один оператор "ветвь (k)". Здесь k - номер фрагмента P в просматриваемом списке. Увеличивается на единицу счетчик фрагментов программного блока x1. Затем - откат к повторному просмотру списка фрагментов. Если при просмотре никаких изменений программы не произошло, то выход из процедуры.

19.19.2 Процедура "завершениепрограммы"

Эта процедура выполняет одно-единственное преобразование, после чего обращается к процедуре "редакцияпрограммы". Преобразование связано с достаточно редким случаем компиляции приема нормализатора, работающего в режиме разбора случаев. Примером такого нормализатора может служить оператор "нормпредел", вычисляющий пределы.

Переменной x6 присваивается заголовок нормализатора и проверяется, что в описании его формата имеется элемент "разборслучаев". Если программа приема данного нормализатора имеет оператор "равно(x P)", где P - рекурсивное обращение к тому же самому нормализатору x6, то результатом обращения может оказаться терм T вида "разборслучаев(...)", указывающий на необходимость немедленного разбора случаев. В этой ситуации нужно, во-первых, вернуться по цепочке рекурсивных обращений к корневому обращению и, во-вторых, откатиться для него к той точке в начале программы, где организуется разбор подслучаев согласно T. Поэтому выполняется просмотр программы с целью поиска указанных операторов "равно(x P)". Для найденного такого оператора после него в программу вставляются операторы "не(заголовок(x разборслучаев))", "иначе(n)", "метка(продолжение)". Здесь n - номер, на единицу больший числа фрагментов программы; оператор "метка(продолжение)" является указателем на возможность продолжения в данной точке выделения общего начала двух склеиваемых программ приемов (если таковое понадобится), несмотря на наличие оператора перехода "иначе(n)". Затем к концу списка фрагментов программы присоединяются: фрагмент с операторами "входит(подчинено x3)", "иначе($n+1$)", "ответ(x)", а также фрагмент с операторами "замена(x6 0)", "подслучаи(x1 x2 x3 x)", "переходпометке(1)". Первый из них решает задачу выхода из промежуточной точки цепи рекурсивных обращений; второй - регистрирует информацию о разборе случаев в комментариях нормализатора и организует откат к точке инициализации разбора случаев согласно измененным комментариям, уменьшая предварительно до нуля текущий уровень сканирования x6.

19.19.3 Процедура "редакцияпрограммы"

Обращение к процедуре имеет вид "редакцияпрограммы(x1 x2)". Здесь x1 - набор комментариев к списку x2 фрагментов редактируемой программы. При компиляции приема сканирования задачи в список x1 передается единственный комментарий (заголовок A) с заголовком компилируемого приема. Другие возможные комментарии - (связприставка A) со списком A программных переменных, которые необходимо сохранить при редактировании, и (переменная x), где x - программная переменная, встречающаяся в операторах, вставляемых в начальный отрезок программы после редактирования. Они блокируют действия, связанные с изменением номеров указанных в них программных переменных.

Процедура "редакцияпрограммы" сравнительно сложна, и для выхода в различные ее точки удобно использовать ветвь "Завершающая обработка программы", "Завершающее редактирование программы" оглавления программ.

Исключение псевдооператоров "переходпометке", "меткаперехода"

При компиляции в программу могли заноситься псевдооператоры "меткаперехода(n x)", "переходпометке(n)". Второй из них задает откат к первому, причем x является первой программной переменной, не определенной после отката. Эта ситуация, впрочем, случается сравнительно редко, так как обычно программа приема имеет линейную структуру. Чтобы программа на ЛОСе фактически могла реализовать откат, нужно заменить псевдооператоры "переходпометке" на соответствующие операторы "продолжение", "обрыв", "сброс(i)", а псевдооператоры "меткаперехода" удалить. С этой целью реализуется цикл просмотра фрагментов x4 программного модуля x2, а для текущего фрагмента - просмотр операторов x6. Как только обнаруживается

оператор "переходпометке(n)", символьный номер перехода n присваивается переменной x7 и начинается просмотр цепочки надфрагментов текущего фрагмента программы. При этом заполняется накопитель x8 пар (вхождение надфрагмента в список фрагментов - вхождение последнего оператора этого надфрагмента, из которого достижим оператор x6).

В некоторый момент должен встретиться псевдооператор x12 вида "меткаперехода(n x)", обозначающий точку отката. Тогда инициализируется нулем счетчик x14 числа перечисляющих операторов на пути от точки отката до x6 (контрольная точка "прием(3)"), и начинается цикл просмотра операторов данного пути. Последовательно просматриваются вхождения x15 пар накопителя x8. Переменной x16 присваивается вхождение начального оператора отрезка пути в фрагменте пары x15, переменной x17 - вхождение последнего оператора этого отрезка. Переменная x18 перечисляет вхождения от x16 до x17 (не включая x17). Если встречается оператор "Обрыв", то из x14 вычитается единица, так как выполнение его приводит к отбрасыванию последнего перечисления. Если встречается перечисляющий оператор, то x14 увеличивается на 1. Заметим, что при определении типа оператора используется первая не определенная перед его выполнением программная переменная x13, которая инициализируется по "меткаперехода(n x)" и корректируется при продвижении вдоль пути. Если оператор заведомо не перечисляющий, то x14 не изменяется. Компилятор избегает вводить в программу операторы, работающие в смешанном режиме (проверка - перечисление). Это, однако, может произойти для операторов вида "альтернатива(A равно(y t) B)", где y - первая не определенная переменная, B - перечисляющий оператор. Тогда компилятор заменяет обычное присвоение "равно" на перечисляющее присвоение "Равно", переводя таким образом оператор в разряд перечисляющих. В прочих случаях, встретившись с оператором смешанного типа, компилятор выходит в отладчик ЛОСа на фрагмент аварийного прерывания "повторение трассировка(стоп 0) продолжение".

По окончании просмотра пути (контрольная точка "прием(5)") предпринимается изменение оператора x6 на "продолжение" в случае x14 = 0; "обрыв" в случае x14 = 1, и "сброс(x14)"в прочих случаях.

Далее цикл поиска псевдооператоров x6 вида "переходпометке(...)" продолжается. По его завершении (контрольная точка "прием(6)") реализуется цикл исключения из программы псевдооператоров "меткаперехода(...)".

Удаление повторного оператора

Вообще говоря, в одном и том же фрагменте программы могут встречаться одинаковые операторы. Если между ними встречается хотя бы один перечисляющий оператор, то возможен откат, после которого значения входных переменных второго оператора окажутся иными, чем они были у первого. Однако, если одинаковые операторы встречаются сразу друг за другом, то один из них может быть удален. Это и делается ветвью, расположенной после контрольной точки "прием(7)". Здесь x5 - повторное вхождение оператора; x7 - его предыдущее вхождение. После удаления оператора указатель текущего вхождения в измененный фрагмент программы корректируется так, чтобы просмотр был продолжен с той же точки.

Использование справочника "редакцияпрограммы"

Многие действия по упрощению программы вынесены в приемы справочника "редакцияпрограммы". При обращении к нему определены следующие переменные: x1 - набор комментариев к списку x2 фрагментов редактируемой программы; x3 - текущее вхождение в x2; x4 - текущее вхождение в фрагмент x3; x5 - оператор по вхождению x4; x6 - вхождение текущего логического символа в x5 (он является символом обращения к справочнику); x7 - первая не определенная перед выполнением оператора x5 программная переменная. Если прием справочника изменяет программу, то результатом обращения служит пара (новое текущее вхождение фрагмента - новое текущее вхождение оператора), определяющая точку, с которой следует продолжать просмотр программы.

Значением переменной x3 служит вхождение текущего фрагмента x5 программы в список фрагментов; x4 - индикатор изменения фрагмента, изначально устанавливаемый на 0. x6 - вхождение текущего оператора x8 в x5; x7 - первая не определенная перед выполнением этого оператора программная переменная. Если обращение к справочнику дает ненулевой результат, то x4 заменяется на 1. После полного просмотра программы проверяется, равно ли x4 единице. Если равно, то цикл просмотра повторяется.

В качестве примера приведем несколько приемов справочника "редакцияпрограммы":

1. Символ обращения к справочнику - "символ" (см. фрагмент программы символа "символ", начинающийся с оператора "обращение(редакцияпрограммы)"). Если текущий оператор имеет вид "символ(левыйкрай(A)B)", то он заменяется на "заголовок(A B)"; если он имеет вид "символ(первыйоперанд(A)B)", то заменяется на "первыйсимвол(A B)", и т.п.

2. Символ обращения к справочнику - "операнд". Если текущий оператор - корневой и имеет вид "операнд(A x)", где x - не определенная перед обращением к оператору переменная, то находятся все достижимые из него операторы, содержащие x. Составляется список x9 ссылок на эти операторы; каждая ссылка есть пара (вхождение фрагмента - вхождение оператора в фрагмент). Если список x9 пуст, либо состоит из ссылки на оператор "не(равно(x B))", причем в последнем случае усматривается, что по вхождению A расположен более чем одноместный символ, то текущий оператор избыточен - он определяет не используемую переменную x. Поэтому реализуется удаление оператора. Оно сопровождается уменьшением на единицу номеров переменных, следующих после x, во всех операторах, достижимых из удаленного. Так как удаленный оператор - перечисляющий, то необходима также коррекция величин отката в достижимых из него операторах "обрыв", "сброс(...)". Первая коррекция выполняется процедурой "сдвигпеременных"; вторая - процедурой "коррекцияоткатов". Обе эти процедуры часто используются в других приемах справочника, а также в самой программе "редакцияпрограммы".

3. Символ обращения к справочнику - "вариант". Если текущее операторное выражение имеет вид "вариант(A t_1 t_2)", причем оператор A либо его отрицание встречались в программе до текущего оператора, то выражение заменяется, соответственно, на t_1 либо t_2. Чтобы данное преобразование было корректным, необходима проверка сохранения значений переменных оператора A на отрезке

от вхождения его (либо его отрицания) в программу до текущего оператора. Эти значения могли измениться при откатах либо после выполнения оператора "замена". Проверку сохранения значений выполняет процедура "сохрзначений", которая тоже достаточно часто используется при преобразованиях программы.

Не приводя здесь сколь-нибудь полного перечня приемов справочника "редакцияпрограммы", которых насчитываются многие десятки, ограничимся перечислением тех логических символов, для которых имеются эти приемы: "не", "и", "или", "длялюбого", "существует", "альтернатива", "вариант", "выч", "Выч", "конкатенация", "позиция", "символ", "переменная", "подчинено", "операнд", "левыйкрай", "суффикс", "сборка", "пересекаются", "запись", "пересечениесписков", "заголовок", "плюс", "вычитание", "подтерм", "равно", "входит", "параметры", "набороперандов", "истина", "ложь", "известно", "унисборка", "длинанабора", "числзначение", "возведениевстепень", "наборчленов", "точкипрямой", "Наборчленов".

Ввод вспомогательных переменных для повторяющихся программных выражений

Первоначальная версия программы приема обычно содержит множество повторяющихся вхождений одинаковых программных выражений. Естественно вычислить значение такого выражения один раз, присвоить его вспомогательной программной переменной, и обращаться к ней во всех последующих случаях. Заметим, что такое преобразование влияет на объем оптимизируемых программ наиболее существенным образом. При этом, однако, необходимо отслеживать возможность изменения значения выражения при откатах, переприсвоениях значений переменным и изменениях структур данных. Во всех сколь-нибудь подозрительных случаях такого рода компилятор должен избегать замен выражения на ранее вычисленное его значение. Приводимые ниже средства контроля допустимости ввода вспомогательного обозначения имеют, в общем, эвристический характер, достаточный для "типовых" программ, создаваемых текущей версией компилятора. По мере развития компилятора здесь, безусловно, понадобятся существенные дополнения.

Просмотр программы для выделения повторяющихся подвыражений начинается после контрольной точки "прием(10)". Инициализируется пустым словом накопитель x3, в который будут заноситься пары $(s\ A)$, где s - логический символ, A - набор пар $(t\ B)$. Здесь t - программное выражение, имеющее заголовок s; B - набор пар (вхождение фрагмента программы - вхождение оператора в фрагмент), адресующих содержащие t операторы. Переменной x4 присваивается список логических символов, операнды которых заведомо можно не рассматривать (например, из-за того, что они суть не операторные выражения, а операторы). Далее начинается цикл просмотра. Переменная x5 указывает вхождение текущего фрагмента x6; x7 - вхождение текущего оператора x8. Оператор "ответ" и оператор, следующий за оператором "заменавхождения", не рассматриваются. Блокируется также рассмотрение накопителей (выводимо ...) использованных приемом утверждений. Далее переменная x9 пробегает вхождения в текущий оператор x8 неоднобуквенных подтермов, игнорируя операнды символов набора x4, а также те операнды выражений "перечисление(...)", "выписка(...)", "вариант(...)", на которых расположены операторы. Пропускаются случаи присвоения переменной набора, который впоследствии может быть изменен.

Если текущая позиция x9 не отвергается, то для нее выполняется переход через "ветвь 6". Здесь проверяется, что текущий подтерм не имеет переменных, связанных

внешними кванторами и описателями. Переменной x10 присваивается его заголовок, и происходит предварительное отсечение подтермов по заголовку. Далее предпринимается попытка найти в x3 пару, начинающуюся с x10. Если такой пары нет, то она заносится в x3 - для текущего подтерма, и просмотр подтермов продолжается. Иначе найденная пара (s A) становится значением переменной x11. Переменной x12 присваивается текущий подтерм, и предприниается попытка найти в A пару, начинающуюся с x12. Если ее нет, то она регистрируется в A, и просмотр подтермов продолжается. Иначе начинается просмотр пар x14, ссылающихся на ранее найденные вхождения в программу подтерма x12. Если этот просмотр будет доведен до конца, и вспомогательная переменная для x12 не возникнет, то переход через "иначе 3", где в A будет занесена очередная ссылка на подтерм, и просмотр программы продолжится.

Рассмотрение пары x14 начинается с проверки того, что она задает вхождение оператора P, из которого достижим текущий оператор x8. После контрольной точки "прием(12)" находится первая не определенная перед выполнением P программная переменная x15. Находится список x16 свободных переменных программного выражения x12 и проверяется, что все они к моменту выполнения P уже определены. Если в x12 встречаются операторы "известно", "неизвестная" либо операторное выражение "неизвестные", применение которых в случае задачи на доказательство некорректно, то проверяется наличие в программе оператора, определяющего отличие типа задачи от "доказать". При отсутствии такого оператора создание вспомогательного обозначения может вывести подтерм x12 за рамки той области, где его использование корректно. Далее - переход через "ветвь 4".

Если в x12 имеется подвыражение с заголовком "первыйоперанд", "первыйтерм", и т.п., то нужны гарантии того, что операнд соответствующего вхождения v с нужным номером существует - иначе произойдет остановка программы по некорректному обращению. В этом случае проверяется, не расположено ли текущее вхождение x9 либо одно из ранее встречавшихся вхождений выделяемого подтерма внутри выражения "вариант(U ...)", где условие U проверяет наличие по вхождению v заданного символа. Тогда ввод вспомогательного обозначения блокируется, так как может вывести x12 из того контекста, где имелись гарантии корректного обращения. В благоприятном случае - переход через "ветвь 1".

Если в x12 входит подвыражение "списокпосылок(...)", причем некоторый предшествующий текущему оператор изменял список посылок, то ввод обозначения блокируется. Иначе - переход через "ветвь 1".

Особого контроля требует склейка повторяющихся выражений "выч(...)". Их значениями служат представления чисел в машинном формате, а для ускорения вычислений предусмотрена возможность изменять значения чисел, сохраняя ссылку на них. Это делается оператором "Выч(изменение ...)" причем оператор "Выч(программа ...)" тоже может вызвать аналогичные изменения. Находится список x17 переменных - левых частей присвоений, в правую часть которых входит x12, и выполняется проверка наличия операторов, изменяющих хотя бы одну из переменных данного списка. В подозрительных случаях склейка блокируется, иначе - переход через "ветвь 1".

После контрольной точки "прием(14)" расположена последняя проверка допустимости ввода вспомогательного обозначения. Устанавливается отсутствие выполняемого после P оператора, либо изменяющего набор, пересекающийся по своим параметрам с x12 (кроме переменных x1,x2,x3 для приемов сканирования задачи), либо изменяющего значение входящей в x12 переменной (кроме x2, x3 для приемов нор-

мализаторов).

После контрольной точки "прием(15)" начинается изменение программы. Предпринимается увеличение на единицу номеров всех программных переменных, начиная с переменной x15, в операторах, достижимых после оператора P. Таким образом, переменная x15 высвобождается для использования ее как обозначения выражения x12. При сдвиге номеров, выполняемом процедурой "сдвигпеременных", особо учитывается случай оператора "замена", первым операндом которого служит не переменная, а ее номер. После контрольной точки "прием(16)" выполняется коррекция операторов накопителя повторных вхождений x3, заключающаяся в таком же сдвиге их переменных. После контрольной точки "прием(17)" выполняется аналогичный сдвиг в выражении x12 номеров связанных переменных, которые (в отличие от свободных) могут быть не меньше x15.

Переменной x17 присваивается оператор "равно(x15 x12)", который будет использован для нахождения значения x12. Затем (контрольная точка "прием(18)") реализуется цикл просмотра всех операторов, достижимых из P, и замены в них вхождений подтермов x12 на переменную x15. Накопитель x18 заполняется тройками (вхождение фрагмента программы - вхождение оператора в фрагмент - старая (до замены на x15) версия этого оператора). После контрольной точки "прием(19)" выполняется аналогичное преобразование достижимых из P операторов в накопителе x3.

После контрольной точки "прием(20)" анализируется оператор, предшествующий P. Если он представлял собой точку выхода в отладчик ЛОСа, то место вставки оператора x17 выбирается перед ним. Переменной x19 присваивается фрагмент программы, полученный вставкой x17 в фрагмент оператора P; переменной x20 - вхождение в x19 вставленного оператора. После контрольной точки "прием(21)" выполняется коррекция ссылок из x3 на вхождения операторов в измененный фрагмент. Корректируются также значения x6, x7 для продолжения просмотра программы после ее преобразования. Наконец, после контрольной точки "прием(22)" старая версия фрагмента оператора P заменяется на x19. После этого накопитель x3 пополняется вхождениями в правую часть оператора присвоения x17, и откат к продолжению просмотра программы.

Сдвиг операторов присвоения к концу программы

Важной частью оптимизации программы по быстродействию является перемещение операторов присвоения, не выполняющих какой-либо работы по проверке условий на текущий контекст, ближе к концу программы приема. Возможно, в этом случае данный оператор вообще не нужно будет выполнять - если перед ним окажется отсекающий текущую ситуацию проверочный оператор. В основном, операторы "чистого" присвоения обеспечивают создание новых термов (например, заменяющего терма). Часто они содержат обращения к нормализаторам и вспомогательным задачам, вследствие чего достаточно трудоемки.

Началом данного цикла обработки программы служит контрольная точка "прием(23)". Переменная x3 пробегает вхождения фрагментов в их список; x4 - текущий фрагмент; x5 - вхождение текущего оператора x6 в x4. Для распознавания оператора "чистого" присвоения, т.е. такого, который всегда истинен и лишь определяет значения своих выходных переменных, служит справочник "новаяпеременная". Фактически приемы его созданы пока лишь для символов "равно", "альтернатива" и "операнд", но их хватает для обработки сколь-нибудь трудоемких операторов присвоения, создаваемых текущей версией компилятора. Справочник выдает набор x7

переменных, значения которых определяются оператором.

Если прием имеет заголовок "знач" (т.е. относится к техническому анализатору), то проверяется, не расположен ли оператор x6 раньше дизъюнкции, перечисляющей уровни сканирования. В этом случае - переход к следующему оператору. Иначе - переход через "ветвь 3".

