Математическая Теория Программных Систем

А.М.Миронов

Содержание

Аннотация

Данный текст представляет собой теоретическое обоснование для практического проекта, целью которого является разработка инструментальной среды многоуровневого проектирования программных систем, позволяющей на каждом уровне проектирования производить разработку программной системы одновременно с построением формального доказательства того, что проектируемая система обладает требуемыми качествами.

Глава 1

Введение

1.1 Предмет и основные задачи математической теории программных систем

Основным предметом изучения математической теории программных систем являются

- математические модели программных систем, и
- логические языки, предназначенные для описания свойств программных систем.

Задачи математической теории программных систем в основном связаны со следующими проблемами индустрии программного обеспечения:

1. анализ качества программных систем
2. оптимизирующие преобразования программных систем
3. автоматизация проектирования программных систем, обладающих заданными свойствами.

1.2 Проблема качества программных систем

Современный этап развития индустрии программных систем (которые мы будем называть ниже просто **системами**) характеризуется значительным усложнением процесса их разработки.

В то же время, существующие методы контроля качества разрабатываемых систем характеризуются

- неполнотой,
- высокой сложностью, и
- недостаточной надёжностью.

Данная ситуация неизбежно влечет за собой увеличение числа ошибок при разработке систем.

Требования к качеству систем отражены в стандарте [1]. Отметим наиболее важные из них.

1. **Корректность**, т.е. соответствие системы своему предназначению.
2. **Безопасность**, отсутствие неавторизованной утечки информации в процессе работы системы. Способность к быстрому восстановлению работы после сбоя, возникшего в результате атаки на систему.
3. **Устойчивость** системы в случае непредусмотренного поведения окружения и при работе с неправильными входными данными.
4. **Эффективность** использования ресурсов времени и памяти. Оптимальность реализованных в системе алгоритмов.
5. **Адаптируемость** системы к небольшим изменениям окружения путём изменения её настроек, без изменения её внутренней структуры.
6. Чёткая и понятная **документированность** внутренней структуры системы, позволяющая быстро модифицировать систему в случае существенного изменения условий её использования (например в случае расширения или сужения множества допустимых входных данных).
7. **Переносимость**, т.е. способность системы одинаково хорошо работать на разных платформах и в разных конфигурациях.

Как правило, анализ соответствия системы предъявляемым к ней требованиям производится либо путём её визуального анализа, либо методом **тестирования**.

Однако, если какое-либо из свойств системы может быть выражено формально, например, в виде формулы математической логики, то анализ этого свойства может быть проведён методами **верификации**.

Рассмотрим эти методы подробнее.

1.3 Тестирование

Тестирование системы заключается в анализе её поведения на некоторых выборочных входных данных.

Тестирование (в сочетании с имитационнным моделированием) является в настоящее время основной формой контроля качества систем, и занимает примерно две трети общего времени, затрачиваемого на их разработку.

Тестирование обладает очевидным фундаментальным недостатком: если его возможно провести не для всех допустимых входных данных, а только лишь для их небольшой части (что имеет место почти всегда), то оно не может служить гарантированным обоснованием того, что система обладает проверяемым свойством.

Как отметил Хоар, тестирование может лишь помочь выявить некоторые ошибки, но отнюдь не доказать их отсутствие.

Ошибки в системах могут быть весьма тонкими, и чем тоньше ошибка, тем сложнее (а иногда и просто невозможно) обнаружить её выборочным тестированием. Но во многих системах наличие даже незначительных ошибок категорически недопустимо. Например, наличие даже небольших ошибок в таких системах, как

- системы управления атомными электростанциями,
- медицинские устройства с компьютерным управлением,
- бортовые системы управления самолетов и космических аппаратов,
- системы управления секретными базами данных,
- системы электронной коммерции,

может привести к существенному ущербу для экономики и самой жизни людей.

Гарантированное обоснование качества систем может быть получено только при помощи альтернативного подхода, принципиально отличного от тестирования. Данный подход называется **верификацией**.

1.4 Верификация

Верификация системы состоит из следующих частей.

1. Построение **математической модели** анализируемой системы.
2. Представление проверяемых свойств в виде формального текста (называемого **спецификацией**).
3. Построение **формального доказательства** наличия или отсутствия у системы проверяемого свойства.

Как правило, верификация применяется для анализа первого требования к системе – её корректности. Отметим, что это требование является главным, т.к. в случае его нарушения эксплуатация системы невозможна, даже если она удовлетворяет всем остальным требованиям.

Рассмотрим отдельно каждую из вышеперечисленных частей верификации.

1.4.1 Математические модели систем

Как правило, **математическая модель системы** (называемая ниже просто **моделью**) представляет собой граф,

- вершины которого называются **состояниями**, и изображают ситуации (или классы ситуаций), в которых может находиться система в различные моменты времени, и
- рёбра которого могут иметь метки, изображающие действия, которые может исполнять система.

Функционирование системы изображается в данной модели переходами по рёбрам графа от одного состояния к другому. Если проходимое ребро имеет метку, то эта метка изображает действие системы, исполняемое при переходе от состояния в начале ребра к состоянию в его конце.

Одна и та же система может быть представлена различными моделями, отражающими

- разную степень абстракции при построении модели системы, и
- разные уровни детализации действий, исполняемых системой.

При построении модели системы следует руководствоваться следующими принципами.

1. Модель системы не должна быть чрезмерно детальной, т.к. излишняя сложность модели может вызвать существенные вычислительные проблемы при её формальном анализе.
2. Модель системы не должна быть чрезмерно упрощённой, она должна
   - отражать те аспекты системы, которые имеют отношение к проверяемым свойствам, и
   - сохранять все свойства моделируемой системы, представляющие интерес для анализа

   т.к. в случае несоблюдения этого условия результаты верификации не будут иметь смысла.

1.4.2 Спецификация

Спецификация – это описание свойств системы в виде формального текста.

Спецификация может выражать, например,

- связь между входными и выходными значениями, или
- зависимость между свойствами системы и свойствами её компонентов, которая имеет вид импликации

$$\bigwedge_{i=1}^{n} \left(\begin{array}{l} \text{свойство} \\ i\text{–го компонента} \\ \text{системы} \end{array} \right) \rightarrow \left(\begin{array}{l} \text{свойство} \\ \text{всей системы} \end{array} \right)$$

Как правило, спецификация имеет вид логической формулы, но может иметь и другой вид. Например, спецификацией может служить

- некоторая эталонная модель, относительно которой предполагается, что она обладает заданным свойством, и в этом случае верификация заключается в построении доказательства эквивалентности эталонной и анализируемой моделей, или
- представление анализируемой системы на некотором более высоком уровне абстракции (данный вид спецификаций используется при многоуровневом проектировании систем: реализацию системы на каждом уровне проектирования можно рассматривать как спецификацию для реализации этой системы на следующем уровне).

При построении спецификаций следует руководствоваться следующими принципами.

1. Одно и то же свойство системы может быть выражено на разных языках спецификаций (ЯС), и
   - на одном ЯС оно может иметь простую спецификацию, а
   - на другом – сложную.

   Например, связь между входными и выходными значениями для программы, вычисляющей разложение целого числа на простые множители, имеет
   - сложный вид на языке логики предикатов, но
   - простой вид на другом ЯС.