Проверяется отсутствие в x6 подоператора "титрбуф(...)" - если он есть, то свобода перемещений оператора ограничена необходимостью перемещать в связке с ним сопровождающие обращения к "титрбуф". После контрольной точки "прием(25)" создается пара x8 индикаторов, которые будут корректироваться при анализе операторов, следующих после x6. Единичное значение первого из них указывает, что был пройден оператор, не являющийся оператором "чистого" присвоения. Перенесение имеет смысл лишь в случае, если был пройден хотя бы один такой оператор. Единичное значение второго означает, что уже был предпринят анализ возможности перенесения x6 через перечисляющий оператор.

Инициализируются переменные x9 и x10, которые будут указывать текущий оператор при поиске новой позиции оператора x6. x9 - вхождение текущего фрагмента просмотра; x10 - вхождение текущего оператора. Находится первая не определенная перед выполнением оператора x10 программная переменная x11. Откат к оператору "ветвь 1", расположенному после контрольной точки "прием(26)", означает конец цикла продвижения точки (x9,x10) вдоль программы. Для текущего оператора x12 по вхождению x10 выполняются последовательно следующие действия:

1. Если x12 имеет заголовок, принадлежащий списку "трассировка", "иначе", "Обрыв", "уровеньобращения", заменавхождения", "ответ", "повторение", "попытказамены", "расширениепосылок", "заменавхожений", "лимит", то обрыв просмотра;

2. Если после x12 расположен оператор "иначе", то также обрыв просмотра;

3. Если x6 имеет вид "равно(x t)", а x12 - вид "обл(...)", т.е. перечисляет утверждения U некоторого контекста, причем после этого перечисления встречается проверочный оператор "равныетермы(x ...)", то при перенесении x6 за x12 нужно будет многократно повторно вычислять t для каждого U, чтобы выполнить проверку условия "равныетермы(...)". Поэтому лучше оставить x6 "за скобками" и оборвать просмотр;

4. Если x12 имеет вид "ветвь N", а после него идет оператор "продолжение", то для продолжения просмотра линейной цепочки операторов выполняется переход к фрагменту N (контрольная точка "прием(27)");

5. Если x12 содержит переменные списка x7, определенные оператором x6, то обрыв просмотра;

6. Если x12 содержит символ "изменение", либо подтерм "Выч(запись ...)", либо подоператор "замена(...)", изменяющий значение переменной терма x6, то обрыв просмотра.

7. Если оператор x12 определяет значение какой-то новой переменной x13, которое впоследствии может быть изменено оператором x16, причем оператор x6 - перечисляющий, и между операторами x12, x16 используются определяемые им переменные, то обрыв просмотра (контрольная точка "прием(38)").

8. Если оператор x12 - перечисляющий и второй индикатор пары x8 еще не равен 1, то проверяется, что далее нет операторов, изменяющих переменные из x6 либо встречающиеся в x6 наборы, после чего данный индикатор устанавливается на 1. При неудаче - обрыв просмотра.

9. Выполняется сдвиг текущей позиции на единицу вправо. Корректируется первая не определенная программная переменная x11. Если усматривается, что оператор x12 был не "чисто" присваивающим, то первый индикатор пары x8 устанавливается на 1. Затем - откат к продолжению просмотра (оператор "повторение" после контрольной точки "прием(26)").

По завершении просмотра - переход через оператор "ветвь 1" после точки "прием(26)". Проверяется, что вхождение x10, перед которым предполагается разместить оператор x6, не расположено сразу за вхождением x5 этого оператора. Проверяется, что первый индикатор пары x8 равен 1. Проверяется, что если сразу после x6 фрагмент обрывается и продолжается через "ветвь" фрагментом x9, то x10 не является вхождением первого оператора фрагмента x9. После этого окончательно принимается решение о перенесении x6 на новое место (контрольная точка "прием(29)"). Находятся первая и последняя (по номерам) переменные x12, x13 списка x7 выходных переменных оператора x6. Переменной x14 присваивается длина списка x7 (номера переменных в нем идут подряд). Во всех операторах, достижимых из x10, номера их переменных, начиная с переменной x11 (первой не определенной перед выполнением оператора x10), увеличиваются на x14 - чтобы высвободить место для выходных переменных оператора x6, которые после перенесения его должны начинаться с x11. Затем предпринимается еще один цикл просмотра этих же операторов (контрольная точка "прием(31)", и номера всех переменных списка x7 выходных переменных оператора x6 увеличиваются на разность x11 и x12. В результате диапазон переменных x7 сдвигается так, что начинается с переменной x11. После перехода через "иначе 1" - находится результат x17 аналогичной перенумерации выходных переменных самого оператора x6. Теперь можно во всех операторах, достижимых из оператора x6 (пока находящегося на старой позиции) уменьшить на x14 номера переменных, идущих строго после переменной x13, и таким образом устранить пробел, возникающий при перенесении.

Наконец, происходит вставка оператора x17 на позицию перед x10 и исключение позиции x5 текущего фрагмента x4. Затем корректируются значения x3, x4 и x5 (последнее устанавливается перед позицией, с которой был перенесен оператор), и откат к продолжению просмотра программы (см. оператор "повторение" в фрагменте с контрольной точкой "прием(23)".

Цикл исключения избыточных присвоений одной переменной значения другой переменной

После контрольной точки "прием(33)" выполняется просмотр программы и поиск операторов вида "равно(x y)", где x - еще не определенная программная переменная; y - уже определенная переменная. Если такой оператор x6 найден, то просматриваются все достижимые из него операторы P, в которых переменная x либо используется, либо изменяется. Если между x6 и P не могло произойти изменение значений x, y, то в операторе P вхождения переменной x заменяются на y. Если не было найдено ни одного оператора P, не удовлетворяющего указанному условию, то оператор x6

исключается. Затем выполняется уменьшение на единицу в операторах, достижимых из точки удаления, номеров всех переменных, начиная с переменной x.

Повторный цикл обращений к справочнику "редакцияпрограммы"

После контрольной точки "прием(34)" реализуется повторный цикл обращений к справочнику "редакцияпрограммы", такой же, как и первый цикл.

Повторный цикл смещения операторов "чистого" присвоения

После контрольной точки "прием(36)" реализуеся повторный цикл смещения к концу программы операторов "чистого" присвоения, такой же, как и первый цикл.

Цикл исключения повторяющихся операторов

После контрольной точки "прием(37)" располагается цикл исключения избыточных повторных операторов. В отличие от приведенного выше цикла исключения повторяющихся операторов, здесь не обязательно операторы должны идти непосредственно друг после друга. Для проверки того, что между двумя последовательными вхождениями одного и того же оператора значения переменных оператора не изменялись, используется процедура "сохрзначений".

Цикл обращений к справочнику "перестановка" для перемещения операторов программы

После контрольной точки "прием(35)" выполняется цикл просмотра программы с обращениями для каждого оператора к справочнику "перестановка". Этот справочник служит, чтобы в некоторых (достаточно редких) случаях переместить оператор или группу операторов на другое место, не изменяя результата действий программы.

По завершении перечисленных преобразований выдается результат редактирования программы.

19.20 Запись программы приема в блок программ

Для записи программы приема в блок программ используется процедура "записьприема(x1 x2 x3 x4)". Здесь x1 - логический символ, в программу которого вставляется программа приема; x2 - тройка (f, n, c), где f - логический символ, за которым закреплен прием; n - номер узла теоремы приема; c - программное выражение, значением которого является заголовок приема. x3 - набор фрагментов программы откомпилированного приема; x4 - ссылка на узел приема. Кроме включения фрагментов x3 в программу символа x1, процедура создает ссылку на программу приема из его узла x4. Выйти на начало программы "записьприема" можно через пункт "Процедура регистрации программы приема в блоке программ" оглавления программ.

19.20.1 Ввод ограничителя склейки программ

При вставке программы приема в программу символа x1 будет происходить сравнение некоторой "старой" ветви программы с вставляемой ветвью, для выделения

возможно более длинного общего линейного начала обеих ветвей. Как правило, в это начало не должны попадать операторы, содержащие изменяемые при откатах переменные. Поэтому предпринимается просмотр фрагментов x3 для составления списка x6 всех переменных, значения которых могут изменяться при откатах. Из данного списка исключается часть заведомо изменяемых переменных, которые могут выноситься в общую часть. Например, в программе нормализатора таковыми являются переменные x1, x2. Значениями их служат преобразуемый терм и список посылок. После срабатывания приема данные переменные могут измениться, однако это не накладывает никаких ограничений на склейку начальных отрезков программ различных приемов.

После составления списка x6 - переход через "ветвь 3". Здесь выполняется просмотр программы вплоть до первого оператора, содержащего переменную списка x6. Перед ним вставляется фиктивный оператор "фикс". Он отсутствует в "старых" фрагментах программы, так что вынесение его в общую часть программ невозможно - этот оператор становится ограничителем допустимой области склейки. В дальнейшем, уже при непосредственной записи фрагмента в блок программ процедурой "записьпрограммы", ограничители "фикс" отбрасываются.

После вставки ограничителя "фикс" - откат к переходу через "ветвь 2".

19.20.2 Разрезание больших фрагментов

Если фрагмент программы очень большой, то он разрезается на части - хотя бы для того, чтобы фрагменты программ целиком отображались на экране. Анализ размеров фрагментов осуществляется после контрольной точки "прием(2)". Предварительно переменной x7 присваивается число фрагментов программы приема. Для текущего фрагмента x9 находится сумма x10 длин всех его операторов. Если она больше 280, то фрагмент считается подлежащим разрезанию на части. Находится список x11 длин операторов фрагмента x9. Он просматривается синхронно со списком x9 операторов фрагмента, и на каждом шаге определяется сумма x12 длин всех операторов, предшествующих текущему. Если она больше 260, а текущий оператор не является оператором перехода "иначе", то все операторы, расположенные до текущего (не включая его) выносятся в фрагмент x16; операторы начиная с текущего - в фрагмент x15. К концу фрагмента x16 добавляются операторы "ветвь x7", "продолжение"; фрагмент x9 заменяется на x16, а фрагмент x15 добавляется к концу списка фрагментов. Значение x7 увеличивается на единицу, и откат к повторному просмотру списка фрагментов.

По завершении цикла разрезаний фрагментов инициализируется нулем накопитель x6, в который будут заноситься несколько первых операторов программы приема, используемые в терминале "прием" узла приема для ускорения поиска ветви программы приема.

19.20.3 Инициализация программы логического символа

Если логический символ x1 вообще не использовался в блоке программ либо имел вырожденную программу, состоящую из единственного оператора "продолжение", то переход через "иначе 2" к контрольной точке "прием(3)". В этом случае вся программа символа x1 будет переопределена по списку x3 фрагментов откомпилированного приема. Предварительно в первый фрагмент данного списка включается ссылка на

прием - фиктивный оператор "контрольприема(f n c)". Значениями операндов служат, соответственно, логический символ, за которым закреплен прием, номер узла приема, и заголовок приема. Данный оператор обычно игнорируется интерпретатором ЛОСа, хотя в специальном режиме его программа может выполняться как обычная программа. Такой режим позволяет, например, анализировать частоту попаданий на начальные отрезки программы заданного приема и суммарной трудоемкости, приходящейся на эту программу. В случае приема сканирования задачи, распознаваемого по первому оператору "решить", вставка в программу оператора "контрольприема" предпринимается после оператора "уровень(...)". Иначе - он вставляется перед вторым оператором программы. В первом случае накопитель x6 заполняется цепочкой операторов, обрывающейся на операторе "уровень" (включая его); во втором - полагается состоящим из единственного первого оператора первого фрагмента. Собственно запись программы x3 в блок программ вместо старой версии программы символа x1 осуществляется процедурой "записьпрограммы(x1 0 x3 ...)". После записи - откат к контрольной точке "прием(10)", где создается логический терминал "прием" узла приема (см. ниже).

19.20.4 Случай приема сканирования задачи

После контрольной точки "прием(4)" в данном и всех оставшихся случаях пара (x7,x8) оказывается ссылкой на корень программы символа x1; x9 - первым оператором первого фрагмента новой программы.

Начиная с контрольной точки "прием(5)" рассматривается случай программы приема сканирования задачи. Она распознается по заголовку "решить" оператора x9. Находится корневой фрагмент x10 программы символа x1.

Если первый оператор фрагмента x10 отличен от "решить" (см. контрольную точку "прием(11)"), то ветвь программы символа x1, используемая при сканировании задачи, еще не создана. В этом случае после оператора "решить" в первый фрагмент F новой программы x3 вставляется оператор "иначе(x7 x8)", ссылающийся на корневой фрагмент x10 старой программы символа x1. После оператора "уровень(...)" в тот же фрагмент F вставляется фиктивный оператор "контрольприема(...)". Старая программа символа x1 отключается от каталога программ, но сохраняется в блоке программ. После этого выполняется обращение к процедуре "записьпрограммы" для регистрации набора фрагментов x3 в качестве новой программы символа x1. Наличие ссылки "иначе(x7 x8)" из x3 на старую ветвь приводит к тому, что эта ветвь снова становится подключенной к программе символа x1. Далее - откат к созданию логического терминала "прием".

Если же первый оператор фрагмента x10 есть оператор "решить", то переход через "иначе 4". Здесь переменной x11 присваивается первый фрагмент новой программы x3. Проверяется, что фрагменты x10 и x11 одновременно имеют либо не имеют оператора "стандарт(x2)". Это означает, что в задачах один и тот же логический символ не следует использовать как в виде символа операции или отношения, так и в виде константы. В противном случае приемы, точкой привязки которых будет служить данный символ, должны относиться к какому-либо одному из указанных вариантов. Определяется оператор x12 фрагмента x11, идущий сразу же после первых одного либо двух операторов "решить", "стандарт(x2)", "равно(x2 x3)" (такой оператор используется, если x1 - тип задачи) и проверяется, что он имеет вид "уровень($a_1 \ldots a_n$)". Переменной x13 присваивается набор термов $a_1, \ldots, a_n$ - уровней срабатывания приема.

Выбор точки присоединения новой программы

Чтобы определить точку присоединения новой программы к уже имеющейся ветви программ приемов сканирования задачи, выполняется просмотр "главного ствола" последней. Он начинается с контрольной точки "прием(14)". Предварительно создается буфер просмотра x14, куда будут заноситься ссылки (B_1 B_2 B_3 B_4 B_5) на подлежащие рассмотрению фрагменты программы символа x1. Здесь B_1 - набор номеров переходов в пути, ведущем от корня программы символа x1 к фрагменту; B_2 - ссылка на фрагмент; B_3 - ссылка на его надфрагмент (в корневом случае - 0); B_4 - такая позиция в корневом фрагменте новой программы, что все предшествующие ей операторы уже отождествлены с операторами старой программы, вдоль которых имело место перемещение к фрагменту B_2; B_5 - индикатор оператора "уровень(...)", пройденного при перемещении от корня программы символа x1 к B_2. Данный индикатор равен 0, если оператор "уровень" еще не встречался; равен 1, если его набор уровней строго шире, чем $a_1, \ldots, a_n$, и равен 2, если эти наборы совпадают. Сначала в x14 находится ссылка на корневой фрагмент программы. Кроме накопителя x14, инициализируется нулем также накопитель x15 ссылки на фрагмент, отобранный в качестве точки присоединения. Ему будет присвоен набор ($C_1 \ldots C_7$), где C_1, C_2, C_3 такие же, как B_1, B_2, B_3 у набора из x14; C_4 - сам отобранный фрагмент; C_5 - текущая позиция в нем, C_6 - текущая позиция в корневом фрагменте новой программы; C_7 - такое же, как B_5. Операторы до C_5, C_6 в "старой" и "новой" программах представляют собой их общую начальную часть.

После контрольной точки "прием(17)" вводится накопитель x16 наборов (D_1 D_2 D_3 D_4) ссылок на первые вхождения операторов "уровень($u_1 \ldots u_m$)" в просмотренные фрагменты программы символа x1. D_1 - путь к фрагменту от корня программы; D_2 - ссылка на фрагмент; D_3 - ссылка на внешний фрагмент либо 0; D_4 - набор (a $u_1 \ldots u_m$). Здесь a равно 0 тогда и только тогда, когда после оператора "уровень" не идет оператор "иначе", а до него в данном фрагменте не было операторов перехода. При занесении очередного набора в накопитель x16 его последний разряд D_4 сначала равен символу "пустоеслово", и лишь в процессе просмотра фрагмента заменяется на указанное выше значение.

Переменной x10, которой до этого был присвоен корневой фрагмент программы символа x1, переприсваивается 0. В цикле просмотра ее значением будет текущая позиция корневого фрагмента новой программы, сравниваемая с очередной позицией "старой" программы. Далее расположен оператор "повторение". При откате к нему находится первый элемент x17 буфера x14, здесь же исключаемый из буфера. Переменной x18 присваивается фрагмент, на который ссылается набор x17; x10 - исходная позиция в корневом фрагменте новой программы для дальнейшего сравнения старой и новой программ; x12 - сначала переприсваивается 0, а затем эта переменная будет хранить индикатор внешнего оператора "уровень(...)" (см. выше B_5); x9 переприсваивается 0, который после перехода через оператор "контроль-приема" заменяется на 1. Переменной x19 присваивается исходная позиция в фрагменте x18 для дальнейшего сравнения старой и новой программ. Находится число x20 операторов перехода, предшествующих в фрагменте x18 позиции x19. Оно будет корректироваться по мере перемещения x19 вглубь фрагмента, причем x21 сохранит исходное значение.

Далее размещается оператор "повторение", откаты к которому будут происходить после каждого сдвига позиций x19, x10 в "старом" и "новом" сравниваемых фрагментах программы. На очередном шаге сравнения операторов рассматривают-

ся следующие подслучаи:

1. На позиции x19 расположен оператор перехода "ветвь(m)" (контрольная точка "прием(15)").

   Если еще не был пройден оператор "контрольприема", то счетчик x20 числа операторов перехода увеличивается на 1; в накопитель x14 заносится ссылка на тот фрагмент F, к которому ведет данная "ветвь". Если фрагмент x18 упомянут в последнем наборе накопителя x16, пройденная "ветвь" - первая в нем с начала просмотра, а индикатор x12 внешнего оператора "уровень" равен 0, то ссылка на F регистрируется также в x16. После этого позиция x19 сдвигается на единицу вправо (x10 сохраняется), и откат к повторному сравнению позиций.

   Если оператор "контрольприема" уже был пройден, то после контрольной точки "прием(41)" анализируется частный случай - на позициях x19 и x10 находятся операторы "ветвь", а после них - завершающие фрагменты операторы "продолжение". Тогда организуется переход к просмотру фрагментов F_1, F_2, продолжающих "старый" и "новый" фрагменты. Из "старого" в начало F_1 переносится оператор "контрольприема". Накопитель x14 сбрасывается; переменной x17 переприсваивается набор аналогичного формата, ссылающийся на F_1; позиции x19 и x10 переустанавливаются на левые края фрагментов F_1 и F_2, причем отбрасывается первый фрагмент списка x3, так что F_2 становится его началом. Номера ссылок в x3 уменьшаются на единицу; в накопитель x15 заносится ссылка на левый край набора F_1, и откат к повторному сравнению позиций.

   Если указанный частный случай не имел места, то после контрольной точки "прием(18)" проверяется, что оператор "уровень" уже был пройден. Если это так, причем накопитель x15 либо не инициализирован, либо текущая позиция x10 расположена правее укаазанной в x15, то в качестве новой версии точки присоединения программы берется текущая версия. Здесь происходит откат к точке извлечения из x14 очередного элемента.

2. На позиции x19 расположен оператор "уровень($u_1 \dots u_m$)" (контрольная точка "прием(19)"). Переменной x22 присваивается набор ($u_1 \dots u_m$). Если текущий фрагмент представлен в накопителе x16 набором x23, причем последний разряд этого набора еще не изменялся (т.е. равен символу "пустоеслово"), то он заменяется на (a $u_1 \dots u_m$). Если до x19 встречались операторы перехода, либо непосредственно после x19 расположен оператор "иначе", то a равно 1, иначе - 0.

   Если список x13 уровней срабатывания нового приема включается в x22, то при строгом включении индикатору x12 присваивается 1, а при совпадении - 2. Позиция x19 сдвигается на единицу вправо; если на позиции x10 был размещен оператор "уровень", то она тоже сдвигается на единицу вправо. Затем - откат к повторному сравнению позиций.

   Если список x13 не включается в x22, причем первый элемент списка x22 меньше первого элемента списка x13, а непосредственно за x19 идет оператор "иначе n", то в накопитель x14 заносится ссылка на фрагмент n. Если фрагмент x18 упомянут в последнем наборе накопителя x16, левее x19 нет операторов перехода, а индикатор x12 внешнего оператора "уровень" равен 0, то ссылка на n регистрируется также в x16.

Далее, если x13 не включалось в x22, проверяется отличие индикатора x12 от 0. Если при этом накопитель выбранного фрагмента x15 равен 0, либо x10 расположено правее указанной в нем позиции, либо если эти позиции совпадают, но x12 равно 2, а соответствующее значение из x15 равно 1, то происходит переприсвоение накопителю x15 ссылки на текущий фрагмент x18. Затем - откат к рассмотрению очередного фрагмента из накопителя x14.

3. На позиции x19 расположен оператор "метка(...)" либо "контрольприема(...)". В последнем случае индикатор x9 прохождения оператора "контрольприема" заменяется на 1. В обоих случаях x19 сдвигается на единицу вправо, и откат к повторному сравнению позиций.

4. На позиции x19 расположен оператор "менее(u x5)", где u меньше наименьшего из уровней срабатывания нового приема. Тогда x19 сдвигается на единицу вправо, и откат к повторному сравнению позиций.

5. На позиции x19 расположен оператор "иначе n". Если оператор "контрольприема" еще не пройден, причем на позиции x10 нет оператора "иначе", то позиция x19 сдвигается на единицу вправо, и откат к повторному сравнению позиций. В прочих случаях - переход через "иначе 2". Если индикатор x12 пройденного оператора "уровень" отличен от 0, то предпринимается сравнение точки для вставки новой программы, отобранной в накопителе x15, с текущей точкой - так же, как и выше. Предварительно оператор, находящийся слева от позиции x10, сравнивается с оператором, находящимся слева от "иначе". Если они совпадают, то x10 сдвигается на единицу влево - так как при подключении программы данный оператор "иначе" пройден не будет. Далее - откат к рассмотрению очередного элемента накопителя x14.

6. На позициях x19 и x10 расположены одинаковые операторы, заголовки которых отличны от символов "лимит", "заменавхождения", и которые не содержат переменных, изменяемых последующими операторами новой программы. Если позиции x10, x19 - не концевые в своих фрагментах, то обе они сдвигаются на единицу вправо, и откат к повторному их сравнению. Иначе - при пройденном операторе "уровень" накопитель x15 переустанавливается на текущий вариант точки подключения программы; в обоих случаях далее выполняется откат к рассмотрению очередного элемента накопителя x14.

7. Оператор "контрольприема" еще не пройден, но оператор "уровень" уже пройден. На позициях x19 и x10 расположены операторы x22 и x23 с одинаковыми заголовками, причем непосредственно усматривается, что оба оператора - чисто проверочные и одновременно истинны быть не могут (контрольная точка "прием(31)"). Если после x19 идет оператор перехода "иначе n", то в x14 заносится набор, соответствующий фрагменту n, и откат к рассмотрению x14. Иначе - проверяется, пройден ли оператор "уровень", и если пройден, то x15 переустанавливается на текущую точку.

8. Остаточный случай - проверяется, пройден ли оператор "уровень". Если пройден, то выполняется выбор между старой версией точки присоединения x15 и новой, определяемой текущими значениями x19, x10. В любом случае далее - откат к выбору очередного элемента списка x14.

Вставка новой программы начиная с выбранной точки

После того, как список x14 исчерпан, происходит переход через оператор "иначе 1" перед контрольной точкой "прием(13)". Если накопитель точки вставки x15 отличен от 0, то реализуется вставка новой программы, иначе - переход к описываемому ниже циклу повторного выбора точки вставки. Заметим, что после выбора точки вставки могут быть отброшены несколько первых фрагментов новой программы x3 - если они полностью совпали с пройденной частью старой программы.

После контрольной точки "прием(40)" анализируется оператор старой программы, предшествующий точке вставки. Если он имеет вид "ветвь n", т.е. в этой точке уже была вставлена какая-то ветвь, то выполняется продвижение вглубь программы: находится первый оператор фрагмента n; если он имеет вид "ветвь m", то осуществляется переход к m, и т.д. При каждом переходе набор x15 корректируется согласно данному продвижению. По окончании продвижения - откат к переходу через "ветвь 2".

После контрольной точки "прием(34)" переменной x17 присваивается отобранный для вставки фрагмент; переменной x18 - та его позиция, перед которой нужно выполнить вставку. Переменной x19 присваивается заголовок оператора перехода, через который будет подключаться ветвь новой программы. Сначала он полагается равным символу "ветвь".

После контрольной точки "прием(35)" происходит сравнение оператора x20, расположенного на позиции x18, с тем оператором x21 новой программы, который будет начинать вставляемую ветвь. Если усматривается, что оба оператора - чисто проверочные и одновременно истинными быть не могут, то заголовок x19 оператора перехода изменяется на "иначе", а позиция x18 сдвигается на единицу вправо. Далее - откат к переходу через "ветвь 2".

Если обнаруживается, что оператор "контрольприема" в фрагменте x17 располагается до точки вставки, то он переносится на позицию, непосредственно предшествующую позиции x18, и выбирается в качестве новой точки вставки. Затем - откат к переходу через "ветвь 1".

После контрольной точки "прием(37)" находится набор x20 операторов первого фрагмента новой программы, начинающихся с указанной в x15 текущей позиции (предшествующие операторы были идентифицированы с операторами старой программы). В начале набора x20 вводится оператор "контрольприема(...)", ссылающийся на новый прием. Если указанный в наборе x15 индикатор прохождения оператора "уровень" отличен от 2, то предшествующие операторы "уровень" имели более широкий список допустимых уровней, чем нужно новому приему. В этом случае к началу набора x20 добавляется также "собственный" оператор "уровень(...)" нового приема. Далее начало списка x3 вставляемых фрагментов заменяется на x20. Номера переходов в фрагментах списка x3 увеличиваются на 1, а к началу списка присоединяется фрагмент x17, в котором перед позицией x18 вставлен оператор перехода к фрагменту x20. Для замены старой ветви фрагмента x17 на новую, определяемую списком x3, применяется процедура "записьпрограммы". Создается список x6 нескольких первых операторов программы (начиная с корня программы символа x1), который будет далее использован в логическом терминале "прием" узла приема. Наконец, с помощью процедуры "коррекцияссылок" предпринимается изменение предшествующих точке вставки пар операторов "уровень($u_1 \dots u_n$)", "иначе m", если их список уровней пересекается со списком x13 уровней нового приема. Каждая такая пара заменяется на "ветвь m", "уровень($u_1 \dots u_n$)". Затем - откат к контроль-

ной точке "прием(10)" для создания терминала "прием".

Повторный цикл выбора точки присоединения

В описанном выше цикле просмотра фрагментов программы символа x1 точка присоединения создавалась лишь после прохождения через оператор "уровень". Если такой ситуации не возникло, то реализуется еще один цикл просмотра фрагментов (контрольная точка "прием(38)"). Так как уровни срабатывания нового приема существенно отличны от уровней срабатывания приемов, ранее зарегистрированных в программе символа x1, то здесь уже не будет предприниматься попытка выделения общей начальной части программ. Анализироваться будут только имеющиеся в старой программе операторы "уровень", так как вставку новой ветви придется делать около одного из них - либо через оператор "ветвь", идущий перед оператором "уровень", либо через оператор "иначе", идущей непосредственно после "уровня".

Выполняется просмотр наборов x18 накопителя x16, в которых представлены первые вхождения операторов "уровень($u_1 \dots u_m$)" в фрагменты программы символа x1, достижимые без прохождения через прочие операторы "уровень".

Такие фрагменты относятся к части программы, которую будем называть ее "главным стволом". Фактически в x16 представлена лишь часть главного ствола - переходы через "иначе" после операторов "уровень", наименьшее значение которых оказывалось больше наименьшего уровня нового приема, блокировались. Некоторые фрагменты "главного ствола" могут начинаться с оператора "менее(U x5)", отсекающего случаи, в которых текущий уровень не превосходит значения U. За исключением этих ускорителей (вставляемых, как указано ниже), в начале фрагмента главного ствола расположен либо оператор "ветвь", после которого идет оператор "уровень", либо оператор "уровень", после которого может идти "иначе". Списки уровней при прохождении главного ствола упорядочены по их первым элементам.

Просмотр списка x16 ведется в направлении от конца к началу. Находится тот набор, у которого u_1 не больше наименьшего уровня срабатывания нового приема. Переменной x19 присваивается фрагмент, содержащий данный оператор "уровень". Далее рассматриваются два случая:

1. x18 является концевым оператором на главном стволе - перед ним не идет "ветвь", и после него не идет "иначе" (контрольная точка "прием(49)"). Если наборы уровней оператора x18 и нового приема не пересекаются, причем до x18 не было оператора "менее(u x5)", у которого u не меньше наименьшего уровня срабатывания нового приема, то после x18 размещается оператор "иначе(2)", в противном случае - перед x18 вставляется оператор "ветвь(2)". После такой коррекции фрагмента x19 - откат к переходу через "ветвь 3".

   Здесь в первый фрагмент новой программы сразу после его оператора "уровень" вставляется оператор "контрольприема". Номера ссылок внутри списка фрагментов x3 увеличиваются на 1; к началу списка добавляется фрагмент x19, и процедура "записьпрограммы" заменяет старую ветвь фрагмента x19 на ветвь, определенную списком x3. Как и выше, доопределяется x6, применяется процедура "коррекцияссылок", и откат к контрольной точке "прием(10)".

2. Оператор x18 не является концевым на главном стволе - либо перед ним идет оператор "ветвь(n)", либо после него - оператор "иначе(n)". Находится вхождение x20 данного оператора перехода x21. Вводится накопитель x22 типа то-

го оператора перехода, который будет использован для продолжения главного ствола после первого фрагмента вставляемой программы. Изначально он устанавливается на символ "ветвь". Для уточнения значения x22 начинается просмотр главного ствола, идущего от фрагмента n; переменная x23 используется как текущий список операторов перехода к подлежащим рассмотрению фрагментам. Если при просмотре встречается оператор "уровень(...)", значения которого пересекаются с уровнями нового приема, то просмотр обрывается и значение x22 сохраняется. При этом, если оператор "уровень($U_1 \dots U_m$)" фрагмента n имел значение U_1 большим, чем в списке x13, то в начале данного фрагмента вставляется ускоряющий фильтр "менее($U_1 - 1$ x5)". Если же цикл просмотра завершился и пересечений с x13 не обнаружилось, то x22 заменяется на "иначе". Далее - откат к переходу через "ветвь 1".

Если после оператора x18 идет "иначе n", причем его уровни пересекаются с x13, то оператор "иначе n" переносится влево от x18 и заменяется на "ветвь 2". После оператора Q вида "уровень" первого фрагмента программы x3 вставляется оператор "контрольприема". Если x22 - "ветвь", то перед Q помещается оператор "ветвь n", в противном случае после него - "иначе n". Если связанный с x18 оператор перехода еще не был заменен в фрагменте x19 на "ветвь 2", то он заменяется на "иначе 2" либо "ветвь 2", с сохранением своего заголовка. Измененный указанным образом фрагмент x19 присоединяется к началу списка x3. Номера всех переходов в x3 до этого были увеличены на 1. Преобразования завершаются стандартным образом - применяется процедура "записьпрограммы", переопределяется x6, и применяется процедура "коррекцияссылок".

Переопределение корневого фрагмента программы символа x1

Если уровни срабатывания нового приема меньше уровней срабатывания ранее зарегистрированных приемов, то даже на повторном цикле точка вставки выбрана не будет. Этот случай рассматривается после контрольной точки "прием(50)". Находится корневой фрагмент x17 программы символа x1. В качестве нового корневого фрагмента будет выбран первый фрагмент F списка x3.

Берется отрезок x19 фрагмента x17, полученный отбрасыванием нескольких первых операторов, образующих стандартное начало ветви приемов сканирования задачи. Чтобы определить тип оператора перехода от F к x19, вводится накопитель x22. Сначала ему присваивается символ "ветвь"; затем предпринимается просмотр главного ствола ветви фрагмента x17. Если хотя бы один из операторов "уровень" этого главного ствола пересекается по своим значениям с набором x13, то символ "ветвь" сохраняется; в противном случае переменной x22 переприсваивается символ "иначе". Если в отброшенном начале фрагмента x17 имелся оператор перехода "иначе", то он переносится в фрагмент F. Предполагается, что это "иначе" идет после оператора "решить". В зависимости от типа x22, в фрагмент F перед или после оператора "уровень" вставляется оператор перехода к последнему фрагменту списка x3, которым становится фрагмент x19. В F вставляется оператор "контрольприема", и реализуется обращение к процедуре "записьпрограммы". Переопределяется x6, и откат к контрольной точке "прием(10)".

Создание логического терминала "прием"

После контрольной точки "прием(10)" предпринимается регистрация ссылки на программу из узла приема. Эта ссылка хранится в логическом терминале "прием". Она состоит из названия символа x1, за которым расположены несколько первых операторов программы - хотя бы для того, чтобы сориентироваться вдоль главного ствола при поиске программы приема. Начальная точка программы распознается по оператору "контрольприема(...)".

Старая версия терминала "прием" удаляется; формируется новая версия x7. Из нее удаляются все операторы, идущие после оператора "уровень". Если остается чрезмерно длинный список операторов, то он укорачивается. Наконец, создается новый терминал "прием" с содержимым x7. Затем реализуется оператор "прогфайл(компонента ...)". Он исключает все фрагменты только что измененной программы символа x1 из зоны программ, ибо в ней остались старые версии фрагментов. Это гарантирует немедленное переключение интерпретатора ЛОСа на выполнение измененной программы.

Заметим, что откаты к описанной процедуре создания терминала "прием" предпринимаются не только для приемов сканирования задачи, но и для всех прочих типов приемов.

19.20.5 Случай приема проверочного оператора, синтезатора, либо пакета продукций

Если новая программа не начиналась с оператора "решить", и таким образом не относилась к приему сканирования задачи, то выполняется переход к контрольной точке "прием(6)". Здесь компиляция разветвляется. В данном подразделе рассмотрим случай, когда первый оператор x9 новой программы имеет вид "программа", причем заголовок приема начинается с одного из символов "быстрпроверка", "контрольбуфера", "см", "значение", "знач", "продукция", "спуск". Таким образом, был откомпилирован прием проверочного оператора, синтезатора либо вычислительного пакета продукций (контрольная точка "прием(33)").

Переменной x10 присваивается корневой фрагмент программы символа x1; переменная x11 инициализируется нулем.

Если фрагмент x10 начинается с оператора "решить", после которого расположен оператор перехода "иначе n", то переменной x11 присваивается набор, сохраняющий исходные значения переменных x1, x7, x8, x10. Переменной x1 переприсваивается пара (x7,x8) - ссылка на надфрагмент фрагмента n; (x7,x8) становится ссылкой на n, а переменной x10 переприсваивается сам фрагмент n.

Если фрагмент x10 начинается с оператора "решить", после которого не идет оператор перехода "иначе", то такой переход создается, и через него подключается новая программа x3. Для этого в первый фрагмент списка x3 вставляется оператор "контрольприема"; в фрагмент x10 после оператора "решить" вставляется оператор "иначе 2"; номера всех ссылок в фрагментах списка x3 увеличиваются на 1; в начале списка x3 присоединяется фрагмент x10, и предпринимается обращение к процедуре "записьпрограммы".

Если фрагмент x10 не начинается с оператора "программа", то на его место будет вставлен начальный фрагмент F списка x3. При этом от F через оператор "иначе", размещаемый сразу после оператора "программа", будет создан переход к ветви фрагмента x10. Для этого выполняются следующие преобразования: в фрагмент F

вводится оператор "иначе(x7 x8)"; в него же вставляется оператор "контрольприема". Если компилируется прием вычислительного пакета продукций, имеющего в своем формате элемент "истина" (при отсутствии срабатываний продукций значение оператора считается истинным), то в начало фрагмента F вставляется ссылка "ветвь" на новый фрагмент "выход", добавляемый к концу списка x3. Ссылка из программы символа x1 на ветвь фрагмента x10 исключается, но сам фрагмент сохраняется в блоке программ. Наконец, выполняется обращение к процедуре "записьпрограммы", причем входные данные определяются с помощью набора x11.

После контрольной точки "прием(39)" рассматривается случай, когда фрагмент x10, как и первый фрагмент новой программы, начинается с оператора "программа". Здесь, как и для программ приемов сканирования задачи, будет предпринят поиск общей начальной части "старой" и "новой" программ. Инициализируется набор x12 ссылок на фрагменты, образующие путь от корня программы символа x1 к текущему фрагменту; последний соответствует началу набора x12. Значением переменной x13 будет текущая позиция текущего фрагмента x10; значением переменной x14 - текущая позиция текущего фрагмента новой программы. Инициализируется нулем вспомогательная переменная x15. Единичное ее значение будет означать, что первый фрагмент новой программы пройден целиком, и текущая позиция x14 относится к одному из последующих фрагментов.

Далее размещен оператор "повторение", откаты к которому происходят при переходе к очередной паре сравниваемых операторов x16, x17, расположенных на позициях x13, x14. Если компилируется прием вычислительного пакета продукций, имеющего указатель "истина" в своем формате, то выполняется переход через оператор "ветвь" в начале "старой" программы.

Если оператор x16 имеет вид "ветвь n", то проверяется выполнение следующих условий:

1. Оператор x17 не имеет заголовка "ветвь";

2. Справа от x17 не расположен оператор "иначе";

3. Оператор P, идущий непосредственно после x16 (с пропуском возможного оператора "контрольприема"), либо совпадает с x17, либо, как и оператор x17, является чисто проверочным и несовместным с x17. В последнем случае после x17 не идет "иначе".

Если хотя бы одно из условий нарушено, то для дальнейшего сравнения с новой программой выбирается фрагмент n. Переопределяются x12, x13, x10, и откат к очередному циклу сравнения пары операторов.

Если же все условия выполнены, то x13 сдвигается на единицу вправо, а величины откатов в новой программе корректируются так, чтобы прохождение через дополнительную "ветвь" не изменило ее функционирования.

Далее рассматривается случай, в котором операторы x16 и x17 заведомо не могут быть истинными одновременно, причем после оператора x16 расположен оператор "иначе n", а после x17 не идет оператор "иначе". Тогда выполняется переход к фрагменту n, переприсваивается переменной x10. x13 переустанавливается на начало этого фрагмента; если оператор x17 представляет собой отрицание оператора x16, то позиция x14 смещается на единицу вправо. Затем откат к повторному сравнению операторов.

Если предыдущая ситуация не имеет места, то переход через "ветвь 2". Здесь рассматривается случай, в котором либо операторы x16, x17 совпадают и после x17 не идет "иначе", либо оператор x16 имеет вид "контрольприема(...)",а после него идет оператор P, совпадающий с x17, не имеющий заголовка "ветвь" и не сопровождаемый оператором "иначе". В последнем случае операторы P и x16 меняются в наборе x10 местами. Если x14 - последняя позиция в текущем фрагменте новой программы, то процедура "записьприема" обрывается - новая программа совпала с уже имеющейся. Иначе x13 и x14 сдвигаются на одну позицию вправо; если после сдвига x13 указывает на оператор "иначе", то выполняется еще один сдвиг x13 вправо. Затем - откат к повторному сравнению операторов.