   Поэтому для представления свойства системы в виде спецификации важно выбрать такой ЯС, на котором спецификация этого свойства имела бы наиболее ясный и простой вид.
2. Если свойство системы изначально было выражено на естественном языке, то при переводе его в спецификацию важно обеспечить адекватность
   - естественно-языкового описания этого свойства, и
   - его спецификации,

   т.к. в случае несоблюдения этого условия результаты верификации не будут иметь смысла.

1.4.3 Построение доказательств

Существует два основных метода построения формального доказательства того, что модель удовлетворяет или не удовлетворяет своей спецификации:

1. **model checking (MC)**, использование которого даёт наибольший эффект в том случае, когда модель **не удовлетворяет** спецификации, и
2. **логический вывод**, который более эффективен (по сравнению с MC) для обоснования того, что модель **удовлетворяет** спецификации.

Поскольку заранее неизвестно, удовлетворяет ли модель спецификации или нет, то для верификации модели следует применять одновременно оба метода.

Model checking

MC представляет собой автоматический анализ модели, которая может быть задана

- либо явно, путём перечисления всех состояний и соединяющих их рёбер
- либо неявно, путём задания булевых функций, изображающих отношение переходов и множество начальных состояний.

Если модель не удовлетворяет спецификации, то в качестве доказательства этого факта MC предъявляет **опровергающее вычисление**, т.е. последовательность действий модели, на которой нарушается эта спецификация.

Логический вывод

Как правило, логический вывод заключается в построении и формальном обосновании утверждений (называемых **инвариантами**), которые должны обладать следующими свойствами:

1. инварианты истинны в начальный момент работы системы
2. инварианты сохраняют свою истинность после каждого шага работы системы, и
3. из конъюнкции данных инвариантов следует спецификация системы.

Глава 2

Модели систем

В данной главе рассматриваются модели программных систем, состоящих из программ, которые взаимодействуют друг с другом посредством общих переменных.

2.1 Выражения

2.1.1 Понятие выражения

Мы предполагаем, что заданы

- некоторое множество $Types$ **типов**, и каждому типу $\tau \in Types$ сопоставлено конечное множество $\mathcal{D}_\tau$ **значений** данного типа
- множество Var **переменных**, причём каждой переменной $x \in Var$ сопоставлен тип $\tau(x) \in Types$
- множество Fun, элементы которого называются **функциональными символами**, причём каждому $f \in Fun$ сопоставлены
  - тип $\tau(f)$, являющийся знакосочетанием вида
$$(\tau_1, \ldots, \tau_k) \to \tau \qquad (2.1)$$
где $\tau_1, \ldots, \tau_k, \tau \in Types$, и
  - функция, обозначаемая тем же символом f, и имеющая вид
$$f: \quad \mathcal{D}_{\tau_1} \times \ldots \times \mathcal{D}_{\tau_k} \to \mathcal{D}_\tau$$

Из переменных, значений и функциональных символов можно строить **выражения**. Каждому выражению e сопоставляется некоторый тип $\tau(e)$.

- Каждая переменная $x \in Var$ является выражением типа $\tau(x)$.
- Для каждого типа $\tau \in Types$ произвольный элемент множества $\mathcal{D}_\tau$ является выражением типа τ. Такие выражения называются **константами**.
- Для
  - каждого списка выражений $e_1, \ldots, e_k$, и
  - каждого функционального символа f, тип которого имеет вид
$$(\tau(e_1), \ldots, \tau(e_k)) \ \to \ \tau$$
знакосочетание
$$f(e_1, \ldots, e_k)$$
является выражением типа τ.

Совокупность всех переменных, входящих в выражение e, обозначается символом $Var(e)$.

2.1.2 Означивания

Пусть e – некоторое выражение.

Означиванием переменных, входящих в e, называется соответствие ξ, которое связывает каждую переменную x из e с некоторым значением $\xi(x) \in \mathcal{D}_{\tau(x)}$.

Каждое означивание ξ переменных из e сопоставляет всему выражению e некоторое значение $\xi(e)$, определяемое рекурсивно:

- если $e = x \in Var$, то $\xi(e)$ уже определено
- если $e \in \mathcal{D}_\tau$, то $\xi(e)$ совпадает с e
- если $e = f(e_1, \ldots, e_k)$ то
$$\xi(e) = f(\xi(e_1), \ldots, \xi(e_k))$$

Выражения e_1 и e_2 называются **эквивалентными**, если для каждого означивания ξ входящих в них переменных имеет место равенство
$$\xi(e_1) = \xi(e_2)$$
Знакосочетание $e_1 = e_2$ выражает тот факт, что e_1 и e_2 эквивалентны.

2.1.3 Булевозначные выражения

Множество $Types$ содержит тип `bool`, значениями которого являются константы 0 и 1.

Булевозначным выражением называется выражение типа `bool`.

Булевозначное выражение e называется **истинным** на означивании ξ, если $\xi(e) = 1$, и **ложным** на ξ, если $\xi(e) = 0$.

Множество Fun содержит символы булевских операций $\neg$, $\wedge$, $\vee$, где

- символ $\neg$ имеет тип

$$\texttt{bool} \to \texttt{bool}$$

- символы $\wedge$ и $\vee$ имеют тип

$$(\texttt{bool}, \texttt{bool}) \to \texttt{bool}$$

Функции, соответствующие этим символам, определяются точно так же, как в логике высказываний.

Для каждого булевозначного выражения e выражение $\neg e$ может также обозначаться символом $\overline{e}$.

Как и в логике высказываний, символы $\wedge$ и $\vee$ пишутся не перед, а между выражениями, которые они связывают.

Для произвольного списка $e_1, \ldots, e_k$ булевозначных выражений знакосочетания

$$e_1 \wedge e_2 \wedge \ldots \wedge e_k \quad \text{и} \quad e_1 \vee e_2 \vee \ldots \vee e_k$$

являются сокращённой записью выражений

$$e_1 \wedge (e_2 \wedge (\ldots \wedge e_k) \ldots) \quad \text{и} \quad e_1 \vee (e_2 \vee (\ldots \vee e_k) \ldots)$$

соответственно. Данные выражения также могут обозначаться знакосочетаниями

$$\left\{ \begin{array}{c} e_1 \\ \ldots \\ e_k \end{array} \right\} \quad \text{и} \quad \left[\begin{array}{c} e_1 \\ \ldots \\ e_k \end{array} \right]$$

соответственно.

Для каждой пары e_1, e_2 булевозначных выражений

- знакосочетание $e_1 \to e_2$ является сокращённым обозначением булевозначного выражения

$$\overline{e_1} \vee e_2$$

- знакосочетание $e_1 \leftrightarrow e_2$ является сокращённым обозначением выражения

$$(e_1 \to e_2) \wedge (e_2 \to e_1).$$

Для каждого семейства булевозначных выражений вида $\{e_i \mid i \in I\}$ и каждого условия $\varphi(i)$ на элементы множества индексов I знакосочетания

$$\bigwedge_{\varphi(i)} e_i \quad \text{и} \quad \bigvee_{\varphi(i)} e_i \tag{2.2}$$

обозначают булевозначные выражения

$$e_{i_1} \wedge \ldots \wedge e_{i_k} \quad \text{и} \quad e_{i_1} \vee \ldots \vee e_{i_k}$$

где $\{i_1, \ldots, i_k\}$ – множество всех индексов $i \in I$, удовлетворяющих условию $\varphi(i)$. Если множество таких индексов пусто, то (2.2) совпадают с константами 1 и 0 соответственно.