Если предыдущая ситуация не имела места, то переход через "иначе 1" к фрагменту с контрольной точкой "прием(43)". Ветвь, расположенная после этой контрольной точки, связана с циклом преобразований старой и новой программ, направленных на получение возможно более длинного общего их начала. Используются преобразования двух типов. Первый тип - перемещение вглубь программы чисто проверочного оператора, после которого (или которых) продолжается общая часть двух программ. Второй тип связан с появлением в одной из программ оператора "равно(x t)", вводящего вспомогательное обозначение x при усмотрении повторяющегося вхождения выражения t. Если после такого оператора идет некоторый оператор P, а в другой программе вспомогательное обозначение для t не было введено и идет оператор Q, совпадающий с P после переобозначения в нем t на x, то можно преобразовать другую программу, вводя обозначение x и удлиняя таким образом общее начало двух программ. Указанные действия можно выполнять и для случая группы вспомогательных обозначений. Мы не будем подробно рассматривать ветвь контрольной точки "прием(43)", ввиду ее громоздкости. Заметим только, что возникла она из-за необходимости оптимизировать структуру вычислительных пакетов, программируемых на ГЕНОЛОГе.

Для продолжения действий по сравнению операторов x16, x17 выполняем переход через "ветвь 1" перед контрольной точкой "прием(43)".

Если оператор x16 имеет заголовок "контрольприема", после него идет оператор "ветвь n", и далее - оператор "продолжение", то проверяется, верно ли, что x17 имеет вид "ветвь m" и после x17 расположен оператор "продолжение". В такой ситуации, во-первых, оператор "контрольприема" переносится из фрагмента x10 в начало фрагмента n, и, во-вторых, выполняется переход к началам фрагментов n, m для дальнейшего сравнения операторов. Значение индикатора x15 изменяется на 1.

Если оператор x16 имеет заголовок "контрольприема" и относится к приему с заголовком "уровень", то ветвь фрагмента x10 осуществляет увеличение на единицу текущего уровня в вычислительном пакете продукций. Если после x16 идет оператор P вида "ветвь n", то сам пакет продукций пока пустой - после P расположена не программа какой-либо продукции, а единственный оператор "продолжение". В этом случае предпринимается преобразование фрагмента P - вместо оператора "продолжение" помещается оператор "контрольприема" для нового приема, и далее следует начальный фрагмент его программы. Оператор x16 переносится в фрагмент n. В данном случае переопределяется x6, и откат к созданию терминала "прием".

В отсутствии предыдущей ситуации анализируется совместность оператора x17 и оператора x19, совпадающего с x16 либо, если заголовком x16 служит символ "контрольприема", расположенного от него справа. Если эти операторы - чисто проверочные и несовместны, то проверяется, что после них не идут операторы "иначе". Оператор "контрольприема", если он был, переносится вправо от x19. После x19

вставляется оператор "иначе 2". Затем преобразуется первый фрагмент списка x3. Отбрасывается его начальный отрезок, предшествующий x17. Если x17 - отрицание x19, то отбрасывается и x17. В фрагмент вставляется ссылка "контрольприема" на новый прием. Номера всех переходов в фрагментах списка x3 увеличиваются на 1; к началу списка x3 присоединяется модифицированный фрагмент x10, и выполняется обращение к процедуре "записьпрограммы". Затем - откат к созданию терминала "прием".

Наконец, рассматривается случай, в котором дальнейшее продвижение вглубь старой и новой программ при сравнении операторов невозможно. Тогда перед позицией x13 фрагмента x10 старой программы помещается оператор "ветвь 2", а предшествующие x14 операторы первого фрагмента новой программы отбрасываются. В начало этого фрагмента вставляется оператор "контрольприема"; номера всех переходов в фрагментах списка x3 увеличиваются на 1; к началу списка x3 добавляется модифицированный фрагмент x10, и предпринимается обращение к процедуре "записьпрограммы". После того, как произошла регистрация новой ветви в блоке программ, нужно проконтролировать откаты в старой программе, доходившие хотя бы до вставленного оператора "ветвь". Увеличение этих откатов на единицу выполняется процедурой "сдвиготкатов". Затем - переход к созданию терминала "прием".

19.20.6 Случай приема нормализатора либо анализатора

В случае приемов нормализаторов и анализаторов начальный отрезок программы имеет специальную организацию, несколько усложняющую процедуру вставки новых ветвей. Это происходит из-за того, что нормализаторы и анализаторы работают в режиме итеративных преобразований. После каждого срабатывания приема реализуется откат к заданной точке программы - оператору "повторение", откуда начинается новый цикл поиска приема. Для изменения текущего уровня сканирования либо выхода из пакета по исчерпании уровней служит специальная вставка. Переход к ней через "ветвь" расположен непосредственно после оператора "повторение". Программы новых приемов должны вставляться таким образом, чтобы их ответвление от старой программы происходило после прохождения указанного оператора "ветвь".

Началом рассмотрения программы служит контрольная точка "прием(7)". Предварительно инициализируются переменные x10 и x11. Первая из них будет хранить ссылку на фрагмент программы символа x1, предшествующий текущему; вторая - на текущий фрагмент. Сначала переменной x10 присваивается символ x1.

Далее располагается оператор "повторение", откаты к которому будут происходить при продвижении вдоль главного ствола программы символа x1. Здесь рассматриваются только начальные операторы, т.е. "решить"; "программа"; "обращение(N)". Если найден фрагмент, начальный оператор которого совпадает с начальным оператором x9 новой программы, то переход через "иначе 2". Если весь главный ствол пройден до конца и совпадение не найдено, рассматривается последний фрагмент главного ствола. У него после первого оператора P не идет "иначе", и вся новая ветвь подключается через "иначе", вставляемое после P.

Если найдено совпадение начальных операторов, то инициализируется цикл выделения общего начального отрезка старой и новой программ (контрольная точка "прием(51)"). Инициализируется пустым словом накопитель x13, который будет нужен для коррекций величин откатов в новой ветви программы. В x13 будут сохраняться такие вхождения v в фрагменты новой программы, что величины отката в операто-

рах, обеспечивающих откаты до v, требуют увеличения на единицу. Пополнение x13 будет происходить при переходах через операторы "ветвь" старой программы.

Инициализируется накопитель x14 ссылок на фрагменты, образующие путь от корня найденной ветви программы символа x1 к текущему фрагменту (ссылка на него размещается в начале набора x14). Переменной x12 присвоен текущий фрагмент старой программы; x15 - текущее вхождение в него; x16 - текущее вхождение в текущий фрагмент новой программы. Переменная x17 служит индикатором ухода от первого фрагмента списка x3 вглубь этого списка.

После контрольной точки "прием(9)" расположен оператор "повторение" для откатов при переходе к очередной паре сравниваемых позиций. Переменным x18 и x19 присваиваются, соответственно, оператор старой и новой программ, находящиеся на этих позициях.

Далее идет ряд действий, идентичных описанным в предыдущем разделе. Во-первых, анализируется переход через оператор x18 вида "ветвь n". Во-вторых, рассматривается случай, в котором операторы x18 и x19 заведомо не могут быть истинными одновременно, причем после оператора x18 расположен оператор "иначе n", а после x19 не идет оператор "иначе". После перехода через "ветвь 2" рассматривается случай, в котором либо операторы x18, x19 совпадают и после x19 не идет "иначе", либо оператор x18 имеет вид "контрольприема(...)",а после него идет оператор P, совпадающий с x19, не имеющий заголовка "ветвь" и не сопровождаемый оператором "иначе".

После перехода через "иначе 1" начинаются отличия от предыдущего раздела. Прежде всего, рассматривается случай, когда операторы x18, x19 совпадают, после них расположены операторы "иначе", а вслед за ними - одинаковые операторы "метка(продолжение)". Последние вводятся для того, чтобы указать на возможность пропуска операторов "иначе" при склейке начальных отрезков старой и новой программы. Соответственно, предпринимается смещение позиций x15 и x16 на две единицы вправо, и откат к началу сравнения очередной пары операторов.

Если операторы x18 и x19 различны, причем x18 имеет вид "контрольприема", а после него идет оператор P, совпадающий с x19, то рассматривается случай следования за P оператора "иначе". Если этот оператор расположен на отрезке программы, предшествующем основному ее циклу, либо, аналогично предыдущему случаю, после x19 тоже идет "иначе", и вслед за каждым "иначе" размещаются операторы "метка(продолжение)", то происходит сдвиг позиций x15, x16 вправо, сопровождаемый перенесением вправо от текущей позиции оператора "контрольприема".

Далее снова выполняются действия, аналогичные описанным в предыдущем разделе. Если удается усмотреть, что операторы x18 и x19 - чисто проверочные и несовместны, причем после них не идут операторы "иначе", то новая ветвь подключается через "иначе", вставляемое после оператора x18. Учитывается сдвиг оператора "контрольприема".

Если оператор x18 имеет заголовок "контрольприема", причем относится к приему P с заголовком "окончание" либо "внешвывод", то пакетный нормализатор либо анализатор, прием которого компилируется, пока пустой, а прием P представляет собой переключатель уровней данного пакета. Здесь возникает ситуация, аналогичная случаю приема с заголовком "уровень" из предыдущего раздела: вместо оператора "продолжение", размещенного после оператора "ветвь n", подключается начало новой программы, а x18 переносится в начало фрагмента n.

Если после операторов x18 и x19 идут "переносы" - пары операторов "ветвь", "продолжение", то предпринимается переход к начальным операторам следующих

фрагментов. При этом индикатор x17 устанавливается на 1.

Наконец, при невозможности продолжить выделение общего начального отрезка двух программ выполняется подключение достижимой из x19 ветви новой программы через оператор "ветвь", вставляемый перед оператором x18.

19.21 Развитие компилятора ГЕНОЛОГа

В этом разделе мы приведем описание системы используемых на практике средств для развития ГЕНОЛОГа и его компилятора, проиллюстрировав их рядом упражнений.

Как уже подчеркивалось, в процессе обучения решателя неизбежно возникают ситуации, требующие пополнения ГЕНОЛОГа, и, как следствие, расширения его компилятора. Это обусловлено тем, что ГЕНОЛОГ - не просто еще один язык программирования, а своего рода коллекция способов алгоритмизации теоретических знаний, пополняемая по мере освоения новых разделов в той степени, в какой этим разделам присуща своя "логико-алгоритмическая" специфика. Основной запас выразительных возможностей ГЕНОЛОГа, видимо, стабилизировался, и в большинстве случаев оказывается вполне достаточным. Тем не менее, вряд ли этот запас сейчас следует считать исчерпывающим. По крайней мере, возникающие время от времени принципиально новые направления в обучении логической системы (например, эффективная компиляция обычных вычислений с теоремного уровня; автоматизация развития базы теорем и т.п.) вызывают определенные "всплески" активности, связанной с развитием ГЕНОЛОГа и его компилятора. Видимо, постепенно все большее число процедур, пока реализованных на ЛОСа, будут переводиться на ГЕНОЛОГ. Этот переход представляется естественным для любых процедур "продукционного" типа, распадающихся на независимо применяемые "правила" или "приемы", а большинство сколь-нибудь крупных программ ЛОСа (включая даже программы интерфейсов) именно таковы.

Принцип развития ГЕНОЛОГа "по мере надобности", конечно, приводит к тому, что нынешняя его версия выглядит достаточно бессистемной. Иногда даже ближайшие вариации на тему какого - либо веденного в язык элемента отсутствуют просто потому, что обучающий материал, по той или иной случайности, не предоставил повода для их рассмотрения. Такие пробелы могут обнаружиться лишь впоследствии, и тот, кто с ними столкнется, должен будет не только создать новые приемы решателя, но и пополнить компилятор всем необходимым для их обработки. С другой стороны, строгое следование при развитии как решателя, так и ГЕНОЛОГа принципу "бритвы Оккама" позволяет отсекать множество действительно ненужных возможностей, которые попали бы в систему при ее развитии по принципу "из общих соображений". Выяснение подлинных причин избыточности таких возможностей - дело будущих исследований.

Несмотря на значительный объем компилятора ГЕНОЛОГа, процесс пополнения его не столь сложен, как это могло бы показаться. В действительности это мог бы делать даже новичок, имеющий лишь общее представление об архитектуре компилятора и знакомый с ЛОСом. Предварительное детальное изучение приведенного выше (кстати, не очень полного) материала о компиляторе вовсе не обязательно. Для анализа работы компилятора в конкретных случаях следует использовать отладчик ЛОСа. Основные блоки компилятора суть "базы" независимых приемов, так что достаточно добавить в такую базу еще один прием или найти с помощью от-

ладчика старый прием и модифицировать его. Пополнение языка фильтров приемов вообще не требует анализа компилятора с помощью отладчика, так как обработка фильтров вынесена в специальные справочники.

19.21.1 Поиск в программе компилятора

Начнем с рассмотрения наиболее простой и естественной задачи - поиска некоторого места в программе компилятора, например, для вставки какой-либо новой процедуры или установки прерывания отладчика ЛОСа при выходе на это место. Прежде всего, еще раз напомним общую схему действий компилятора при обработке приемов различных типов.

Основные этапы компиляции приемов

1. Прием сканирования задачи. Происходит обращение к справочнику "новыйприем" на логическом символе - заголовке типа приема. Это справочник переадресует компиляцию процедуре "продукция". Она выполняет следующие действия:

   (a) При необходимости модифицируется теорема приема и его описание. В частности, здесь применяется процедура "развязка";

   (b) Выбирается точка привязки в теореме (процедура "точкапривязки");

   (c) Создается программный блок для накопления фрагментов создаваемой программы. По указателям приема вводятся информационные элементы программного блока. Создается набор установок на идентификацию;

   (d) Для создания идентифицирующей части программы приема предпринимается обращение к процедуре "идентификатор";

   (e) Для компиляции фильтров приема и вставки их в программу предпринимается обращение к процедуре "блокпроверок";

   (f) Для компиляции преобразований текущего контекста, выполняемых приемом, предпринимается обращение к процедуре "преобразователь";

   (g) Для вставки в программу фрагментов, временно сохраненных в информационных элементах программного блока, применяется процедура "вставкафрагментов";

   (h) Для завершающей оптимизации программы применяется процедура "завершениепрограммы";

   (i) Для вставки программы приема в блок программ (т.е. в файлы системы) используется процедура "записьприема".

2. Прием пакетного нормализатора. Происходит обращение к справочнику "новыйприем" на логическом символе "замена". Дальнейшие действия реализуются непосредственно программой этого справочника:

   (a) При необходимости модифицируются теорема приема и его описание;

   (b) Еще до создания программного блока вводится начальный фрагмент программы. Если пакет допускает преобразование подтермов, то в него вводится перечисляющий оператор, выделяющий текущий подтерм. Иначе текущий подтерм жестко фиксирован и совпадает со всем термом. При необходимости создаются операторы, контролирующие заголовок заменяемого

подтерма. Для нормализаторов точкой привязки всегда служит корневое вхождение этого подтерма.

(c) Дальше компиляция разветвляется в зависимости от типа приема нормализатора. Случаи приемов "замена(спускоперандов ...)", "замена(лексупорядочение ...)" почти вырожденные. Основная ветвь - приемы "замена(первыйтерм ...)", "замена(второйтерм ...)";

(d) Создается программный блок, в который заносится уже имеющийся начальный отрезок программы. Вводятся информационные элементы этого блока и установки на идентификацию;

(e) Отдельно рассматривается сравнительно редкий случай приемов разгруппировки, имеющих указатель "набор(первыйтерм)";

(f) Для основного случая предпринимается обращение к процедуре "идентификатор", создающей идентифицирующую часть программы;

(g) С помощью процедуры "фильтр" компилируются фильтры приема. Вставка их в программу выполняется процедурой "вставкафильтра";

(h) Находится программное выражение для заменяющего терма приема. Для этого используется процедура "метаперевод";

(i) Компилируются вспомогательные действия, предваряющие замену, и далее в программу заносится оператор, реализующий замену;

(j) Завершающая обработка программы выполняется процедурами "вставкафрагментов" и "завершениепрограммы". Для вставки новой программы в блок программ используется процедура "записьприема".

3. Прием проверочного оператора. Происходит обращение к справочнику "новыйприем" на логическом символе "спуск". Программа этого справочника выполняет следующие действия:

(a) При необходимости модифицируются теорема приема и его описание;

(b) Еще до создания программного блока вводится начальный отрезок программы, в котором анализируются заголовки входных термов;

(c) Создаются заготовка набора ифнормационных элементов программного блока и набор установок на идентификацию;

(d) Вводится программный блок и пополняется список его информационных элементов;

(e) Предпринимается обращение к процедуре "идентификатор", создающей идентифицирующую часть программы;

(f) Компилируются и вставляются фильтры приема;

(g) Компилируются операторы, осуществляющие учет применения приема и выдающие результат;

(h) Программа оптимизируется и регистрируется в блоке программ - так же, как в предыдущих случаях.

4. Прием пакетного синтезатора. Происходит обращение к справочнику "новыйприем" на логическом символе "значение". Последовательность обработки приема дословно воспроизводит случай проверочных операторов.

5. Прием пакетного анализатора. Происходит обращение к справочнику "новыйприем" на логическом символе "внутрвывод". Предпринимается, если это необходимо, модификация теоремы приема и его описания. Выбирается точка привязки - по корневому вхождению непосредственно идентифицируемого антецедента с наиболее редким заголовком. Затем - стандартная последовательность действий: ввод программного блока и списка установок на идентификацию; обращение к процедуре "идентификатор"; обработка фильтров; создание операторов, осуществляющих вывод нового утверждения либо преобразование старого; оптимизация программы и регистрация ее в блоке программ.

6. Продукция вычислительного пакета. Происходит обращение к справочнику "новыйприем" на логическом символе "продукция". Компиляция здесь зависит от типа вычислительного пакета. Однако, основную ее часть по-прежнему выполняет процедура "идентификатор", так как вычисления определяются программно реализуемыми антецедентами теоремы, а консеквент лишь указывает способ регистрации результатов вычислений в текущих структурах данных.

7. Приемы специальных процедур. Некоторые вспомогательные процедуры также реализованы на ГЕНОЛОГе. Например, программа "рислинии", обеспечивающая сканирование технических структур данных рисунка. Компиляция ее приемов осуществляется справочником "новыйприем" на логическом символе "рисунок". Основную работу здесь выполняет процедура "идентификатор"; все антецеденты теоремы приема полагаются программно реализуемыми. По аналогии с этим можно вводить и процедуры компиляции других программ "продукционного" типа.

8. Приемы справочников. Все они компилируются процедурами справочника "новыйприем" на логических символах - заголовках их названий.

Основные блоки компилятора

Как видно из приведенного выше перечисления типовых циклов компиляции, в них используются почти одни и те же блоки. Создание циклов компиляции для приемов новых типов, при наличии своего рода "конструктора" из стандартных блоков различного назначения и различных размеров, обычно не требует слишком большого времени. Полезно будет перечислить здесь компоненты этого "конструктора", начиная с наиболее крупных его блоков:

1. Процедура "точкапривязки". Выполняет выбор точки привязки для приемов сканирования задачи;

2. Процедура "идентификатор". Создает идентифицирующую часть программы приема;

3. Процедура "учетоперанда". Создает операторы, учитывающие заголовок только что идентифицированного вхождения, и создает установки для продолжения идентификации "вглубь" этого вхождения. Фиксирует идентификацию вхождения в информационных элементах программного блока. Обычно обращение к этой процедуре завершает прием процедуры "идентификатор".

4. Процедура "учетнормализаторов". Подготавливает информационные элементы программного блока, необходимые для учета обращений к нормализаторам.

5. Процедура "метаперевод". Находит программное выражение для терма "теоремного" уровня.

6. Процедура "прогрвыражение". Находит программное выражение ЛОСа для объекта, заданного операторным выражением языка фильтров приема.

7. Процедура "блокпроверок". Компилирует фильтры приема сканирования задачи;

8. Процедура "фильтр". Компилирует отдельный фильтр;

9. Процедура "вставкафильтра". Осуществляет вставку реализующего фильтр оператора в компилируемую программу.

10. Процедура "преобразователь". Компилирует преобразующую часть программы приема сканирования задачи.

11. Процедура "вставкафрагментов". Завершает процесс создания чернового списка фрагментов новой программы, вставляя в него вспомогательные фрагменты, сохраненные в информационных элементах программного блока.

12. Процедура "завершениепрограммы". Устраняет из программы вспомогательные метки и выполняет ее завершающую оптимизацию.

13. Процедура "записьприема". Регистрирует программу приема в блоке программ.

Вспомогательные блоки компилятора

Перечислим используемые на различных этапах компиляции вспомогательные блоки компилятора.

1. Модификация теоремы приема и его описания - процедура "развязка". Эта процедура введена специально для геометрических приемов.

2. Подготовка списка установок на идентификацию. Применяется процедура "смантецеденты", создающая список установок на обработку антецедентов теоремы приема.

3. Процедура "посылки". Определяет программное выражение для списка посылок, относительно которых прием выполняет проверку истинности антецедентов.

4. Процедура "текпеременные". Определяет программное выражение для списка всех встречающихся в контексте применения приема переменных.

5. Процедура "количествооперандов". Вставляет в программу проверку равенства числа операндов идентифицируемой и идентифицирующей операций.

6. Процедура "вхождениепосылки". Находит программное выражение для вхождения в задачу посылки или условия, идентифицированных с заданным антецедентом теоремы приема.

7. Процедура "транскоммент". Определяет набор программных выражений для разрядов комментария, вводимого приемом (с отброшенным заголовком комментария).

8. Процедура "нормоператора". Переобозначает связанные переменные оператора так, чтобы они были не меньше заданной программной переменной. Предотвращает использование в качестве связанных переменных оператора тех переменных, которые уже были определены.

9. Процедура "оператор". Выполняет компиляцию идентифицирующего терма, используемого в фильтрах приема.

10. Процедура "значениепеременной". Находит список всех информационных элементов программного блока, определяющих идентификацию (в том или ином формате) заданной переменной.

11. Процедура "символвхождения". Вставляет в компилируемую программу оператор, анализирующий символ по идентифицируемому вхождению с учетом указателей приема.

12. Процедура "остатокоперандов". Создает информационный элемент программного блока (набороперандов ...), задающий текущий остаток неидентифицированных операндов (в формате термов) ассоциативно-коммутативной операции.

13. Процедура "новооператор". Добавляет один или группу операторов к концу последнего фрагмента программы рассматриваемого программного блока.

14. Процедуры "прогинф", "прогинфы". Заносят, соответственно, один или группу информационных элементов в программный блок.

15. Процедура "транститр". Определяет программное выражение для одного из информационных наборов, организующих текст-формульное пояснение срабатывания приема.

16. Процедура "удалениеприема". Удаляет программу приема в блоке программ, а также ссылку на эту программу из узла приема (т.е. логический терминал "прием").

Справочники компилятора

Многие вспомогательные процедуры компилятора вынесены в справочники. Перечислим основные из них.

1. Справочник "блокпроверок". Обеспечивает компиляцию фильтров приема. Логический символ обращения к справочнику - заголовок фильтра. Наиболее крупная процедура этого справочника обрабатывает фильтры "контекст($A_1 \ldots A_n$)"; она была подробно рассмотрена в приведенном выше техническом описании компилятора. Сначала здесь обрабатываются идентифицирующие термы A_i; затем - указатели "вид", "подтерм", "усм" (с помощью процедуры "идентификатор"); затем - фильтры A_i. Остальные процедуры - как правило, весьма простые.

2. Справочник "прогрвыражение". Обеспечивает компиляцию операторных выражений, используемых в фильтрах приемов. Обработка выражений с заголовками "число", "суммавсех", "выписка", "перечисление", "нод", "нок", "максимум", "минимум" переадресуется процедуре "контекст" справочника "блокпроверок". Программы прочих символов для данного справочника обычно невелики.

3. Справочник "значение". Обеспечивает компиляцию идентифицирующего терма языка фильтров, имеющего заданный заголовок.

4. Справочник "оператор". Уточняет формат какой-либо уже созданной программы ЛОСа, чтобы компилятор мог непосредственно использовать ее при определении истинности фильтра приема, реализации идентифицирующего терма или нахождения значения операторного выражения языка фильтров. В этих случаях обращение к программе в описании приема идентично обращению к ней на ЛОСе. Фактически, справочник дублирует возможности справочников "блокпроверок", "прогрвыражение" и "значение"; он введен для упрощения процедуры пополнения языка фильтров.

5. Справочник "вычисл". Используется для получения списка указателей способов обработки программно релизуемых утверждений и выражений теоремного уровня, имеющих заданный логический символ. Приемы этого справочника создаются непосредственно на ГЕНОЛОГе.

6. Справочник "редакцияпрограммы". Используется для завершающей оптимизации результата компиляции.

   Для различных целей при компиляции используются следующие простые справочники, реализованные на ГЕНОЛОГе:

7. Справочник "легковидеть". Определяет заголовок и формат проверочного оператора для обработки утверждений заданного вида.

8. Справочник "проверка". Определяет заголовок и формат усиленного проверочного оператора для обработки утверждений заданного вида.

9. Справочник "нормализатор". Определяет заголовок нормализатора общей стандартизации выражений с заданным заголовком.

10. Справочник "быстрпреобр". Определяет формат пакетного нормализатора с заданным заголовком.

11. Справочник "синтезатор". Определяет формат пакетного синтезатора с заданным заголовком.

12. Справочник "значения". Определяет список названий пакетных синтезаторов, используемых для обработки утверждений, содержащих заданный ключевой логический символ.

13. Справочник "анализатор". Определяет формат пакетного анализатора с заданным заголовком.

14. Справочник "очевидно". По названию проверочного оператора определяет вид утверждения, подлежащего проверке.

15. Справочник "ключоператора". По названию пакетного оператора определяет символ, за которым в базе приемов закреплен этот оператор. Например, для пакетного нормализатора общей стандартизации выражений с заданным заголовком определяет этот заголовок.

16. Справочник "блок". По названию пакетного оператора фильтра определяет его формат.

17. Справочник "усм". По названию идентифицирующего оператора определяет его формат.

18. Справочник "См". Определяет список названий идентифицирующих операторов, используемых для обработки утверждений, содержащих заданный ключевой символ.

19. Справочник "титр". По названию пакетного оператора определяет комментарий, используемый при организации поясняющего текста, сопровождающего обращение к этому оператору.

20. Справочник "программа". По названию вычислительного пакета ГЕНОЛОГа определяет его формат.

21. Справочник "единица". Определяет "единичное" значение заданной двуместной операции и указывает номер ее операнда. Используется компилятором при обработке указателей идентификации "единица".

22. Справочник "заменазнака". Определяет список указателей возможного вынесения обобщенного знака (одноместной операции) из-под заданной двуместной операции. Учитывается возможность перехода при вынесении от одной одноместной операции к другой.

23. Справочник "отрицание". Распознает операции, обратные к самим себе.

24. Справочник "коммутативно". Распознает коммутативные операции и симметричные отношения.

25. Справочник "ассоциативно". Распознает ассоциативные операции.

26. Справочник "схемаоперандов". Определяет схему возможных перестановок операндов для логических символов со специальной симметрией (в случае коммутативных символов не используется).

27. Справочник "вид". Указывает на специальный оператор, используемый для идентификации терма $t(x)$ с единственной переменной. Пока используется лишь при идентификации степенных выражений.

28. Справочник "пересечениесписков". Указывает на специальную процедуру, используемую для выделения общей части наборов операндов двух операций, имеющих заданный ассоциативно - коммутативный заголовок. Применяется для вещественного и комплексного умножений.

Упражнения

В предлагаемых ниже упражнениях нужно найти через оглавление программ различные точки компилятора. Если в оглавлении программ точка не указана явно, для ее поиска следует войти в смежные с ней явно обозначенные точки и попробовать проследить цепь операторов программы вплоть до искомой точки. Можно также применять просмотр всех вхождений заданного подтерма в заданную ветвь программы

(см. интерфейс редактора программ; клавиша F4). В частности, если нужно найти какую-либо контрольную точку "прием(N)", то после нажатия клавиши F4 терм "прием(N)" набирается в окне диалога и нажимается Enter. Заметим, что цифры номера N набираются раздельно, через вставляемые между ними пробелы.

1. Найти точку обращения к процедуре "идентификатор" при компиляции приема сканирования задачи;

2. Найти начало цикла просмотра установок на идентификацию в процедуре "идентификатор";

3. Найти начальную точку непосредственной идентификации кванторов в процедуре "идентификатор";

4. Найти точку процедуры "идентификатор", в которой рассматривается не коммутативная и не ассоциативная операция, имеющая не более 4 операндов.

5. Найти в процедуре "идентификатор" место, где создается оператор для перехода от уже идентифицированного вхождения к еще не идентифицированной его внешней операции.

6. Найти в процедуре "идентификатор" начальную точку обработки ассоциативной и коммутативной операции.

7. Найти в процедуре "идентификатор" точку создания оператора, перечисляющего утверждения текущего контекста для их непосредственной идентификации.

8. Найти в процедуре "идентификатор" точку, где определяется необходимый для обработки антецедента проверочный оператор.

9. Найти в процедуре "идентификатор" точки создания операторов, реализующих обращения к пакетным синтезаторам.

10. Найти точку вставки нового информационного элемента (быстрпреобр ...) в программный блок.

11. Найти в процедуре "метаперевод" точку ввода новой переменной согласно указателям приема.

12. Найти в процедуре "метаперевод" точку, где происходит предварительная компиляция терма без учета указателей его нормализации. Найти точки, в которых определяются операторные выражения для результатов применения пакетного нормализатора или вспомогательной задачи.

13. Найти точку определения программного выражения для заменяющего терма приема сканирования задачи.

14. Найти точку вставки обращения к оператору вывода новой посылки или нового условия.

15. Найти точки обработки оператором "прогрвыражение" функциональной переменной; указателя вхождения в теорему; обычной переменной.

16. Найти точку обращения к процедуре "идентификатор" в процедуре "контекст" справочника "блокпроверок".

17. Найти точку обработки фильтров при компиляции приема проверочного оператора, нормализатора и синтезатора.

18. Найти начальные точки первого и второго циклов обращений к справочнику "редакцияпрограммы" при оптимизации результата компиляции.

19. Найти начальный оператор "повторение" цикла сравнения начальных отрезков старой и новой программ при регистрации результата компиляции приема сканирования задачи в блоке программ.

Указания

1. Входим в оглавление программ; выбираем раздел "Компилятор ГЕНОЛОГа" и в нем - подраздел "Приемы сканирования задачи". Находим пункт "Обращения к процедурам ИДЕНТИФИКАТОР и БЛОКПРОВЕРОК" и нажимаем "курсор вправо". Возникает фрагмент программы с выделенной голубым цветом контрольной точкой "прием(28)". Следующий оператор - искомое обращение к процедуре "идентификатор". Заметим, что процедура "идентификатор", так или иначе, применяется при компиляции практически любых невырожденных приемов - как сканирования задачи, так и пакетных операторов. Поэтому ее начальная точку (а вовсе не различные точки обращений к ней) удобно выбирать в качестве отправной точки при трассировке работы компилятора. Найдите эту начальную точку через оглавление ГЕНОЛОГа.

2. В разделе "Компилятор ГЕНОЛОГа" выбираем пункт "Формирование идентифицирующей процедуры", и далее - пункт "Цикл просмотра установок на идентификацию". После нажатия "курсор вправо" попадаем на контрольную точку "прием(13)", после которой расположен оператор "входит(x9 x2)". Здесь x2 - список установок на идентификацию; x9 - текущая установка, извлекаемая из списка. Заметим, что точкой отката после обработки очередной установки является оператор "повторение" в надфрагменте данного фрагмента (см. контрольную точку "прием(9)"). От него, после выполнения некоторых корректирующих программный блок действий, компилятор снова достигает оператора "входит(x9 x2)". После этого оператора располагается "главный ствол" классификатора установок по их заголовкам. Его пункты достижимы из подраздела "Типы установок на идентификацию". Текущий уровень x7 применяемых приемов компиляции для каждого типа установок анализируется отдельно; иногда здесь возникают свои линейные цепочки классификации приемов компиляции по уровню x7.

3. В подразделе "Формирование идентифицирующей процедуры" выбираем подраздел "Типы установок на идентификацию". Так как речь идет о непосредственной идентификации квантора, то выбираем далее подраздел "Установка на идентификацию (ОПЕРАНД...)". Кванторы, безусловно, относятся к не коммутативным и не асоциативным символам - выбираем соответствующий подраздел. В нем находим пункт "Идентификация кванторов". Так как этот пункт - не концевой (после номера идет круглая скобка), то входим в его подраздел и

выбираем пункт "Исходная точка". Нажимая клавишу "курсор вправо", попадаем на контрольную точку "прием(45)". Она имеет голубой цвет, т.е. включен режим выделения операторов просматриваемой программы. Чтобы выйти из него, нажимаем клавишу "o" - голубой цвет пропадает, и можно перемещаться по фрагментам программы с помощью клавиш курсоров. Для возвращения в исходный пункт оглавления программ из любой точки просмотра программы достаточно нажать клавиши "o" и "End".

Заметим, что хотя выше и было приведено сравнительно подробное описание того, как работает компилятор при идентификации кванторов, на практике в этом и других случаях проще пользоваться трассировкой работы компилятора, чтобы на конкретном примере понять логику его действий. Технику такой трассировки опишем ниже.

4. Аналогично предыдущему, находим раздел "Типы установок на идентификацию"; в нем - подраздел "Установка (ОПЕРАНД...)", затем - "Не коммутативный и не ассоциативный символ", и наконец - "Расмотрение набора операндов, если его длина не более 4". После выхода на контрольную точку "прием(22)" замечаем, что переменной x12 присваивается список вхождений операндов текущей операции x10. Переходя через "ветвь 1", обнаруживаем оператор "длинаменее(x12 5)". Только после него и начинается искомая ветвь программы. Наличие этой ветви объясняется возможностью использования для ссылок на операнды специальных одноместных выражений "первыйоперанд", "второйоперанд", "предпоследоперанд", "последнийоперанд".

5. Перейдя в раздел "Установка на идентификацию (ОПЕРАНД...)", находим его подраздел "Идентификация в направлении к корню терма". Такая идентификация реализуется, если точка привязки была выбрана в глубине идентифицируемого терма. Так как надоперация текущего вхождения определяется однозначно, то для наискорейшего отсечения выгодно сначала перемещаться от точки привязки к корню, проверяя совпадение встречаемых на этом пути логических символов с символами, указанными в теореме приема. Переходим в выбранный подраздел, и далее - в программу через пункт "Исходная точка". Здесь обнаруживаем оператор "операнд(x11 x10)", присваивающий переменной x11 вхождение надоперации текущего операнда x10.

6. Аналогично предыдущему, но выбираем подраздел "Коммутативный и ассоциативный символ". Попадаем на контрольную точку "прием(63)". После нее начинается цепочка ветвей, соответствующих различным текущим уровням x7. Идентификация операндов ассоциативной и коммутативной операции может привести к значительному перебору; поэтому делается все возможное, чтобы его сократить за счет первоочередной идентификации однозначно распознаваемых операндов. Приемы компилятора сгруппированы в четыре ветви - для идентификации переменных (уровень $x7 = 1$); для идентификации одиночных операндов, выделяемых своими заголовками (уровни $x7 = 2$ и $x7 = 5$); для идентификации операции, выделенной указателем "развертка" (уровень 6); для особых случаев идентификации (например, по пересечению групп операндов нескольких операций). Последняя ветвь рассматривается начиная с $x7 = 4$.

7. Установка на непосредственную идентификацию антецедента имеет вид (корень v), где v - вхождение этого антецедента. Поэтому выбираем в разделе "Типы

установок на идентификацию" подраздел "Установка на идентификацию (КОРЕНЬ...)". В нем выбираем пункт "x12 - заготовка оператора, перечисляющего подлежащие идентификации термы". После контрольной точки "прием(116)" находим оператор "равно(x12 0)", который инициализирует накопитель x12. Далее, в зависимости от типа приема, для перечисления подлежащих идентификации с v утверждений будут использоваться различные источники. Просматриваем операторы "замена(x12 ...)", переприсваивающие переменной x12 реализующий перечисление оператор. Например, в том же фрагменте ниже находим переприсвоение переменной x12 оператора "ключ(конец(x13)x11 начало(x3))". Здесь "конец(x13)" - программное выражение для списка посылок пакетного оператора, x11 - заголовок идентифицируемого антецедента, "начало(x3)" - первая не определенная программная переменная, которой присваиваются перечисляемые утверждения.

8. Для обработки антецедента проверочным оператором используется установка на идентификацию (извлекается v), где v - вхождение этого антецедента. Соответственно, находим в разделе "Типы установок на идентификацию" подраздел "Установка на идентификацию (ИЗВЛЕКАЕТСЯ...)". В нем выбираем пункт "Исходная точка". После контрольной точки "прием(119)" ищем обращения к справочнику "легковидеть" либо "проверка", которые и определяют подходящий проверочный оператор (во втором случае - усиленный). Переходя через "ветвь 2", обнаруживаем в новом фрагменте такой оператор - "равно(x15 справка(...))". Он присваивает переменной x15 тройку (заголовок проверочного оператора - набор вхождений в шаблоне проверяемого утверждения, соответствующих входным данным - число уровней срабатывания проверочного оператора).

9. Аналогично предыдущему, но входим в подраздел "Установка на идентификацию (ЗНАЧЕНИЯ...)". Здесь нам нужно найти не точку определения названия синтезатора, а точку формирования оператора для обращения к этому синтезатору. Выбираем подразделы "Обращение не из пакетных синтезатора либо проверочного оператора" и "Обращение из пакетного синтезатора либо проверочного оператора". В них, соответственно, находим пункты "x33 - обращение к пакетному синтезатору" и "x35 - обращение к пакетному синтезатору".

10. Информационные элементы (быстрпреобр ...) создаются для компиляции обращений к нормализаторам. Поэтому в разделе "Компилятор ГЕНОЛОГа" выбираем подраздел "Процедура предварительной обработки установок на нормализацию". Здесь можно выбрать либо пункт "Исходная точка", либо пункт "Цикл просмотра установок на нормализацию", который, очевидно, должен предшествовать вводу элемента (быстрпреобр ...). Перейдя в программу компилятора и восстановив нажатием клавиши "o" обычный режим ее просмотра, далее пытаемся найти оператор, содержащий символ "быстрпреобр". Для этого включаем поиск - нажимаем F4; вводим текстовым редактором слово "быстрпреобр", нажимаем Enter, и далее продолжаем нажимать на F4 для перехода к очередному вхождению в программу данного слова. После нескольких нажатий обнаруживаем оператор "равно(x13 набор(быстрпреобр ...))", создающий требуемый информационный элемент.

11. В разделе "Компилятор ГЕНОЛОГа" выбираем подраздел "Формирование программного выражения для заданного теоремного терма". Входим в него, и далее

выбираем подраздел "Однобуквенный терм" (т.к. обрабатываем переменную). В нем переходим к подразделу "Случай переменной", и далее к пункту "Ввод новой переменной ...".

12. В разделе "Формирование программного выражения для заданного теоремного терма" выбираем подраздел "Учет нормализаторов". Переходя в программу через пункт "Исходная точка", ищем рекурсивное обращение к процедуре "метаперевод" при заблокированной обработке нормализаторов. Переходя через "ветвь 2", "ветвь 1", находим такое обращение: "равно(x6 метаперевод(x1 x5))". Переменная x6 здесь оказывается равна программному выражению, определяющему искомый терм до применения нормализаторов. Далее идет цикл учета нормализаторов - пакетных операторов либо вспомогательных задач, в котором значение x6 последовательно модифицируется. Выйти на точку применения пакетного нормализатора можно через пункт "Обработка нормализатора". Оператор "равно(x10 сборка(...))" формирует обращение к нормализатору для обработки текущей версии x6. После ряда дальнейших модификаций (учет заданных в приеме опций обращения к нормализатору) значение x10 переприсваивается переменной x6. Аналогично, на точку применения вспомогательной задачи попадаем через пункт "Обработка при помощи вспомогательной задачи". Собственно обращения к задаче - через процедуры "вспомописание", "преобразование", - создаются в глубине данной ветви. Их можно найти либо прямым ее просмотром, либо через F4.