2.2 Программы

2.2.1 Понятие программы

Программа представляет собой граф (обычно называемый **блок-схемой**). Одна из вершин программы выделена, и называется **начальной**. Каждая вершина v помечена некоторым **оператором** $Op(v)$ одного из следующих видов:

начало: Данным оператором помечена только начальная вершина. $Op(v)$ имеет вид

$$Init \tag{2.3}$$

где $Init$ – булевозначное выражение, называемое **предусловием** программы.

присваивание: $Op(v)$ имеет вид

$$x := e \tag{2.4}$$

где x – переменная, и e – выражение того же типа, что и x.

проверка условия:

$$Op(v) = \quad b \tag{2.5}$$

где b – булевозначное выражение.

остановка: $Op(v) =$ **halt**

Из вершин с меткой вида (6.2), (6.3), выходит только одно ребро. Из вершин с меткой вида (6.4) выходят два ребра: одно имеет метку "+", другое – метку "−". Из вершин с меткой **halt** не выходит ни одного ребра.

Функционирование программы происходит обычным образом, и заключается в обходе её вершин, с выполнением операторов, сопоставленных проходимым вершинам. После выполнения оператора, соответствующего текущей вершине, происходит переход по выходящему из неё ребру к следующей вершине. На каждом шаге функционирования каждая переменная программы содержит некоторое значение. Значения переменных в начальный момент должны удовлетворять предусловию.

Операторы выполняются следующим образом.

- Оператор (6.3) заносит значения выражения e в переменную x.
- Оператор (6.4) вычисляет значение выражения b, и
  - если оно равно 1, то происходит переход к следующей вершине по ребру с меткой "+",
  - иначе - по ребру с меткой "−".
- Оператор **halt** завершает выполнение всей программы.

2.2.2 Графовые модели программ

Графовая модель программы представляет собой граф G, каждое ребро которого имеет метку, называемую **действием**. Одна из вершин графовой модели выделена, и обозначается символом $Start(G)$.

Графовая модель программы строится следующим образом.

1. На каждом ребре программы рисуется точка. Нарисованные точки являются вершинами графовой модели. Вершиной $Start(G)$ является точка, нарисованная на ребре программы, выходящем из её начальной вершины.

2. Для

   - каждой вершины v программы, и
   - каждой пары a_1, a_2 рёбер программы, таких, что a_1 входит в v, а a_2 - выходит из v

   рисуется ребро графовой модели, соединяющее точку на a_1 с точкой на a_2, и его метка

   - совпадает с $Op(v)$, если $Op(v)$ имеет вид (6.3),
   - имеет вид $b?$, если $Op(v)$ имеет вид (6.4), и a_2 помечено символом "+"
   - имеет вид $\overline{b}?$, если $Op(v)$ имеет вид (6.4), и a_2 помечено символом "−".

3. Для

   - каждой вершины v с меткой **halt**, и
   - каждого ребра a программы с концом в v

   рисуется ребро, началом и концом которого является точка на a, и его метка имеет вид $1?$

2.3 Программные системы

Программной системой называется некоторая конечная совокупность программ.

Функционирование системы заключается в исполнении входящих в неё программ, и может происходить двумя способами:

- **последовательное исполнение:** в каждый такт времени
  - исполняется действие только в одной из программ, и
  - все остальные программы на этом такте времени приостанавливают свою работу,
- **параллельное исполнение:** в каждый такт времени исполняется действие в каждой программе, причём все действия начинаются и заканчиваются одновременно.

2.4 Системы переходов

2.4.1 Понятие системы переходов

Системой переходов (СП) называется пятёрка S вида

$$S = (\ \mathcal{P},\ Q,\ \delta,\ L,\ Q^0\ ) \tag{2.6}$$

компоненты которой имеют следующий смысл.

1. $\mathcal{P}$ – множество, элементы которого называются **утверждениями**.
2. Q – множество, элементы которого называются **состояниями** СП S.
3. δ – бинарное отношение на Q (т.е. $\delta \subseteq Q \times Q$), называемое **отношением перехода**.
4. L – функция вида

   $$L : Q \times \mathcal{P} \to \{0, 1\}$$

   называемая **оценкой**, которая имеет следующий смысл: для каждого $q \in Q$ и каждого $p \in \mathcal{P}$ утверждение p считается
   - **истинным** в состоянии q, если $L(q, p) = 1$,
   - **ложным** в состоянии q, если $L(q, p) = 0$.

   Выражение $L(q, p)$ может записываться более компактно в виде знакосочетания $q(p)$.
5. $Q^0 \subseteq Q$ – множество **начальных состояний**.

СП удобно рассматривать как граф,

- вершинами которого являются состояния, и
- для каждой пары $(q, q') \in \delta$ граф содержит ребро из q в q'.

Ниже мы будем использовать следующие обозначения: для каждого состояния $q \in Q$

$$\begin{aligned} \delta(q) &\stackrel{\text{def}}{=} \{q' \in Q \mid (q, q') \in \delta\} \\ \delta^{-1}(q) &\stackrel{\text{def}}{=} \{q' \in Q \mid (q', q) \in \delta\} \end{aligned}$$

2.4.2 Пути в СП

Путь в СП (2.6) – это последовательность состояний

$$\pi\ =\ (\ q_0,\ q_1,\ \dots\ ) \tag{2.7}$$

такая, что для каждого $i \geq 0 \quad q_{i+1} \in \delta(q_i)$.

Если последовательность (2.7) бесконечна, то путь π называется **бесконечным**, в противном случае он называется **конечным**. Как правило, под путями подразумеваются бесконечные пути, а если рассматриваемый путь является конечным, то это специально оговаривается.

При рассмотрении СП как графа, путь в ней представляет собой последовательность рёбер, в которой конец каждого ребра совпадает с началом следующего ребра.

Мы будем говорить, что путь π является **путём, выходящим из состояния** q (или просто **путём из** q), если первым состоянием (т.е. состоянием с номером 0) на этом пути является q.

Ниже мы будем использовать следующие обозначения:

- для каждого пути π вида (2.7) и каждого $q \in Q$ знакосочетание
$$q \in \pi$$
означает, что $q = q_i$ для некоторого $i \geq 0$
- для каждого пути π вида (2.7) и для каждой пары q, q' состояний знакосочетания
$$q \underset{\pi}{\geq} q', \quad q \underset{\pi}{>} q', \quad q \underset{\pi}{\leq} q', \quad q \underset{\pi}{<} q'$$
означают, что существуют номера i, j, такие, что
$$q = q_i, \quad q' = q_j$$
и, кроме того,
$$i \geq j, \quad i > j, \quad i \leq j, \quad i < j$$
соответственно.

Если π – конечный путь вида
$$\pi = (\ q_0,\ \ldots,\ q_n\ ) \tag{2.8}$$
то говорят, что π **– путь из** q_0 **в** q_n.

Путь вида (2.8) называется **пустым**, если $n = 0$.

Если π – путь вида (2.8), и π' – конечный или бесконечный путь из q_n
$$\pi' = (q_n,\ q_{n+1},\ \ldots)$$
то определена **конкатенация** π и π', обозначаемая символом $\pi \cdot \pi'$, и являющаяся путём вида
$$(q_0,\ \ldots,\ q_n,\ q_{n+1},\ \ldots)$$

Если π – путь вида (2.8), и $q_0 = q_n$, то такой путь называется **циклом**.

Если π – цикл, то знакосочетание π^ω обозначает бесконечный путь, являющийся бесконечной конкатенацией
$$\pi \cdot \pi \cdot \ldots$$

Для каждого пути π знакосочетание $\inf(\pi)$ обозначает множество
$$\{q \in Q \mid q \text{ имеет бесконечно много вхождений в } \pi\}$$

2.4.3 Построение СП, соответствующей программной системе

Пусть $\Sigma = \{\Pi_1, \ldots, \Pi_k\}$ – некоторая программная система.

Ниже мы будем использовать следующие обозначения: для каждого $i = 1, \ldots k$

- G_i обозначает графовую модель программы Π_i
- V_i обозначает множество переменных, входящих в Π_i
- $\hat{V}_i$ обозначает объединение
$$V_i \cup \{at_i\}$$
где at_i – новая переменная, значениями которой являются вершины графа G_i
- V обозначает объединение $\hat{V}_1 \cup \ldots \cup \hat{V}_k$.

Для каждой переменной $x \in V$ мы будем допускать использование в выражениях её **штрихованного дубликата** x'.

Если

- δ – выражение, в которое могут входить переменные из V и их штрихованные дубликаты, и
- (ξ, ξ') – некоторая пара означиваний переменных из V

то значение выражения δ на паре (ξ, ξ') определяется следующим образом: для каждой переменной x из V

- все её вхождения в δ заменяются на $\xi(x)$
- все вхождения её штрихованного дубликата x' в δ заменяются на $\xi'(x)$

и после этого вычисляется значение получившегося выражения.

Для каждого $i = 1, \ldots, k$ совокупность всех рёбер графа G_i обозначается символом $Edges(G_i)$.

Каждому ребру $\alpha \in Edges(G_i)$ соответствует булевозначное выражение $\delta_i(\alpha)$, выражающее собой связь между означиваниями переменных из V до и после исполнения действия, которым помечено ребро α.

В определении выражения $\delta_i(\alpha)$ будут использоваться следующие обозначения:

- символы n и n' обозначают начало и конец ребра α
- для каждого подмножества $X \subseteq V$ знакосочетание $same(X)$ обозначает выражение
$$\bigwedge_{x \in X} (x' = x)$$

Выражение $\delta_i(\alpha)$ имеет следующий вид:

- если метка ребра α имеет вид $x := e$, то

$$\delta_i(\alpha) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} at_i = n \\ at_i' = n' \\ x' = e \\ same(V_i \setminus \{x\}) \end{array} \right\}$$

- если метка ребра α имеет вид $b?$, то

$$\delta_i(\alpha) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} at_i = n \\ at_i' = n' \\ b \\ same(V_i) \end{array} \right\}$$

Нетрудно видеть, что $\delta_i(\alpha)$ истинно на паре (ξ, ξ') означиваний в точности тогда, когда

- если перед исполнением действия, которым помечено ребро α, каждая переменная $x \in \hat{V}_i$ имела значение $\xi(x)$,
- то после исполнения этого действия каждая переменная $x \in \hat{V}_i$ будет иметь значение $\xi'(x)$.

СП, соответствущая системе Σ, имеет следующие компоненты.

1. **Утверждениями** являются булевозначные выражения с переменными из V.
2. **Состояниями** являются означивания переменных из V.
3. **Отношение перехода** состоит из всех пар (ξ, ξ') означиваний, на которых истинно выражение
   - $\delta_1 \vee \ldots \vee \delta_k$, если программы в системе Σ исполняются последовательно, и
   - $\delta_1 \wedge \ldots \wedge \delta_k$, если программы в системе Σ исполняются параллельно,

   где для каждого $i = 1, \ldots, k \quad \delta_i$ представляет собой дизъюнкцию

$$\bigvee_{\alpha \in Edges(G_i)} \delta_i(\alpha)$$

4. **Оценка** сопоставляет паре (ξ, p) значение $\xi(p)$.
5. **Начальными состояниями** являются означивания, на которых истинно выражение

$$\left\{ \begin{array}{l} Init_1 \ \wedge \ \ldots \ \wedge \ Init_k \\ at_1 = Start(G_1) \\ \ldots \\ at_k = Start(G_k) \end{array} \right\}$$

   где $Init_1, \ldots, Init_k$ – предусловия программ из Σ.

2.4.4 Fairness

Если СП S рассматривается как модель реальной системы Σ, то, в частности, каждому вычислению (т.е. последовательности действий) системы Σ должен соответствовать некоторый путь в S.

Однако, иногда S содержит и такие пути, которые не соответствуют никакому реальному вычислению системы Σ.

Один из возможных способов ограничить множество возможных путей СП S, с целью недопущения к рассмотрению тех из них, которые не соответствуют реальным вычислениям моделируемой системы Σ, заключается во введении условий допустимости на пути СП S. Данные условия называются **условиями fairness**. Пути, которые удовлетворяют этим условиям, называются **fair путями**.

Рассмотрим несколько примеров условий fairness.

1. В СП S не должно быть путей, соответствующих таким бесконечным вычислениям системы Σ, в которых одна из входящих в Σ программ не совершает никаких действий после некоторого момента времени.
2. В системе Σ
   - одна из программ (Π_1) может обращаться с запросами к другой программе (Π_2), и
   - программа Π_2 может посылать программе Π_1 ответы на эти запросы.

   В СП S не должно быть путей, соответствующих таким бесконечным вычислениям системы Σ, в которых
   - одно из действий представляет собой запрос от Π_1 к Π_2, и
   - все последующие действия не являются посылкой ответа от Π_2 к Π_1 на этот запрос.
3. В системе Σ одна из программ может посылать сообщения другой программе, причём сообщения при пересылке могут пропадать.

   В СП S не должно быть путей, соответствующих таким бесконечным вычислениям системы Σ, в которых
   - одна из программ бесконечно много раз посылает сообщения другой программе, и
   - все эти сообщения пропадают.

Одна из возможных формализаций условий fairness может представлять собой задание списка F вида

$$\{F_i \mid i = 1, \ldots, k\}, \ \text{где} \ \forall\, i = 1, \ldots, k \quad F_i \subseteq Q \qquad (2.9)$$

и условие fairness на путь π имеет вид

$$\forall i = 1, \ldots, k \qquad \inf(\pi) \cap F_i \neq \emptyset$$

т.е. π должен бесконечно много раз посещать каждое из множеств, входящих в список F.

Глава 3

Темпоральная логика

3.1 Понятие о темпоральной логике

Одним из языков, на котором можно специфицировать свойства систем, является **темпоральная логика**.

Свойства систем описываются в темпоральной логике при помощи **темпоральных формул** (которые мы будем называть также просто **формулами**).

Примеры свойств, которые могут описываться в темпоральной логике:

1. система при любом варианте своего функционирования не будет находиться ни в одном из состояний из заданного класса
2. система при некотором функционировании когда-нибудь попадёт в некоторое состояние из заданного класса

Как правило, при проведении рассуждений о темпоральных формулах рассматриваются не всевозможные формулы, а только формулы из некоторого ограниченного класса. Классы темпоральных формул принято называть **темпоральными логиками**, или просто **логиками**, т.е. словосочетание "темпоральная логика" имеет два значения:

- в первом значении – это язык, на котором можно выражать спецификации,
- а во втором – некоторый класс темпоральных формул.