13. В разделе "Компилятор ГЕНОЛОГа" выбираем подраздел "Формирование преобразующей части приема, активизируемого при сканировании задачи". Далее переходим в подраздел "Классификация приемов по заголовкам". Так как речь идет о заменяющем терме, выбираем подраздел "Приемы замены". Здесь находим подраздел "Определение заменяющего терма", и далее выбираем пункт "Формирование прогр. выражения для заменяющего терма - общий случай". Программное выражение x8 для заменяющего терма определяется оператором "замена(x8 метаперевод(x7 x4))".

14. Аналогично предыдущему, переходим в раздел "Классификация приемов по заголовкам", но теперь выбираем подраздел "Приемы вывода - в посылках либо условиях". Далее переходим в программу через пункт "Формирование обращения к оператору вывода". Анализируя окрестность контрольной точки "прием(33)", замечаем, что оператор вывода присвоен переменной x12, а регистрация его в компилируемой программе выполняется оператором "новооператор(x4 x12)".

15. В разделе "Компилятор ГЕНОЛОГа" выбираем подраздел "Процедура ПРОГРВЫРАЖЕНИЕ"; в нем - подраздел "Общий случай". Переходим в программу компилятора через пункт "Функциональная переменная". Контрольной точке "прием(21)" предшествует оператор "равно(x5 значение)", распознающий в обрабатываемом терме функциональную переменную по его заголовку x5. Аналогично, через пункт "Переменная" выходим на начальную точку обработки обычной переменной. Чтобы найти точку обработки указателя вхождения в теорему, вспоминаем, что этот указатель представляет собой терм вида "фикс(...)". Поэтому переходим в подраздел "Служебные слова" и выбираем пункт "фикс".

16. Выбираем в разделе "Компилятор ГЕНОЛОГа" подраздел "Формирование фильтров КОНТЕКСТ(...)", и в нем - пункт "Обращение к оператору ИДЕНТИФИКАТОР".

17. Начинаем во всех случаях с раздела "Компилятор ГЕНОЛОГа". В случае проверочного оператора переходим в подраздел "Приемы проверочных операторов" и находим пункт "Учет фильтров приема". После контрольной точки "прием(14)" располагается оператор "ключ(x6 условие x18)", извлекающий терм "условие(и($A_1 \ldots A_n$))" с фильтрами $A_1, \ldots, A_n$ из описания приема x6. Аналогично поступаем в случае приемов синтезаторов. Для нормализаторов переходим в подраздел "Приемы нормализаторов", затем - в подраздел "Классификация приемов по заголовкам", далее - "Приемы ПЕРВЫЙТЕРМ, ВТОРОЙТЕРМ, ЗАМЕЧАНИЕ". Наконец, переходим в программу через пункт "Учет фильтров приема".

18. Переходим в подраздел "Завершающая обработка программы". Затем - в подраздел "Завершающее редактирование программы". Здесь находятся неконцевые пункты "Цикл обращений к справочнику "редакцияпрограммы" " и "Повторный цикл обращений к справочнику "редакцияпрограммы" ".

19. Последовательно проходим подразедлы "Процедура регистрации программы приема в блоке программ", "Регистрация программы приема сканирования задачи", "Цикл просмотра фрагментов в блоке программ". Здесь выбираем пункт "Начало цикла сравнения операторов нового и старого фрагмента ...". После контрольной точки "прием(16)" и до контрольной точки "прием(15)" находим требуемый оператор "повторение". Видно, что прежде всего анализируется случай, когда на текущей позиции старого фрагмента расположен оператор "ветвь".

19.21.2 Трассировка процесса компиляции

Общая схема трассировки

Если новый прием по каким-либо причинам не компилируется, то для выяснения этих причин применяется трассировка процесса компиляции. Она же помогает разобраться с причинами неправильной компиляции приема, выявляемой в процессе его применения или при просмотре программы сразу после компиляции. Так как априори точка отказа или ошибки компилятора неизвестна, применяются повторные перезапуски процесса трассировки с локализацией этой точки путем деления на части интервала, определяемого счетчиком шагов интерпретатора ЛОСа.

Для запуска компилятора в режиме трассировки нужно нажать клавишу Ctr-F5 из просмотра описания приема. Предварительно рекомендуется нажать End, переводящее в главное меню, и вернуться в просмотр приема, нажав "г" и "курсор вправо". Эти действия необходимы для получения на счетчике шагов работы интерпретатора ЛОСа фиксированного значения, сохраняющегося при перезапусках трассировки. После нажатия клавиши появляется оглавление программ - тот его раздел, который был сохранен при последнем использовании данного оглавления. Вообще говоря, этот раздел не обязан относиться к компилятору. Поэтому нужно выйти в корневое меню оглавления программ, найти подраздел "Компилятор ГЕНОЛОГа" и войти в него.

Дальнейшие действия зависят от типа компилируемого приема. Приемы сканирования задачи и приемы пакетных операторов требуют, в качестве основного этапа компиляции, обращения к процедуре "идентификатор". Отказ от компиляции до этого обращения возникает крайне редко. Поэтому, если нет оснований локализовать отказ компилятора каким-то более определенным местом, трассировку обычно начинаем с исходной точки процедуры "идентификатор". Находим в оглавлении программ пункт ("Формирование идентифицирующей процедуры" - "Начало идентификации ...")и нажимаем клавишу "курсор вправо". Это нажатие запускает компилятор, причем при обращении к процедуре "идентификатор" происходит выход в отладчик ЛОСа после контрольной точки "прием(1)".

Заметим теперь, что в правом верхнем углу экрана находится запись "шаг № N". Она определяет текущий шаг N работы интерпретатора ЛОСа. Так как точка отказа или ошибки компилятора неизвестна, для ее локализации приходится использовать счетчик шагов - обычным методом "деления отрезка". Выбираем некоторый номер M, больший N; нажимаем клавишу "ш"; вводим число M, и нажимаем "Enter" для завершения редактирования этого числа. Повторное нажатие "Enter" приведет к кадру отладчика, соответствующему указанному шагу.

Увеличивать шаг следует осторожно, так как компилятор работает быстро, и можно проскочить точку отказа или ошибки. Если это, все же, произошло, нужно повторно запустить трассировку по изложенной выше схеме, разделив отрезок от N до M на несколько частей и выходя в кадры отладчика в конце каждой части. Чтобы исходный номер N при повторном запуске был в точности повторен, предварительно необходимо нажатием клавиши "End" из просмотра описания приема выйти в главное меню, а затем нажатием клавиш "г", "курсор вправо" вернуться в просмотр приема. Рекомендуемый шаг от N до M при первом запуске - не более 20000. Дальше можно предпринять экстраполяцию по числу оставшихся необработанными установок на идентификацию (список x2). Лучше сделать несколько шагов с малым приращением, чем сразу выполнить презапуск.

Если номер M был выбран удачно и компилятор еще не завершил своей работы, то анализируется наличие ошибки в созданной части программы. Набор фрагментов программы - последний элемент текущего программного блока. Обычно этот блок легко распознается по кадру программы компилятора - к нему обращаются операторы "новооператор", "прогинф", и т.д. Для программы "идентификатор" программный блок является значением переменной x3. Заметим, что в процессе компиляции нумерация программных переменных создаваемой программы может сильно отличаться от той нумерации, которая появится после завершающей оптимизации этой программы. Если ошибка в компилируемой программе уже имеется, то нужно выполнить перезапуск трассировки, и т.п. Если обрабатывается не ошибка, а отказ компилятора, то просто выбирается очередное значение $M_1 > M$, и предпринимается новый шаг трассировки). В тех случаях, когда отрезок, на котором локализуется отказ или ошибка, уже достаточно мал (сотни шагов интерпретатора), можно выполнять трассировку непосредственно, введя пошаговый режим нажатием клавиши "2" и далее используя "Enter" для смены кадра.

Если априори есть основания считать, что ошибка или отказ компилятора произошли в некотором конкретном его блоке, то началом трассировки можно выбрать начало этого блока, определив его через оглавление программ. Затем реализуется уже описанная выше схема деления отрезка, кроме, разумеется, случаев, когда на искомую точку попадаем сразу через оглавление программ. Некоторые программы компилятора не подключены к оглавлению программ (например, какие-либо неболь-

шие процедуры справочников). Если возникает необходимость посмотреть на действия компилятора внутри таких программ, в них через редактор ЛОСа вставляется контрольная точка "трассировка(стоп 0)", после чего запускается процесс компиляции.

Упражнения

Чтобы проиллюстрировать технику трассировки компилятора, разберем несколько простых упражнений. В основном, здесь будет выполняться поиск точки программы компилятора, создающей те или иные операторы программы приема. Поэтому упражнение будет начинаться с поиска в программе приема операторов, соответствующих различным элементам описания приема. Следует помнить, что запуск компилятора означает немедленное удаление старой версии программы приема. По окончании трассировки процесса компиляции, если она не была доведена до конца и программа приема не восстановилась, следует восстанавливать ее нажатием F5.

1. В разделе базы приемов "Элементарная алгебра, Дробь, Устранение кубической иррациональности в знаменателе" выбрать прием с подвыражением a^2-ab+b^2. Найти в программе этого приема оператор, реализующий фильтр "контекст(вид(x1 умножение(...)))". Найти оператор, реализующий проверку первого антецедента. С помощью трассировки процесса компиляции определить моменты создания этих операторов.

2. В разделе базы приемов "Элементарная геометрия, Фигуры, Треугольник, Равнобедренный треугольник, Высота равнобедренного треугольника, опущенная на его основание, ...", выбрать прием, срабатывающий на уровнях 3 и 8. Найти в его программе оператор, реализующий второй антецедент теоремы. С помощью трассировки процесса компиляции определить момент создания этого оператора.

3. В разделе базы приемов "Элементарная алгебра, Тригонометрия, Синус, Нормализатор общей стандартизации НОРМСИНУС, Формулы приведения" выбрать последний прием (нажимая "курсор вниз", пока не будет достигнута концевая точка). В программе этого приема найти оператор, создающий заменяющий терм. Определить момент его компиляции.

4. В разделе базы приемов "Элементарная геометрия, Разныестороны, Проверочный оператор РАЗНЫЕСТОРОНЫ, Учет биссектрисы угла" выбрать первый прием (нажимая "курсор вверх" до крайней точки). В программе приема найти операторы, обрабатывающие второй и третий антецеденты. Определить моменты компиляции этих операторов и уточнить, какой из них к какому антецеденту относится.

5. В разделе базы приемов "Математический анализ, Предел, Нормализатор НОРМПРЕДЕЛ, Предел произведения, Правило Лопиталя" имеется единственный прием. Найти в его программе оператор, реализующий фильтр "меньше(число(комментарий(x8)заголовок(x8 частнпроизв))2)". Определить момент компиляции этого оператора "в целом", а также момент компиляции его идентифицирующего терма "комментарий(x8)".

6. В разделе базы приемов "Аналитическая геометрия, Уравнение плоскости в пространстве, Включение прямой в плоскость" найти первый прием. Определить момент компиляции, на котором учитывается указатель "подстановка(... x15 0)".

7. В разделе базы приемов "Элементарная алгебра, Неравенства, Меньшеилиравно, Синтезатор НИЖНЯЯОЦЕНКА, Квадратный трехчлен" имеется единственный прием. Найти в программе этого приема оператор, осуществляющий идентификацию выражения x^2. Определить момент его компиляции.

8. В разделе базы приемов "Элементарная геометрия, Угол, Анализатор УГЛЫ (...), Сумма острых углов прямоугольного треугольника" выбрать первый прием. Найти в программе приема оператор, определяющий выражение для угла ACB. Найти оператор, выполняющий вывод следствий. Определить моменты компиляции этих операторов.

9. В разделе базы приемов "Математический анализ, Общие свойства числовых функций, Исследование функций, Экстремумы функции на множестве, Определение множества точек, где производная не определена либо не найдена" выбрать первый прием. Найти в его программе операторы, реализующие третий антецедент. Определить момент компиляции этих операторов.

10. В разделе базы приемов "Теория множеств, Функции, Образ, Кванторная свертка в условие непересечения с образом" имеется единственный прием. Изменить его описание, добавив указатель "отображение(x6)". Убедиться, что после этого прием не компилируется. С помощью трассировки найти точку отказа и объяснить отказ. Восстановить исходный вид приема и откомпилировать его.

Указания

1. Находим описание требуемого приема. В третьем окне имеется фильтр "контекст(вид(x1 умножение ...))". Чтобы понять, что он означает, последовательно выделяем его подтермы курсором мыши. Нажатие левой кнопки мыши приводит к появлению многоцветной указки, выделяющей подтерм; после этого нажатие правой кнопки дает прорисовку в нижней части экрана пояснений к выделенному подтерму. Если места внизу недостаточно, перед анализом фильтра убираем с экрана какое-либо "лишнее" окно. В данном примере придется убрать первое окно, нажав Ctr-F1. После этого нетрудно выяснить, что фильтр требует наличие у выражения a множителя, имеющего вид степени, знаменатель показателя которой кратен 3, а основание содержит сумму. Для поиска реализующего фильтр оператора нажимаем "Home" и попадаем в редактор ЛОСа. Здесь удобно перейти в режим просмотра операторов, нажав клавишу "o". С помощью клавиш "курсор вправо - влево" перемещаем выделенную голубым фоном область, просматривая находящиеся на ней операторы. Так как фильтр сравнительно велик, короткие операторы можно пропускать сразу. Первый встретившийся "большой" оператор имеет вид "существует(x17 x18 ...)". Нет необходимости вычитывать его во всех подробностях (хотя с помощью многоцветной указки это не так уж и сложно). Достаточно таких косвенных признаков, как появление внутри оператора упоминаемого в фильтре символа "алгебрвхождение", а также символов "степень" и "дробь".

Теперь возвращаемся в базу приемов (клавиши "о", "End") и переходим к следующему пункту упражнения - поиску оператора, реализующего проверку первого антецедента теоремы. Антецедент указывает на отличие от 0 некоторого выражения. Согласно указателю "блокпроверок(1 6)", он обрабатывается проверочным оператором. Этот проверочный оператор легко запомнить - он называется "усмне0" и размещен в пункте "Цифры" раздела "Элементарная алгебра". Поэтому, переходя обратно в программу, ищем оператор "усмне0(...)". В первом фрагменте программы приема его не оказывается. Поэтому, перейдя через "ветвь 2", начинаем просмотр следующего фрагмента. Напоминаем, что для смены фрагментов нужно выйти из режима выделения операторов нажатием "о". В новом фрагменте обнаруживаем искомый оператор "усмне0(x7 x12 пустоеслово x25)".

Следующий пункт упражнения - начать трассировку процесса компиляции для поиска моментов создания найденных операторов. Здесь можно рассмотреть две возможности - либо применить метод "деления отрезка" с начальной точки процедуры "идентификатор", либо, имея некоторое представление о последовательности действий компилятора, локализовать точку создания операторов по оглавлению программ. Проиллюстрируем обе эти возможности, так как первая из них важна для отладки отказов компилятора.

Нажимаем Ctr-F5 и попадаем в оглавление программ. Переходим к его корневому меню, нажимая необходимое число раз "курсор влево". Затем выбираем пункт ("Компилятор ГЕНОЛОГа", "Формирование идентифицирующей процедуры", Начало идентификации ...). Однако, таким образом мы лишь зафиксировали в оглавлении программ ту точку, с которой будем стартовать при перезапусках трассировки для очередного деления отрезка. Чтобы стандартизировать показания счетчика шагов интерпретатора ЛОСа, теперь вернемся в главное меню двойным нажатием клавиши "End". Затем нажимаем "г", "курсор вправо", "Ctr-F5". В результате снова окажемся на пункте "Начало идентификации ...", но теперь уже нажимаем "курсор вправо" для начала трассировки.

На экране возникает кадр отладчика ЛОСа, соответствующий начальной точке программы "идентификатор". В правом верхнем углу расположена запись "шаг № 128636" (из-за развития системы, при выполнении упражнения здесь может оказаться другая константа).

Можно проконтролировать текущее содержимое накопителя программы. Так как программный блок является значением переменной x3, для просмотра его нажимаем клавиши "К", "3" и "Enter". Возникает четверка объектов, из которой нас будет интересовать последний объект - список фрагментов компилируемой программы. Выбираем клавишей "курсор вниз" четвертый пункт и нажимаем "курсор вправо". Возникает одноэлементный набор, состоящий из единственного пока фрагмента. Его единственный оператор обозначен буквой "т" (терм). Для просмотра этого оператора снова нажимаем "курсор вправо". Видно, что оператор имеет вид "решить".

Далее попробуем перейти к шагу с номером 150000. Нажимаем "ш", вводим число 150000, и нажимаем "Enter" дважды. Снова появляется кадр отладчика ЛОСа в некоторой точке программы "идентификатор". Просматриваем первый фрагмент компилируемой программы так же, как на исходном шаге. Заметим,

что отладчик будет пытаться прорисовать операторы формульным редактором. Чтобы перейти к их "нормальному" виду, нажимаем "с". Просмотр операторов (их уже больше двух десятков) показывает, что нужные нам операторы пока не возникли.

Переходим к шагу с номером 170000. Здесь появляется кадр отладчика на программе "учетнормализаторов". Нажатием "PageUp" возвращаемся к программе "иденитификатор" и снова анализируем накопитель компилируемой программы. В нем обнаруживаем оператор "усмне0(подтерм(x9)облвхожд(x2 x3 x4 x1)пустоеслово x27)". Очевидно, это второй из наших искомых операторов, причем пока он имеет вид, отличающийся от того вида, который принимает после оптимизации программы. Для более точной локализации момента появления данного оператора выполняем несколько перезапусков трассировки с делением на части отрезка от 150000 до 170000. Возвращение в главное меню обеспечиваем несколькими нажатиями "Esc".

Локализация момента приводит к обнаружению того, что после шага 167060 искомый оператор создается и присваивается сначала переменной x21. Чтобы определить, в какой области программы компилятора находится это присвоение, нажимаем клавишу "б" и переходим в оглавление программ. Оказывается, это пункт "Установка на идентификацию ИЗВЛЕКАЕТСЯ" - "Обращение к проверочному оператору" - "Исходная точка". Продолжая трассировку в режиме "по шагам" (переходя к очередному шагу нажатием Enter), вскорости оказываемся на контрольной точке "прием(2 1)", после которой оператор "новооператор(x3 x22)" регистрирует в программе оператор "усмне0(...)". Заметим, что на последнем отрезке он прошел некоторую обработку и был переприсвоен переменной x22.

Продолжаем поиск момента появления в программе оператора, реализующего фильтр. Устанавливя выход в отладчик ЛОСа на шаге 300000, Обнаруживаем, что к этому шагу процедура "идентификатор" уже выполнена и реализуется процедура "блокпроверок". Заметим, что собственно кадр отладчика относится к выполнению некоторых вспомогательных действий последней процедуры, и для выхода на ее кадр нужно 4 раза нажать клавишу "Page Up". В процедуре "блокпроверок" программный блок является значением переменной x4. Просматривая его накопитель, обнаруживаем, что оператора для рассматриваемого фильтра пока не создано.

В принципе, можно было бы продолжить выбор точек прерывания "наугад" и таким образом локализовать искомый момент. Однако, так как фильтры приемов сканирования задачи компилируются именно процедурой "блокпроверок", можно последующую трассировку привязать к логике программы этой процедуры. Легко заметить, что текущий выполняемый оператор процедуры "блокпроверок" имеет вид "равно(x10 фильтр(...))". Таким образом, во-первых, стоит проверить, какой именно фильтр сейчас компилируется; если окажется, что не тот, который нам нужен, то можно ввести установку на прерывания при последующих обращениях к оператору "равно(x10 ...)" и таким образом определить момент обработки нужного фильтра. Чтобы найти, какое входное данное оператора "фильтр" представляет собой обрабатываемый фильтр, можно просмотреть последовательно все его входные данные. Первые переменные x1-x4, скорее всего, воспроизводят внешние входные данные процедуры "блокпроверок", поэтому начинаем сразу с переменной x9. Нажимая клавиши "x", "9",

"Enter", получаем на экране значение переменной x9. Этим значением оказался именно наш фильтр. Если бы оператор "фильтр" имел своими входными программными выражениями не переменные, то для просмотра их значений пришлось бы сначала спуститься из программы "блокпроверок" в программу "фильтр" нажатием "Page Down", и далее анализировать значения первых программных переменных уже в этом кадре. Наконец, для уточнения номера нужного входного операнда процедуры "фильтр" можно было бы нажать F2, ввести название этой процедуры и посмотреть ее описание.