Наиболее известны темпоральные логики

- CTL (Computational Tree Logic), и
- LTL (Linear Temporal Logic).

Во всех темпоральных формулах основными структурными элементами являются **утверждения**. Утверждения имеют тот же смысл, что и в системах переходов, т.е. для каждого состояния q каждой СП и каждого утверждения p определено значение $q(p) \in \{0, 1\}$. Совокупность всех утверждений обозначается символом $\mathcal{P}$.

Каждая темпоральная логика Φ должна удовлетворять следующим условиям.

1. $\mathcal{P} \subseteq \Phi$.
2. Символы **1** и **0** принадлежат Φ.
3. Если $\psi, \eta \in \Phi$, то знакосочетания

$$\neg\psi, \quad \psi \wedge \eta, \quad \psi \vee \eta \tag{3.1}$$

тоже принадлежат логике Φ.

Формулы (3.1) называются **булевыми комбинациями** формул ψ и η.

Для более наглядной записи конъюнкций и дизъюнкций формул мы будем использовать фигурные и квадратные скобки, аналогично тому, как это делалось в пункте 2.1.3.

Также мы будем использовать в формулах символы $\overline{}$, $\rightarrow$ и $\leftrightarrow$, с тем же смыслом, с каким они использовались в булевозначных выражениях в пункте 2.1.3.

3.2 Логика CTL

3.2.1 CTL–формулы

Темпоральная логика CTL определяется следующими дополнительными правилами:

- если $\psi \in$ CTL, то CTL также содержит следующие 6 формул:

$$\begin{array}{ll} \mathbf{AX}\,\psi & \mathbf{EX}\,\psi \\ \mathbf{AF}\,\psi & \mathbf{EF}\,\psi \\ \mathbf{AG}\,\psi & \mathbf{EG}\,\psi \end{array}$$

- если $\psi, \eta \in$ CTL, то CTL также содержит следующие 4 формулы:

$$\begin{array}{ll} \mathbf{AU}(\psi, \eta) & \mathbf{EU}(\psi, \eta) \\ \mathbf{AR}(\psi, \eta) & \mathbf{ER}(\psi, \eta) \end{array}$$

Жирные символы (**AX**, и т.д.) в данных формулах называются **CTL–операторами**.

Формулы логики CTL мы будем называть **CTL–формулами**.

3.2.2 Значения CTL–формул

Пусть $S = (\mathcal{P}, Q, \delta, L, Q^0)$ – некоторая СП.

Для каждого состояния $q \in Q$ и каждой CTL–формулы φ её **значением** $q(\varphi)$ в состоянии q является булева константа 1 или 0, которая определяется индуктивно:

1. если $\varphi = p \in \mathcal{P}$, то $q(\varphi)$ уже определено в СП S
2. $q(\mathbf{1}) \stackrel{\text{def}}{=} 1, \quad q(\mathbf{0}) \stackrel{\text{def}}{=} 0$
3. - $q(\overline{\psi}) \stackrel{\text{def}}{=} \overline{q(\psi)}$
   - $q(\psi \wedge \eta) \stackrel{\text{def}}{=} q(\psi) \wedge q(\eta)$
   - $q(\psi \vee \eta) \stackrel{\text{def}}{=} q(\psi) \vee q(\eta)$
4. значения формул, начинающихся с CTL-оператора, определяются следующим образом:
   - $q(\mathbf{AX}\psi) = 1$, если для каждого $q' \in \delta(q)$
   $$q'(\psi) = 1 \tag{3.2}$$
   - $q(\mathbf{EX}\psi) = 1$, если существует $q' \in \delta(q)$, такое, что имеет место (3.2)
   - $q(\mathbf{AF}\psi) = 1$, если для каждого пути π из q существует состояние $q' \in \pi$, такое, что имеет место (3.2)
   - $q(\mathbf{EF}\psi) = 1$, если существует путь π из q и существует состояние $q' \in \pi$, такое, что имеет место (3.2)
   - $q(\mathbf{AG}\psi) = 1$, если для каждого пути π из q и для каждого $q' \in \pi$ имеет место (3.2)
   - $q(\mathbf{EG}\psi) = 1$, если существует путь π из q, такой, что для каждого состояния $q' \in \pi$ имеет место (3.2)
   - $q(\mathbf{AU}(\psi, \eta)) = 1$, если для каждого пути π из q существует состояние $q' \in \pi$, такое, что
   $$\left\{ \begin{array}{l} q'(\eta) = 1, \text{ и} \\ \forall\, q'' \underset{\pi}{\leq} q' \quad q''(\psi) = 1 \end{array} \right\} \tag{3.3}$$
   - $q(\mathbf{EU}(\psi, \eta)) = 1$, если существует путь π из q и существует состояние $q' \in \pi$, такое, что имеет место (5.1)
   - $q(\mathbf{AR}(\psi, \eta)) = 1$, если для каждого пути π из q и для каждого $q' \in \pi$
   $$\left[\begin{array}{l} q'(\eta) = 1, \text{ или} \\ \exists\, q'' \underset{\pi}{\leq} q' : \quad q''(\psi) = 1 \end{array} \right] \tag{3.4}$$
   - $q(\mathbf{ER}(\psi, \eta)) = 1$, если существует путь π из q, такой, что для каждого $q' \in \pi$ имеет место (3.4).

Значением CTL-формулы φ в СП S называется множество

$$Q_\varphi \stackrel{\text{def}}{=} \{q \in Q \mid q(\varphi) = 1\} \tag{3.5}$$

3.2.3 Эквивалентность CTL–формул

Мы будем называть CTL–формулы φ и ψ **эквивалентными**, если для каждого состояния q в произвольной СП имеет место равенство

$$q(\varphi) = q(\psi)$$

Если CTL–формулы φ и ψ эквивалентны, то мы будем обозначать этот факт знакосочетанием $\varphi = \psi$.

Нетрудно доказать, что имеют место следующие соотношения:

- законы де Моргана:
$$\overline{\varphi \wedge \psi} = \overline{\varphi} \vee \overline{\psi}, \qquad \overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi}, \qquad \overline{\overline{\varphi}} = \varphi$$
- $\overline{\mathbf{AX}\varphi} \;=\; \mathbf{EX}\overline{\varphi}$
- $\mathbf{EF}\varphi \;=\; \mathbf{EU}(\mathbf{1}, \varphi)$
- $\overline{\mathbf{AF}\varphi} \;=\; \mathbf{EG}\overline{\varphi}$
- $\overline{\mathbf{AG}\varphi} \;=\; \mathbf{EF}\overline{\varphi}$
- $\overline{\mathbf{AU}(\varphi, \psi)} = \left[\begin{array}{l} \mathbf{EU}(\overline{\psi}, \overline{\varphi} \wedge \overline{\psi}) \\ \mathbf{EG}\overline{\psi} \end{array} \right]$
- $\overline{\mathbf{AR}(\varphi, \psi)} = \mathbf{EU}(\overline{\varphi}, \overline{\psi})$
- $\overline{\mathbf{ER}(\varphi, \psi)} = \mathbf{AU}(\overline{\varphi}, \overline{\psi})$

Из данных соотношений следует, что для любой CTL–формулы φ существует эквивалентная ей CTL–формула ψ, в которую входят только следующие CTL–операторы:

$$\mathbf{EX}, \quad \mathbf{EG}, \quad \mathbf{EU} \tag{3.6}$$

3.2.4 Примеры свойств систем, выражаемых CTL–формулами

1. **EF** (Start $\wedge$ $\overline{\text{Ready}}$)
   (достижимо состояние, в котором свойство Start выполняется, а условие Ready – не выполняется)
2. **AG** (Request $\rightarrow$ **AF** Acknowledgement)
   (если получен запрос, то когда-нибудь на него будет дан ответ)
3. **AG** (**AF** DeviceEnabled)
   (при любом функционировании системы условие DeviceEnabled выполнено бесконечно много раз)
4. **AG** (**EF** Restart)
   (из каждого состояния достижимо состояние, в котором выполняется свойство Restart)

3.3 Model checking для CTL

3.3.1 Задача MC-CTL

Одна из возможных форм задачи **model checking** для CTL (которую мы будем ниже обозначать знакосочетанием **MC-CTL**) заключается в том, чтобы по

- заданной СП $S = (\mathcal{P}, Q, \delta, L, Q^0)$, и
- заданной CTL–формуле φ

вычислить множество Q_φ.

Можно считать, что φ содержит только CTL–операторы вида (3.6).

Для вычисления множества Q_φ можно использовать следующий рекурсивный алгоритм.

1. Если $\varphi = p \in \mathcal{P}$, то Q_φ определяется непосредственно оценкой СП S.

2. Если φ имеет вид $\mathbf{1}$ или $\mathbf{0}$, то Q_φ имеет вид соответственно Q или $\emptyset$.

3. Если φ имеет вид

   $$\overline{\psi}, \quad \psi \wedge \eta, \quad \text{или} \quad \psi \vee \eta$$

   то Q_φ имеет вид соответственно

   $$Q \setminus Q_\psi, \quad Q_\psi \cap Q_\eta, \quad Q_\psi \cup Q_\eta$$

4. Если $\varphi = \mathbf{EX}\psi$, то $Q_\varphi = \{q \in Q \mid \delta(q) \cap Q_\psi \neq \emptyset\}$.

5. Если $\varphi = \mathbf{EU}(\psi, \eta)$, то для вычисления Q_φ мы будем использовать вспомогательное множество $Q' \subseteq Q$, и сначала полалагаем

   $$Q_\varphi := Q_\eta, \quad Q' := Q_\eta$$

   Затем работает цикл:

   while $(Q' \neq \emptyset)$
   { выбираем $q \in Q'$, и удаляем q из Q'
   для каждого $q' \in \delta^{-1}(q)$
   $\left\{ \begin{array}{l} q' \in Q_\psi \\ q' \notin Q_\varphi \end{array} \right\}$? добавляем q' к Q_φ и к Q'
   }

6. Пусть $\varphi = \mathbf{EG}\psi$.

   Рассмотрение этого случая мы начнём с введения вспомогательных понятий.

   Подмножество Q' множества состояний Q называется **сильно связным подмножеством**, если для каждой пары q_1, q_2 состояний из Q' существует непустой путь из q_1 в q_2.

   Максимальное (по включению) сильно связное подмножество называется **сильно связной компонентой** (strongly connected component, SCC).

   Существует алгоритм (называемый алгоритмом Тарьяна) нахождения всех SCC, сложность которого равна $O(|Q| + |\delta|)$.

   Сильно связные компоненты могут рассматриваться не для всего множества Q, а для некоторого его подмножества Q_1. Сильно связная компонента в Q_1 будет сильно связным подмножеством и в Q, но может не быть SCC в Q.

   Согласно определению, соотношение

   $$q(\mathbf{EG}\psi) \;=\; 1 \tag{3.7}$$

   означает, что существует путь $\pi \subseteq Q_\psi$ из q.

   Поскольку множество Q является конечным, то существует такое состояние $q' \in \pi$, что все состояния из совокупности

   $$\{q'' \in \pi \mid q'' \underset{\pi}{\geq} q'\} \tag{3.8}$$

   входят в π бесконечно много раз. Состояния, входящие в (3.8), образуют сильно связное подмножество множества Q_ψ, и, следовательно, содержатся в некоторой SCC множества Q_ψ.

   Обозначим символом π_0 начальный отрезок пути π, который заканчивается в q'.

   Мы установили, что из (3.7) следует соотношение

   $$\left.\begin{array}{l} \exists \;\text{SCC}\; C \text{ множества } Q_\psi, \quad \exists\, q' \in C, \\ \exists \;\text{путь } \pi_0 \subseteq Q_\psi \text{ из } q \text{ в } q' \end{array}\right\} \tag{3.9}$$

   Обратно, из (3.9) следует (3.7), т.к. полагая

   $$\pi \stackrel{\text{def}}{=} \pi_0 \cdot \pi_1^\omega \tag{3.10}$$

   где π_1 – цикл из q' в q', содержащийся в C, имеем: $\pi \subseteq Q_\psi$.

   Порядок вычисления множества Q_φ имеет следующий вид.

   (a) Находим в Q_ψ все SCC, и полагаем Q' равным множеству всех состояний, входящих в эти SCC. Также полагаем $Q_\varphi := Q'$.

   (b) Затем работает тот же цикл, что и в предыдущем пункте.

Нетрудно подсчитать, что сложность предложенного алгоритма равна

$$O(|\varphi| \cdot |S|)$$

где $|\varphi|$ – размер формулы φ, и $|S| = |Q| + |\delta|$, т.к.

- число подформул формулы φ не превосходит $|\varphi|$,
- $|S|$– верхняя оценка числа шагов для анализа каждой подформулы.

3.3.2 Задача fair MC-CTL

Если в СП задан список F условий fairness вида (2.9), то можно определить понятие **fair-значения** CTL–формулы φ в состоянии q этой СП, которое будет обозначаться знакосочетанием $q^F(\varphi)$.

Определение fair-значений отличается от определения из пункта 3.2.2 следующей модификацией:

- везде, где упоминается слово "путь", перед ним ставится эпитет "fair"
- для любых $q \in Q$ и $\varphi \in$ CTL, если не существует ни одного fair пути из q, то $q^F(\varphi) = 0$.

Обозначим символом Q^F_ψ множество

$$\{q \in Q \mid q^F(\psi) = 1\}.$$

Небольшой модификацией рассуждений из предыдущего пункта можно обосновать, что

$$q^F(\mathbf{EG}\psi) = 1$$

тогда и только тогда, когда

$$\left.\begin{array}{l} \exists \text{ fair SCC } C \text{ множества } Q^F_\psi, \quad \exists\, q' \in C, \\ \exists \text{ путь } \pi_0 \text{ из } q \text{ в } q', \text{ причём } \pi_0 \subseteq Q^F_\psi \end{array}\right\} \qquad (3.11)$$

где SCC C множества Q^F_ψ называется **fair**, если

$$\forall F_i \in F \quad F_i \cap C \neq \emptyset$$

Отметим, что выражение

$$q^F(\mathbf{EG1}) \qquad (3.12)$$

принимает значение 1 тогда и только тогда, когда существует fair путь из q. Данное выражение мы будем обозначать знакосочетанием

$$q(fair)$$

Как следует из вышесказанного, его значение может быть вычислено за время

$$O(|F| \cdot |S|)$$

где множитель $|F|$ (число условий в списке F) присутствует по причине того, что надо проверять каждую SCC, является ли она fair.