Чтобы определить результат компиляции фильтра, переходим в кадр, соответствующий процедуре "блокпроверок", нажимаем клавишу "2" для задания режима пооператорной трассировки в этом кадре, и нажимаем "Enter". Тогда появляется кадр, соответствующий завершению выполнения оператора "равно(x10 ...)". Вызывая на экран значение переменной x10, получаем определенный компилятором оператор для проверки фильтра. Продолжая трассировку дальше, замечаем, что переменной x11 присваивается набор конъюнктивных членов оператора x10 (так как он не имел заголовка "и", то набор получается одноэлементный). Наконец, кванторная импликация "длялюбого(x12 если входит(x12 x11) не(заголовок(x12 истина)) то вставкафильтра(x4 x12))" регистрирует фильтр в программе. Если это представляет интерес, можно войти в трассировку процедуры "вставкафильтра" и определить, как выбирается точка его вставки.

В заключение заметим, что найти в программе компилятора точки создания указанных операторов можно было бы сразу, используя оглавление программ. Начнем с оператора, реализующего фильтр. Нажимаем Ctr-F5 из просмотра описания приема и выбираем раздел оглавления "Компилятор ГЕНОЛОГа" - "Формирование фильтров приема, активизируемого при сканировании задачи". В этом разделе нам интересен пункт "Формирование и вставка в программу оставшихся фильтров из описания приема", так как рассматриваемый фильтр явно не относится к числу простейших фильтров, вставляемых в программу приема до начала идентификации. Выбирая данный пункт и нажимая "курсор вправо", попадаем в кадр отладчика ЛОСа на контрольную точку "прием(7)" процедуры "блокпроверок". Продолжая пооператорную трассировку нажатиями "Enter", вскоре приходим к уже знакомому оператору "равно(x10 фильтр(x1 x2 x3 x4 x9))". Проверяем значение x9. Это - не наш фильтр. Поэтому устанавливаем режим трассировки с прерываниями перед обращением к найденному оператору: нажимаем клавишу "курсор вниз", перемещаем голубой фон к нужному оператору, и начинаем нажимать "Enter", пока значение переменной x9 не станет равно нашему фильтру. Это происходит уже на следующем шаге.

Заметим, что при выделении голубым фоном оператора в первый раз лучше нажимать Ctr-Enter а для повторных выходов - Enter. Иначе установка на прерывание при обращении к оператору будет утеряна после выхода из текущей реализуемой процедуры.

Чтобы найти момент создания оператора, реализующего проверку первого антецедента, после нажатия Ctr-F5 переходим к пункту "Формирование идентифицирующей процедуры" - "Типы установок на идентификацию" - "Установка на идентификацию ИЗВЛЕКАЕТСЯ" - "Обращение к проверочному оператору" - "Исходная точка". Нажимаем "курсор вправо", и попадаем на контрольную точку "прием(132)"процедуры "идентификатор". Анализируем текущую

установку на идентифкацию x9 - она имеет вид (извлекается v), где v - вхождение нужного нам антецедента. Поэтому продолжаем пооператорную трассировку нажатиями "Enter". Реализуется цикл просмотра антецедента и обращений к справочнику "легковидеть" для определения нужно проверочного оператора. Этот цикл продолжается, пока не будет найдено ненулевое значение x14 названия оператора. Чтобы ускорить трассировку, в момент появления на экране оператора "не(равно(x14 0))" вводим установку на прерывание после его реализации. Например, выделяем голубым фоном оператор "ключ(x5 условие x15)" и нажимаем "Ctr-Enter". Заметим, что выделение голубым фоном оператора "ветвь 1" и нажатие "Ctr-Enter" сразу перевело бы нас в соответствующий подфрагмент и отсекло трассировку текущего фрагмента. Дальнейшую трассировку до момента вставки оператора в программу можно было бы вести "по шагам", или же сразу выйти через оглавление программ на пункт "Вставка обращения к проверочному оператору". В последнем случае нажимаем сначала "Ctr-г" для возвращения из отладчика ЛОСа в оглавление программ. Затем выбираем указанный пункт оглавления и снова нажимаем "курсор вправо". Это приводит к продолжению компиляции и выходу в отладчик ЛОСа на момент вставки (контрольная точка "прием(2 1)").

В заключение возвращаемся к просмотру приема и компилируем его нажатием клавиши F5.

2. Начиная с этого упражнения, будем искать точку создания нужных операторов непосредственно по оглавлению программ, без использования техники "деления отрезка". Второй антецедент теоремы данного приема - условие перпендикулярности прямых BD и AC. Переходя в программу, обнаруживаем в ней оператор "перпендикуляры". Выделяя его левой кнопкой мыши и нажимая затем правую кнопку, просматриваем описание данного оператора. Он перечисляет усматриваемые из посылок задачи перпендикуляры к заданной прямой. Видимо, это и есть искомый оператор. Чтобы определить момент его создания, будем использовать оглавление программ. Замечаем, что второй антецедент теоремы выделен указателем "усм", т.е. подходящий идентифицирующий оператор для его обработки (в данном случае - "перпендикуляры") определяется компилятором с помощью справочника "усм".

   Для входа в трассировку работы компилятора нажимаем Ctr-F5. Появляется оглавление программ. В нем последовательно проходим разделы "Компилятор ГЕНОЛОГа", "Формирование идентифицирующей процедуры", "Типы установок на идентификацию", "Установка на идентификацию (УСМ...)". В последнем разделе обнаруживаем пункт "x36 - обращение к идентифицирующему оператору". Видимо, в этом пункте и создаются операторы для обработки выделенных указателем "усм" антецедентов. Выделяем пункт и нажимаем клавишу "курсор вправо". Появляется контрольная точка "прием(197)", после которой расположен оператор "равно(x36 сборка(x15 x34))". Нажимая на "Enter", выполняем этот оператор и выводим на экран значение x36. Здесь лучше предварительно перейти от формульного к скобочному режиму просмотра, нажав клавишу "с". Значением x36 оказывается другой идентифицирующий оператор, имеющий заголовок "общаяпрямая". Поэтому выделяем голубым фоном оператор "не(равно(x25 0))", расположенный после "равно(x36 ...)", и начинаем просмотр последующих результатов формирования значения x36, переходя

к очередной версии нажатием клавиши "Enter". На втором шаге получаем искомый оператор "перпендикуляры(...)".

3. Переходя в программу приема, ищем в конце ее оператор, переприсваивающий новое значение переменной x1 (текущему обрабатываемому терму). Это значение - x21. Выше находим оператор "равно(x21 нормкосинус(...))". Возвращаемся в просмотр описания приема и нажимаем Ctr-F5. Последовательно проходим разделы "Компилятор ГЕНОЛОГа", "Приемы нормализаторов", "Классификация приемов по заголовкам", "Приемы ПЕРВЫЙТЕРМ, ВТОРОЙТЕРМ, ЗАМЕЧАНИЕ". В последнем разделе выделяем пункт "Формирование заменяющего терма" и нажимаем "курсор вправо". Нажимаем "Enter" для выполнения оператора "равно(x20 метаперевод(x19 x18))". Выводим на экран значение x20 - это оказывается программное выражение "нормкосинус(...)". Однако, от момента получения данного программного выражения до момента вставки его в программу выполняется много других действий. Чтобы сразу попасть на точку вставки, возвращаемся (Ctr-г) в оглавление программ, выбираем пункт "Оператор замены - общий случай" и нажимаем "курсор вправо". Здесь, после нескольких шагов трассировки, приходим к операторам "равно(x24 ...)", "новооператор(x18 суффикс(x24 ...))", регистрирующим в компилируемой программе выполняющие замену операторы.

4. Два оператора "точкалуча(...)" в программе приема, очевидно, соответствуют второму и третьему антецедентам теоремы. Эти антецеденты выделены указателем "усм". Точку их создания находим через оглавление программ так же, как в уже рассмотренном выше геометрическом приеме. Таким образом, попадаем в окно отладчика ЛОСа перед оператором "равно(x36 сборка(x15 x34))". При первом выходе на этот оператор значением x15 служит символ "точкапрямой". Пропуская несколько аналогичных попаданий в это место, наконец получаем x15 = "точкалуча". Чтобы определить, какой именно антецедент компилируется, рассматриваем установку на идентификацию x9. Она имеет вид (усм v), где v - вхождение второго антецедента. Таким образом, порядок расположения операторов "точкалуча" в программе приема совпал с порядком соответствующих антецедентов. Напомним, что связь между теоремными и программными переменными можно также определить непосредственно при трассировке процесса решения задачи, если заблаговременно перекомпилировать нужный прием нажатием клавиши F6.

5. Фильтр "меньше(число(комментарий(x8)заголовок(x8 частнпроизв))2)" должен порождать оператор вида "меньше(...)", содержащий символ "частнпроизв". В начальной части программы обнаруживаем оператор "меньше(суммавсех(x8 и(входит(x8 x3) заголовок(x8 частнпроизв))1)2)". Так как x3 - список комментариев нормализатора, то он действительно реализует рассматриваемый фильтр. Заметим, что фильтр не содержал каких-либо подлежащих идентификации переменных, и естественно было искать его именно в начале программы. Для поиска точки компиляции фильтра нажимаем Ctr-F5 и выбираем в оглавлении программ пункт "Компилятор ГЕНОЛОГа" - "Приемы нормализаторов" - "Классификация приемов по заголовкам" - "Приемы ПЕРВЫЙТЕРМ, ВТОРОЙТЕРМ, ЗАМЕЧАНИЕ" - "Учет фильтров приема". Нажимаем "курсор вправо" и продолжаем пооператорную трассировку до выхода на оператор "равно(x22 фильтр(...))". Здесь смотрим, какой именно фильтр x21 компили-

руется. Пока это не наш фильтр. Поэтому выделяем голубым фоном оператор, предшествующий оператору "равно(x22 ...)", и продолжаем нажимать "Enter", пока значением переменной x21 не станет нужный фильтр. Когда это наконец происходит, можно войти в трассировку действий оператора "фильтр", сменив на несколько тактов режим с пооператорного на пошаговый (клавиша "1"). Однако, здесь предстоит длинная цепь трассировки. Поэтому лучше через оглавление программ сразу выйти на начало программы, обрабатывающей фильтры "контекст" (операторное выражение "число" будет переадресовано этой программе). Выбираем раздел "Формирование фильтров КОНТЕКСТ" и в нем - пункт "Обработка идентифицирующих термов". Снова возвращаемся в трассировку, и продолжаем ее до появляения оператора "оператор(...)", который и будет обрабатывать текущий идентифицирующий терм x12. Убеждаемся, что значением x12 является терм "комментарий(x8)". Далее можно было бы войти в трассировку действий процедуры "оператор(...)". Однако, нам известен заголовок идентифицирующего терма, и видно, что он будет обрабатываться с помощью справочника "значение". Поэтому вводим установку на прерывание при обращении к программе символа "комментарий" - нажимаем "л", набираем слово "комментарий" и дважды нажимаем "Enter". Появляется кадр начала программы "комментарий". Далее нажатием клавиши "2" вводим пооператорную трассировку - иначе каждое следующее нажатие "Enter" будет означать переход к следующему моменту выхода на начало данной программы. Доходим до момента создания оператора x9 вида "входит(x30 x3)", регистрируемого в программном блоке x4. Этот оператор и выполняет перечисление комментариев нормализатора.

6. Указатель "подстановка(... x15 0)" определяет возможность отсутствия явного вхождения члена px в уравнение плоскости; при этом p будет идентифицироваться с 0. Таким образом, нам нужно найти момент создания компилятором оператора, идентифицирующего значение x15 (т.е. p). Нажимаем Ctr-F5 и последовательно проходим разделы оглавления "Компилятор ГЕНОЛОГа", "Формирование идентифицирующей процедуры", "Типы установок на идентификацию", "Установка на идентификацию ОПЕРАНД". Так как выражение px в теореме приема является операндом коммутативной и ассоциативной операции "плюс", то далее входим в раздел "Коммутативный и ассоциативный символ". Отбрасывая очевидно непригодный случай просмотра операндов - переменных, входим в подраздел "Общий цикл просмотра операндов". Здесь обнаруживаем пункт "Учет указателя ПОДСТАНОВКА", выбираем его и нажимаем "курсор вправо". Оказываемся перед оператором "биключ(левпозиция(x3 1)подстановка x11 x12)". Если продолжить трассировку нажатием "Enter", то произойдет откат - мы еще не дошли до компиляции вхождения px. Поэтому поступаем более осторожно - нажимаем "ф" для прориовки на экране всего фрагмента, выделяем голубым фоном оператор, следующий за "биключ(...)", и нажимаем "Ctr-Enter". Проверяем, что значением x11 является вхождение слагаемого px. Затем переходим в пооператорную трассировку (нажимаем клавишу "2") и продолжаем ее. Здесь происходит сложная обработка - по ходу дела выполняются вспомогательное обращение к процедуре "идентификатор" для нового программного блока x20; обращение к процедуре "сжатиефильтра", и лишь затем оператором "новооператор(x3 сборка(и суффикс(x17 x25)))" в программе регистрируется результат компиляции. Выполняем указанный опе-

ратор, вызываем на экран текущую программу блока x3, и находим в ней последний, только что созданный, оператор. Видно, что он сначала инициирует нулем значение p (программная переменная x34), после чего просматривает слагаемые и пытается идентифицировать их с px. При удаче значение x34 изменяется. Переменная x35, инициированная нулем, сохраняет при этом вхождение слагаемого, идентифицированного с px. Это нужно, чтобы далее его уже не рассматривать.

7. Степенные выражения с константным натуральным показателем степени идентифицируются с помощью вспомогательных операторов "выделениестепени", "Выделениестепени". Переходя в программу приема, находим такой оператор - "выделениестепени(x15 2 x2 x3 x16 x17)". Возвращаемся в просмотр приема и нажимаем Ctr-F5. Если попытаться найти точку идентификации степенного выражения в подразделе, связанном с рассмотрением не коммутативных и не ассоциативных символов, то окажется, что компилятор вообще не попадает в этот подраздел. Это происходит из-за того, что идентифицирующий терм может не иметь заголовка "степень" (например, представлять собой числовую константу), и идентификацию его приходится выполнять там, где анализируются заголовки идентифицированных вхождений, т.е. в процедуре УЧЕТОПЕРАНДА. Поэтому, войдя в раздел "Формирование идентифицирующей процедуры", переходим к подразделу "Процедура УЧЕТОПЕРАНДА". В нем обнаруживаем пункт "Идентификация с применением справочника ВИД". Напомним, что справочник "вид" создавался специально для идентификации выражений типа степенных. Если не знать этого, то пришлось бы следить за обращениями к процедуре "учетоперанда" в ее начальной точке. Выходим в кадр отладчика на указанном выше пункте. Нажимаем "ф" для просмотра всего фрагмента программы. Несколько далее обнаруживаем обращение к справочнику "вид". Выделяем голубым фоном оператор, следующий после оператора "не(равно(x12 0))", и нажимаем "Ctr-Enter". Проверяем, что значением переменной x1 (только что идентифицированное вхождение в теорему) является вхождение терма x^2. Нажимая далее "Enter", продолжаем трассировку до оператора "равно(x15 ...)", присваивающего переменной x15 оператор "выделениестепени(...)".

8. В теореме приема встречаются два угла - ABC и ACB. Чтобы различить их, замечаем, что в выводимом утверждении имеется равенство для второго и не имеется равенства для первого. Переходя в программу приема, анализируем программные выражения для частей выводимого утверждения. Обнаруживаем выражение "запись(равно x20 ...)", а несколько выше - оператор "равно(x20 нормугол(запись(угол ...)))". Отсюда заключаем, что x20 и есть угол ACB. Оператор, выводящий следствия в пакетном анализаторе, легко найти по его заголовку "внутрвывод". Чтобы определить момент формирования операторного выражения для угла ACB, находим в оглавлении программ подраздел "Компилятор ГЕНОЛОГа", "Приемы анализаторов", "Формирование выводимого утверждения", и входим в кадр отладчика. Оказываемся непосредственно перед обращением к процедуре "метаперевод". Нажимаем "1" для входа в пошаговую трассировку и нажимаем "Enter". Оказываемся на первом операторе процедуры "метаперевод". Выводим на экран обрабатываемое утверждение x1. В нем нам нужно выделить момент обработки подтерма "угол(ACB)". Нажимаем "ф" для просмотра всего фрагмента и выделяем оператор "обращение(0)" голубым цветом. Нажимаем "Ctr-Enter", и далее последовательно

нажимаем "Enter" до тех пор, пока в качестве значения x1 не получаем "угол(ACB)". Теперь можно проследить все необходимые подробности компиляции данного выражения. Для выхода на точку создания оператора, выполняющего вывод следствия, возвращаемся в оглавление программ (Ctr-г) и выбираем пункт "Вставка оператора "внутрвывод(...)" ".

9. Третий антецедент теоремы приема выделен указателем "развертка". Его обработка состоит в поиске среди утверждений текущего контекста подмножества термов "Частнпроизв(f i $h(i)$)"; i последовательно принимает значения от 1 до m. Константа m идентифицируется до обработки этого антецедента. Переходим в программу приема и ищем в ней символ "Частнпроизв". Он обнаруживается в операторе "длялюбого(x24 если Номера(1 x21 x24)то ...)". Анализируя работу оператора, замечаем, что накопитель x23 заполняется найденными термами "Частнпроизв(...)", а накопитель x22 - их подтермами $h(i)$. Оба эти накопителя инициализируются пустыми словами непосредственно перед оператором.

   Возвращаемся в просмотр приема, нажимаем Ctr-F5 и входим в подраздел "Формирование идентифицирующей процедуры", "Типы установок на идентификацию". Так как утверждения "Частнпроизв(...)" ищутся непосредственно в текущем контексте, выбираем далее подраздел "Установка на идентификацию КОРЕНЬ". В нем выбираем подраздел "Учет указателя РАЗВЕРТКА". Так как длина m идентифицируемого списка утверждений известна (из теоремы приема видно, что она может быть идентифицирована как длина связывающей приставки x), то выбираем подраздел "Длина набора известна". Найденный в программе приема оператор "длялюбого(x24 ...)" предпринимал для текущего значения i (кодируемого переменной x24) просмотр утверждений контекста и идентификацию их с "Частнпроизв(...)". Очевидно, для получения группы операторов, реализующих данную идентификацию, необходимо обращение к процедуре ИДЕНТИФИКАТОР на вспомогательном программном блоке. Поэтому выбираем концевой пункт "Обращение к процедуре ИДЕНТИФИКАТОР" и нажимаем "курсор вправо". Для входа в пооператорную трассировку нажимаем "2", и далее продолжаем ее нажатиями "Enter" до момента присвоения переменной x29 оператора "существует(...)". Выводим его на экран и замечаем, что оператор реализует поиск идентифицирующего терма для текущего значения i. Продолжаем трассировку, пока переменной x31 не будет присвоен итоговый оператор "длялюбого(x19 если Номера(...))". После оптимизации программы нумерация переменных изменится, и он совпадет с оператором, который выше был найден в программе.