Знакосочетание $fair$ можно рассматривать как новое утверждение, значение которого в каждом состоянии q равно значению выражения (3.12).

Fair-значения сложных формул можно вычислять по следующим правилам:

1. $q^F(p) = q(p \wedge fair)$.
2. $q^F(\mathbf{EX}\psi) = q\big(\mathbf{EX}(\psi \wedge fair)\big)$
3. $q^F(\mathbf{EU}(\psi, \eta)) = q\big(\mathbf{EU}(\psi, (\eta \wedge fair)\big)$.

Из вышесказанного вытекает, что для произвольной CTL–формулы φ значение $q^F(\varphi)$ может быть вычислено за время

$$O(|\varphi| \cdot |S| \cdot |F|)$$

3.4 Монотонные операторы и их неподвижные точки

3.4.1 Монотонные операторы

Пусть Q – некоторое конечное множество, и $\mathbf{2}^Q$ – совокупность всех его подмножеств.

Монотонный оператор на $\mathbf{2}^Q$ – это отображение

$$\mathcal{F} : \mathbf{2}^Q \to \mathbf{2}^Q \qquad (3.13)$$

обладающее следующим свойством:

$$\forall\, A, B \in \mathbf{2}^Q \qquad A \subseteq B \quad \Rightarrow \quad \mathcal{F}(A) \subseteq \mathcal{F}(B)$$

Очевидно, что композиция монотонных операторов является монотонным оператором.

Подмножество $A \subseteq Q$ называется **неподвижной точкой** оператора (3.13), если имеет место равенство

$$A = \mathcal{F}(A)$$

Ниже вместо словосочетания "неподвижная точка" будет использоваться аббревиатура **FP** (fixpoint).

Оператор (3.13), как правило, задаётся в виде алгебраического выражения, в котором используются теоретико-множественные операции и переменная-аргумент. Мы будем указывать аргумент в скобках справа от $\mathcal{F}$.

Оператор $\mathcal{F}(Z)$ может иметь несколько FP.

FP оператора $\mathcal{F}(Z)$ называется

- **наименьшей** (и обозначается $\mu Z.\mathcal{F}(Z)$), если она содержится во всех остальных FP $\mathcal{F}(Z)$, и
- **наибольшей** (и обозначается $\nu Z.\mathcal{F}(Z)$), если она содержит все остальные FP $\mathcal{F}(Z)$.

Каждый монотонный оператор $\mathcal{F}(Z)$ вида (3.13) имеет наименьшую и наибольшую FP, и

$$\begin{aligned} \mu Z.\mathcal{F}(Z) &= \bigcup_{i \geq 0} \mathcal{F}^i(\emptyset) = \mathcal{F}^{i_0}(\emptyset) \\ \nu Z.\mathcal{F}(Z) &= \bigcap_{i \geq 0} \mathcal{F}^i(Q) = \mathcal{F}^{j_0}(Q) \end{aligned}$$

для некоторых i_0 и j_0, где $\mathcal{F}^i(A) \stackrel{\text{def}}{=} \underbrace{\mathcal{F}(\ldots(\mathcal{F}}_{i}(A)))$.

Один из возможных алгоритмов вычисления наименьшей и наибольшей FP монотонного оператора (3.13) имеет следующий вид:

$$A := \left\{\begin{array}{l} \emptyset \ \text{(для наименьшей FP)} \\ Q \ \text{(для наибольшей FP)} \end{array}\right.$$

$$\textbf{do} \left\{\begin{array}{l} B := A \\ A := \mathcal{F}(A) \end{array}\right\} \textbf{while } (B \neq A)$$

return A

3.4.2 Вычисление Q_φ на основе понятия FP

Изложенный в пункте 3.3.1 алгоритм вычисления множества Q_φ может быть преобразован в той его части, которая связана с вычислением значений формул, начинающихся с **EG** и **EU**.

Нетрудно доказать, что следующие операторы являются монотонными:

1. операторы

$$A\cap \quad \text{и} \quad A\cup : \ \mathbf{2}^Q \to \mathbf{2}^Q$$

(где $A \subseteq Q$ – фиксированное подмножество), которые сопоставляют каждому $B \in \mathbf{2}^Q$ подмножества $A \cap B$ и $A \cup B$ соответственно, и

2. оператор

$$\mathbf{EX} : \mathbf{2}^Q \to \mathbf{2}^Q$$

который сопоставляет каждому $B \in \mathbf{2}^Q$ подмножество

$$\mathbf{EX}(B) \stackrel{\text{def}}{=} \{q \in Q \mid \delta(q) \cap B \neq \emptyset\}$$

Из данных определений вытекает, что для любых CTL–формул ψ, η имеют место соотношения:

$$\left\{ \begin{array}{l} Q_{\mathrm{EX}\psi} = \mathbf{EX}(Q_\psi) \\ Q_{\mathrm{EG}\psi} = \nu Z.\Big(Q_\psi \cap \mathbf{EX}(Z)\Big) \\ Q_{\mathrm{EU}(\psi,\eta)} = \mu Z.\Big(Q_\eta \cup (Q_\psi \cap \mathbf{EX}(Z))\Big) \end{array} \right. \tag{3.14}$$

Таким образом, вычисление значений формул, начинающихся с **EG** и **EU**, может быть сведено к задаче вычисления соответствующих FP.

3.4.3 Задача fair-MC-CTL с условиями fairness в виде CTL–формул

Если условия fairness в списке (2.9) выражены CTL–формулами, т.е.

$$\forall i = 1, \ldots, k \quad F_i = Q_{\psi_i} \quad \text{где } \psi_i \in \text{ CTL} \tag{3.15}$$

то множество $Q^F_{\mathrm{EG}\psi}$ может быть представлено в виде

$$Q^F_{\mathrm{EG}\psi} \ = \ \nu Z.\left\{ \begin{array}{l} Q_\psi \\ \bigcap_{i=1}^k \mathbf{EX}\ \mathbf{EU}\Big(Q_\psi, Z \cap Q_{\psi_i}\Big) \end{array} \right\} \tag{3.16}$$

где фигурные скобки изображают операцию пересечения множеств.

Для обоснования равенства (3.16) мы отдельно докажем, что его левая часть содержится в его правой части, и наоборот.

1. Утверждение о том, что $Q^F_{\mathrm{EG}\psi}$ содержится в правой части (3.16), следует из того, что $Q^F_{\mathrm{EG}\psi}$ является FP оператора в правой части (3.16), т.е.

$$Q^F_{\mathrm{EG}\psi} \ = \ \left\{ \begin{array}{l} Q_\psi \\ \bigcap_{i=1}^k \mathbf{EX}\ \mathbf{EU}\Big(Q_\psi, Q^F_{\mathrm{EG}\psi} \cap Q_{\psi_i}\Big) \end{array} \right\}$$

Данное равенство верно потому, что его левая и правая части состоят из всех состояний q, из которых существует fair путь $\pi \subseteq Q_\psi$.

2. Обратное включение следует из того, что если Z – FP оператора из правой части (3.16), т.е.

$$Z = \left\{ \begin{array}{l} Q_\psi \\ \bigcap_{i=1}^k \mathbf{EX}\ \mathbf{EU}\Big(Q_\psi, Z \cap Q_{\psi_i}\Big) \end{array} \right\}$$

то из каждого $q \in Z$ существует fair путь $\pi \subseteq Q_\psi$, т.е. $Z \subseteq Q^F_{\mathrm{EG}\psi}$.