10. В четвертом окне приема (в дополнение к старому его содержимому) набираем терм "отображение(x6)". Это означает, что выражение $f(x)$ будет идентифицироваться с произвольным термом, не обязательно имеющим заголовок "значение". Далее нажимаем "Enter" и F3. Однако, снизу появляется красная черта, означающая отказ от компиляции. В этой ситуации иногда удается угадать причину отказа или хотя бы локализовать его, не прибегая к описанной выше процедуре "деления отрезка". В нашем примере видно, что заменяемая (правая) часть теоремы приема вполне стандартна, и каких-либо трудностей с использованием здесь указателя "отображение(x6)" не ожидается. Вероятно, они появятся уже после формирования идентифицирующей части приема и даже после учета его фильтров (последние никак не связаны с x6). Нажимаем Ctr-

F5, находим в оглавлении программ, например, начало процедуры ИДЕНТИФИКАТОР, и переходим от него к кадру отладчика ЛОСа. Нажимаем PageUp для перехода в окно внешней процедуры "продукция". Здесь нажимаем "2" и переходим к пооператорной трассировке. После первого нажатия "Enter" успешно проходим процедуру "идентификатор" и оказываемся перед процедурой "блокпроверок", учитывающей фильтры приема. Снова нажимаем "Enter", и снова процедура оказывается успешно завершенной. Теперь можно было бы установить прерывание при обращении к процедуре "метаперевод" и анализировать процесс формирования программного выражения для заменяющего терма. Однако, в этом терме наиболее сомнительное место - вхождение переменной f (т.е. x6), и можно сразу обратиться к нему через оглавление программ. Нажимаем "Ctr-г"и входим в подраздел "Формирование программного выражения для заданного теоремного терма", "Однобуквенный терм", "Случай переменной", и выбираем для возвращения в отладчик ЛОСа пункт "Исходная точка". Проверяем значение x1 - сначала оно равно b. Для повторного выхода на этот пункт выделяем голубым фоном контрольную точку "прием(7)" и нажимаем Ctr-Enter. Снова проверяем значение x1 - теперь уже оно равно f. Нажимаем "2" для перехода в пооператорную трассировку, и продолжаем ее до тех пор, пока компилятор, исчерпав средства для идентификации f, не выдает отказ оператором "стоп". В заключение возвращаемся к описанию приема и восстанавливаем его исходный вид нажатием "Б".

19.21.3 Пополнение языка новыми элементами

Использование справочников ГЕНОЛОГа

Для тех типов элементов описания приема, которые сравнительно часто приходится пополнять, введены специальные справочники. Создание нового элемента языка здесь сводится к регистрации его в соответствующем help-оглавлении и написании программы нужного справочника, обеспечивающей компиляцию этого элемента. Иногда приходится создавать вспомогательную процедуру ЛОСа, обращение к которой заносится в компилируемую программу.

Начнем с наиболее типичного случая - ввода нового типа фильтра или идентифицирующего терма языка фильтров. Для этого служат упоминавшиеся выше справочники "блокпроверок", "значение" и "оператор". Во всех сомнительных случаях рекомендуется действовать по аналогии с фильтрами, уже имеющимися в системе, анализируя содержимое относящихся к ним программ. Они однотипны, и обучиться самостоятельному созданию новых элементов языка фильтров приемов можно достаточно быстро.

Например, пусть нужно ввести в язык фильтров ГЕНОЛОГа возможность проверять некоторое условие $P(x_1 \ldots x_n)$ для ранее идентифицированных объектов $x_1, \ldots, x_n$. Если это условие может быть реализовано каким-то единственным, и притом не очень сложным, оператором ЛОСа, то лучше воспользоваться справочником "блокпроверок".

Прежде всего, выбираем для обозначения условия P какой-либо еще не использованный в языке фильтров логический символ s. Для этого из просмотра произвольного приема нажимаем Ctr-у, переводящее в оглавление типов фильтров и идентифицирующих термов. Затем нажимаем Ctr-л и получаем доступ к просмотру всех еще не использованных в данном оглавлении символов (можно перелистывать стра-

ницы с помощью PageUp-PageDn). Выбираем подходящее название s, возвращаемся в оглавление, находим в нем подходящую точку, и вводим новый концевой пункт. В нем размещаем текст, поясняющий смысл условия $P(\ldots)$. Затем нажимаем "курсор вправо", "Enter", и в окне текстового редактора набираем шаблон фильтра $s(x_1 \ldots x_n)$, вводя его в тех же переменных $x_1, \ldots, x_n$, которые использовались в поясняющем тексте. По завершении редактирования нажимаем "Enter".

Далее возвращаемся в главное меню и входим в редактор ЛОСа для символа s. Перемещаясь вдоль главного ствола программы символа s, выбираем место для вставки обращения к справочнику "блокпроверок". Например, подключаем такое обращение в конце главного ствола. Вводим первые два оператора "обращение(блокпроверок) метка(икс(6))". Затем, используя операторное выражение "прогрвыражение", определяем программные выражения $X_1, \ldots, X_n$ для аргументов $x_1, \ldots, x_n$ в тех форматах, которые нужны при проверке условия P. Используя эти программные выражения, формируем терм T, значением которого служит оператор ЛОСа, проверяющий истинность условия $P(x_1 \ldots x_n)$. Завершаем программу оператором "ответ(T)".

Если проверка условия P достаточно сложна, то для нее вводим специальную процедуру ЛОСа. Вводим для названия этой процедуры новый логический символ s (раздел "Ресурсы и установки" главного меню). Регистрируем выбранное название в оглавлении типов фильтров и идентифицирующих термов, как и выше. Входим в программу символа s (пока пустую). Нажимаем F3, "Enter" и регистрируем описание оператора $s(x_1 \ldots x_n)$. Затем возвращаемся в редатор ЛОСа, нажимаем "р", и приступаем к набору программы оператора s. По завершении ее ввода, не выходя из ветви символа s, подключаем фрагмент справочника "оператор". Он имеет вид "обращение(оператор) ответ(набор($a_1 \ldots a_n$))", где $a_1, \ldots, a_n$ - указатели формата входных значений $x_1, \ldots x_n$ только что созданного оператора s. Типы указателей формата перечислены в справочной информации символа "оператор". Сразу после этих действий фильтр $s(x_1 \ldots x_n)$ можно использовать в приемах. Заметим, что при использовании справочника "оператор" регистрация названия s в оглавлении типов фильтров и идентифицирующих термов не обязательна - по умолчанию редактор ГЕНОЛОГа будет выдавать первый абзац справочной информации символа s.

Новые идентифицирующие термы вводятся в язык фильтров аналогичным образом. Если оператор ЛОСа, реализующий перечисление для рассматриваемого идентифицирующего терма, сравнительно несложен, то применяется справочник "значение". Это делается аналогично тому, как выше применялся справочник "блокпроверок". Если же перечисление требует программирования специальной вспомогательной процедуры, то снова применяется справочник "оператор". Как и ранее, он выдает набор указателей типов, но уже для объединенного списка входных и выходных переменных. В последнем случае указатель имеет вид "выход(a_i), где a_i - тип значения.

Для операторных выражений языка фильтров снова имеем две возможности. Если для получения значения выражения можно использовать какое-либо сравнительно несложное программное выражение ЛОСа, то применяется справочник "прогрвыражение". При написании его программы следует учитывать указанный в обращении к справочнику требуемый формат результата. Если же вычисление значения требует написания вспомогательной процедуры ЛОСа, то такая процедура создается, и применяется справочник "оператор". К концу набора указателей типов входных данных, выдаваемому этим справочником на названии новой процедуры, добавляется указатель типа значения вычисляемого ею выражения.

Реже возникает необходимость вводить новые типы идентифицирующих операторов. Они применяются, в основном, в планиметрии. Однако, в будущем могут возникнуть и другие разделы базы приемов, требующие интенсивного использования таких операторов. Идентифицирующий оператор используется для быстрой проверки или идентификации каких-либо условий, сводящихся к имеющимся в текущем контексте с помощью двух-трех несложных логических шагов. Чтобы проверка была действительно быстрой, операторы реализуются непосредственно на ЛОСе, хотя представляют собой, по существу микро-пакеты продукций. Как уже говорилось выше, идентифицирующие операторы используют дополнительные источники ускорения - специальные адресные структуры над списком утверждений и выражений задачи, а также буферы, сохраняющие результаты нескольких последних обращений к оператору. Подробнее разобраться с их устройством лучше на каком-либо конкретном примере самостоятельно (можно выбрать для этого оператор "точкипрямой"). Чтобы ввести в компилятор возможность использования нового идентифицирующего оператора, нужно проделать следующее:

1. Выбрать название процедуры, реализующей оператор, и написать на ЛОСе ее программу;

2. Связать новую процедуру с обработкой определенной группы утверждений (антецедентов теоремы приема) при помощи нового приема справочника "усм". Этот прием реализуется на ГЕНОЛОГе. Подробности о формате его теоремы можно найти в справочной информации о символе "усм".

3. Выделить какой-либо логический символ, встречающийся среди обрабатываемых процедурой антецедентов, в качестве "ключевого", и связать его с названием процедуры, введя прием справочника "См". Подробности об этом справочнике - в информации символа "См". Теорема приема его - та же, что для справочника "усм".

Для компиляции программно реализуемых антецедентов теоремы приема используются вспомогательные процедуры ЛОСа, связываемые с антецедентами при помощи логических шаблонов. Эта связь определяется приемами справочника "вычисл". Подробности легко найти в справочной информации символа "вычисл". Справочник обеспечивает связь с логическими конструкциями процедур ЛОСа, реализующих как операторы, так и операторные выражения. Таким образом, пополнение языка здесь совсем несложно - выбирается название новой процедуры, она реализуется на ЛОСе, и создается связывающий ее с компилятором прием справочника "вычисл".

Упражнения

Приведем несколько простых упражнений на ввод новых программ перечисленных выше справочников ГЕНОЛОГа.

1. Ввести фильтр "простэквивалентность(x1)", означающий, что x1 - кванторная импликация с элементарными антецедентами и эквивалентностью элементарных утверждений в консеквенте. Использовать уже имеющийся одноименный оператор ЛОСа.

2. Ввести фильтр для проверки того, что заданное вхождение в утверждение расположено только под внешними символами "и", "или", "существует", либо в

консеквенте под символом "длялюбого". Выбрать новое название для этого свойства вхождения, написать программу проверки его на ЛОСе, и воспользоваться справочником "оператор".

3. Ввести идентифицирующий терм "разделы(x1 x2)", где x2 перечисляет названия всех надразделов, к которым относится логический символ x1.

4. Ввести операторное выражение "внутрзаголовок(x1)", значением которого служит заголовок первой неодноместной операции терма x1, встречающейся при перемещении от его корня.

5. Обеспечить возможность использовать в программно реализуемых антецедентах выражение $n!$ для машинного формата "целое со знаком".

Указания

1. Прежде всего, входим в оглавление языка фильтров ГЕНОЛОГа. Для этого просматриваем какой-либо прием и нажимаем "Ctr-y". Выбираем раздел "Характеристики терма", вводим в конце этого раздела новый концевой пункт и сопровождаем его текстом "Терм x1 представляет собой простую эквивалентность". Нажимаем "курсор вправо", "Enter" и вводим текстовым редактором под горизонтальной чертой текст "простэквивалентность(x1)". Снова нажимаем "Enter", затем - "курсор влево" и "End". Еще раз нажимая "End", возвращаемся в главное меню. Из него входим в программу символа "простэквивалентность", переходим через "иначе 1" и входим в редактирование текущего фрагмента нажатием "p". После оператора "обращение(арность)" вставляем оператор "иначе 1" и нажимаем "Enter". В пустом окне, предназначенном для нового фрагмента, вводим первые операторы программы справочника "блокпроверок" - "обращение(блокпроверок)", "метка(икс(6))". Чтобы уточнить формат обращения к справочнику, нажимаем, не выходя из текстового редактора, клавишу F4, и набираем название справочника - "блокпроверок". Нажимаем "Enter", и из появляющегося на экране текста находим, что при обращении к справочнику текущий обрабатываемый фильтр "простэквивалентность(t)" является значением переменной x5. Так как нам нужно программное выражение для t, возвращаемся в редактирование программы (например, нажатием клавиши пробела) и набираем оператор "равно(x6 прогрвыражение(x1 первыйтерм(x5)набор(терм)x4))". Чтобы уточнить формат обращения к процедуре "прогрвыражение", предварительно можно опять воспользоваться F4. Для проверки того, что не получен отказ, вставляем оператор "не(равно(x6 0))". Наконец, добавляем оператор "ответ(запись(простэквивалентность x6))".

2. В этом примере введем новый оператор ЛОСа, который будет проверять указанное условие. Назовем его, например, "позитивноевхождение(x1)". Создаем через подраздел "Ресурсы и установки" главного меню новый логический символ "позитивноевхождение". Входим в просмотр его программы (пока пустой), нажимаем F3 и набираем описание формата обращения к оператору "позитивноевхождение(x1)". После названия оператора ставим точку, далее - объясняем, что обозначает входная переменная x1 (вхождение в терм), и при каком условии оператор истинен. Создав таким образом необходимую справочную информацию, возвращаемся в просмотр программы, нажимаем "p" и создаем

первые операторы - "программа", "метка(икс(2))". Требуемое условие легко выразить одним оператором ЛОСа - "длялюбого(x2 если подчинено(x1 x2) то или(символ(x2 и)символ(x2 или)символ(x2 существует) и(символ(x2 длялюбого)подчинено(x1 последнийоперанд(x2)))))". После него размещаем заключительный оператор "выход". Прежде чем завершить создание фрагмента нажатием клавиши "Enter", помещаем после оператора "программа" оператор "иначе 1". Тогда, после завершения фрагмента, появится пустое окно текстового редактора для ввода следующего фрагмента. Здесь размещаем программу справочника "оператор", которая позволит компилятору воспользоваться только что созданной процедурой. Эта программа состоит всего из двух операторов - "обращение(оператор)", "ответ(набор(вхождение))".

3. По справочной информации символа "разделы" находим, что процедура ЛОСа, выполняющая требуемое перечисление, уже существует. Так как справочник "оператор" для нее еще не создан, вводим программу этого справочника. Она имеет вид: "обращение(оператор)" "ответ(набор(переменная выход(переменная)))". Заметим, что здесь указатель "переменная" относится одновременно и к переменной, и к логическому символу.

4. Хотя процедура "внутрзаголовок(x1 x2 x3)" уже имеется, воспользоваться в нашем примере справочником "оператор" не удастся, ибо эта процедура реализует не операторное выражение ЛОСа, а оператор. Поэтому вводим программу справочника "прогрвыражение" для символа "внутрзаголовок". Сначала набираем операторы "обращение(прогрвыражение)", "метка(икс(5))". Добавляем к ним оператор, определяющий программное выражение для вхождения x1: "равно(x5 прогрвыражение(x1 первыйтерм(x2)набор(вхождение)x4))". Проверяем, что это выражение найдено: "не(равно(x5 0))". Заносим в компилируемую программу обращение к процедуре "внутрзаголовок": "новооператор(x4 запись(внутрзаголовок x5 начало(x4) плюссимв (начало(x4)1)))". Запоминаем программную переменную "плюссимв(начало(x4)1)", которой присвоен требуемый "внутренний заголовок" - "равно(x6 плюссимв(начало(x4)1))". Сдвигаем указатель первой неиспользованной программной переменной блока x4: "изменение(левыйкрай(x4) плюссимв(начало(x4)1))". Наконец, выдаем результат - "ответ(набор(x6))".

5. Сначала нужно написать на ЛОСе процедуру для вычисления факториала в формате "целое со знаком". Так как для символа "факториал" нет программы операторного выражения, то этот же символ можно использовать в качестве названия новой процедуры. Далее на ГЕНОЛОГе создаем прием справочника "вычисл" по теореме "вычисл(факториал факториал(x1) вход(x1 целое) выход(целое) факториал(x1))".

ЛИТЕРАТУРА

1. А.Черч. Введение в математическую логику. Том 1. М.: ИЛ, 1960, с.484.

2. С.К.Клини. Введение в метаматематику. М.: ИЛ, 1957. 526с.

3. С.К.Клини. Математическая логика. М.: Мир, 1973, 480с.

4. Ч.Чень, Р.Ли. Математическая логика и автоматическое доказательство теорем. М.: Мир, 1983, 360 с.

5. Э.Мендельсон. Введение в математическую логику. М.: Наука, 1971, с.320.

6. Г.Метакидес, А Нероуд. Принципы логики и логического программирования. М.: Факториал, 1998, 288с.

7. И.Братко. Программирование на языке Пролог для искусственного интеллекта. М.: Мир, 1990, 560 с.

8. Дж.Малпас. Реляционный язык Пролог и его применение. М.: Наука, 1990, 464 с.

9. Э.Хант. Искусственный интеллект. М.: Мир, 1978, 558с.

10. Ж.-Л. Лорьер. Системы искусственного интеллекта. М.: Мир, 1991, 568с.

11. E.A.Bender. Mathematical methods in artificial intelligence. Los Alamitos, IEEE, Comp.Society Press, 1996, 638p.

12. Д. Пойа. Математика и правдоподобные рассуждения. М.: Наука, 1975, 463с.

13. Д. Пойа. Математическое открытие. М.: Наука, 1976, 448с.

14. Лавров И.А., Максимова Л.Л., Задачи по теории множеств, математической логике и теории алгоритмов, М., "Наука", 1975, 232 с.

15. Антонов Н.П., Выгодский М.Я., Никитин В.В., Санкин А.И.. Сборник задач по элементарной математике, М., "Наука", 1964, 528 с.

16. Вавилов В.В., Мельников И.И., Олехник С.Н., Пасиченко П.И. Задачи по математике. Алгебра. М., "Наука", 1988, 431 с.

17. Вавилов В.В., Мельников И.И., Олехник С.Н., Пасиченко П.И. Задачи по математике. Уравнения и неравенства. М., "Наука", 1988, 237 с.

18. Потапов М.К., Олехник С.Н., Нестеренко Ю.В. Конкурсные задачи по математике. М., "Наука", 1992, 478 с.

19. Сканави М.И. Сборник задач по математике. М., "Высшая школа", 1988, 431 с.

20. Ваховский Е.Б., Рывкин А.А. Задачи по элементарной математике. М., "Наука", 1971, 360 с.

21. Лидский В.Б., Овсянников Л.В., Тулайков А.Н., Шабунин М.И., Федосов Б.В. Задачи по элементарной математике. М., 1973, 415 с.

22. Сергеев И.Н. Математика. Задачи с ответами и решениями. М., "Высшая школа", 2003, 336 с.

23. Шарыгин И.Ф. Геометрия, 9-11 классы. М., "Дрофа", 1997, 396 с.

24. Шарыгин И.Ф., Гордин Р.К. Сборник задач по геометрии. М., Астрель, 2001, 396 с.

25. Демидович Б.П. Сборник задач и упражнений по математическому анализу. М., "Наука", 1969, 544 с.

26. Виноградова И.А., Олехник С.Н., Садовничий В.А. Задачи и упражнения по математическому анализу. т.1. М., "Высшая школа", 2000, 722 с.

27. Моденов П.С., Пархоменко А.С. Сборник задач по аналитической геометрии. М., "Наука", 1976, 384 с.

28. Филиппов А.Ф. Сборник задач по дифференциальным уравнениям. М., "Наука", 1965, 100 с.

29. Вентцель Е.С., Овчаров Л.А. Задачи и упражнения по теории вероятностей. М., "Высшая школа", 2000, 364 с.

30. Пантелеев А.В., Якимова А.С. Теория функций комплексного переменного и операционное исчисление в примерах и задачах. М., "Высшая школа", 2001, 445 с.

31. Черноуцан А.И. Физика. Задачи с ответами и решениями. М., Книжный дом "Университет", 2001, 335.

32. Подколзин А.С. Об организации баз знаний, ориентированных на автоматическое решение задач. "Дискретная математика", 1990, т.2., вып.1., с. 13-30.

33. Подколзин А.С. Система автоматического решения задач по элементарной алгебре. "Дискретная математика", 1994, т.6., вып.4., с. 35-57.

34. Подколзин А.С. Компьютерный решатель математических задач. ДАН РФ, 1994, т.335, № 4.

35. Подколзин А.С. Компьютерное моделирование процессов решения математических задач. Изд-во ЦПИ при мех.-мат. факультете МГУ, 2001. 235 с.