Отметим, что если

- задача fair-MC-CTL состоит не в нахождении множества $Q^F_{\mathrm{EG}\psi}$, а в вычислении значения

$$q^F(\mathbf{EG}\psi) \tag{3.17}$$

при описанных выше допущениях (3.15), и

- в случае, когда значение (3.17) равно 1, требуется предъявить fair путь $\pi \subseteq Q_\psi$ из q в виде (3.10),

то можно пытаться строить данный путь сразу, ещё до завершения вычисления GFP (3.16):

- при первом вычислении внутренней FP

$$\mathbf{EU}\Big(Q_\psi, Z \cap Q_{\psi_1}\Big)$$

мы порождаем последовательность

$$Q^1_0 \subseteq Q^1_0 \subseteq Q^2_0 \subseteq \ldots$$

которую мы запоминаем, и используя которую мы находим конечный путь $\rho_1 \subseteq Q_\psi$ из q в состояние $q_1 \in Z \cap Q_{\psi_1}$

- затем таким же образом строим конечный путь $\rho_2 \subseteq Q_\psi$ из q_1 в состояние $q_2 \in Z \cap Q_{\psi_2}$

- и т.д.

Если искомый путь существует, то его можно построить, используя определённые выше пути $\rho_1, \rho_2, \ldots$

3.5 μ–исчисление

CTL (и некоторые другие логики, например, PDL) можно вложить в более мощную логику, называемую **μ–исчислением**. Это, в частности, позволяет свести задачу MC-CTL к некоторой задаче для μ–исчисления.

μ–исчисление позволяет описывать свойства СП с несколькими отношениями перехода, т.е. СП вида

$$(\mathcal{P},\ Q,\ \{\delta_a \mid a \in T\},\ L,\ Q^0) \tag{3.18}$$

где T – некоторое фиксированное множество, элементы которого называются **переходами**, и для каждого перехода $a \in T \quad \delta_a \subseteq Q^2$.

3.5.1 μ–формулы

Мы будем предполагать, что задано множество RV, элементы которого называются **реляционными переменными (РП)**.

Множество формул μ–исчисления (которые мы будем называть **μ–формулами**) обозначается символом Φ_μ. Данное множество обладает свойствами темпоральных логик, изложенными в пункте 3.1, и кроме того

1. $RV \subseteq \Phi_\mu$
2. для каждого $a \in T$ и каждой μ–формулы φ
$$[a]\varphi \in \Phi_\mu \quad \text{и} \quad \langle a\rangle\varphi \in \Phi_\mu$$
3. для каждой РП Z и каждой μ–формулы φ
$$\mu Z.\varphi \in \Phi_\mu \quad \text{и} \quad \nu Z.\varphi \in \Phi_\mu$$

Вхождения РП в μ–формулы бывают **свободными** и **связанными**:

- вхождение РП Z в μ–формулу Z – свободное,
- если φ имеет вид
$$\overline{\psi}, \quad \psi \wedge \eta, \quad \psi \vee \eta, \quad [a]\psi, \quad \langle a\rangle\psi$$
то каждому вхождению каждой РП Z в φ соответствует некоторое вхождение Z в ψ или η, и каждое вхождение Z в φ имеет тот же статус (свободное или связанное), который имеет соответствующее вхождение Z в ψ или η
- если φ имеет вид $\mu Z.\psi$ или $\nu Z.\psi$, то
  - все свободные вхождения Z в ψ (а также вхождение Z рядом с μ и ν) становятся связанными в φ, и
  - все остальные вхождения РП в φ имеют тот же статус, который имеют соответствующие им вхождения этих РП в ψ.

μ–формула называется **правильной**, если для каждой её подформулы вида $\mu Z.\varphi$ или $\nu Z.\varphi$ число отрицаний в подформуле φ, располагающихся над каждым свободным вхождением Z в φ, является чётным.

Ниже все рассматриваемые μ–формулы предполагаются правильными.

3.5.2 Значения μ–формул

Означиванием РП в СП (3.18) называется отображение ζ вида

$$\zeta : RV \to \mathbf{2}^Q \tag{3.19}$$

Значением μ–формулы φ в СП (3.18) на означивании (3.19) называется подмножество $\zeta(\varphi) \subseteq Q$ определяемое рекурсивно следующим образом:

1. $\forall\, p \in \mathcal{P} \quad \zeta(p) \stackrel{\text{def}}{=} Q_p$
2. значения РП определяются означиванием (3.19)
3. - $\zeta(\mathbf{1}) \stackrel{\text{def}}{=} Q, \quad \zeta(\mathbf{0}) \stackrel{\text{def}}{=} \emptyset$
   - $\zeta(\ \overline{\psi}\ ) \stackrel{\text{def}}{=} Q \setminus \zeta(\psi)$
   - $\zeta(\psi \wedge \eta) \stackrel{\text{def}}{=} \zeta(\psi) \cap \zeta(\eta)$
   - $\zeta(\psi \vee \eta) \stackrel{\text{def}}{=} \zeta(\psi) \cup \zeta(\eta)$
4. - $\zeta(\langle a\rangle\psi) \stackrel{\text{def}}{=} \{q \in Q \mid \delta_a(q) \cap \zeta(\psi) \neq \emptyset\}$
   - $\zeta([a]\psi) \stackrel{\text{def}}{=} \{q \in Q \mid \delta_a(q) \subseteq \zeta(\psi)\}$
5. - $\zeta(\mu Z.\psi) \stackrel{\text{def}}{=} \mu Z.\Big(A \mapsto \zeta[Z := A](\psi)\Big)$
   - $\zeta(\nu Z.\psi) \stackrel{\text{def}}{=} \nu Z.\Big(A \mapsto \zeta[Z := A](\psi)\Big)$

   где знакосочетание
   $$A \mapsto \zeta[Z := A](\psi) \tag{3.20}$$
   обозначает монотонный оператор вида (3.13), сопоставляющий каждому подмножеству $Q' \subseteq Q$ подмножество
   $$\zeta[Z := Q'](\psi)$$
   где $\zeta[Z := Q']$ обозначает означивание, отличающееся от ζ лишь на РП Z, которой означивание $\zeta[Z := Q']$ сопоставляет значение Q'.

   Монотонность оператора (3.20) следует из правильности ψ.

Заметим, что значение $\zeta(\varphi)$ зависит только от значений ζ на тех РП, которые имеют свободные вхождения в φ. В частности, если в φ нет ни одного свободного вхождения РП (такие формулы называются **замкнутыми**) то её значение в СП (3.18) является одним и тем же для всех означиваний. Данное значение называется просто **значением** φ в СП (3.18).

Если значения μ–формул φ и ψ совпадают на всех означиваниях во всех СП вида (3.18), то мы будем обозначать этот факт знакосочетанием $\varphi = \psi$.

Нетрудно доказать, что имеют место соотношения

$$\begin{array}{ll} \overline{[a]\varphi} = \langle a\rangle\overline{\varphi} & \overline{\mu Z.\varphi} = \nu Z.\overline{\varphi(\ \overline{Z}\ )} \\ \overline{\langle a\rangle\varphi} = [a]\overline{\varphi} & \overline{\nu Z.\varphi} = \mu Z.\overline{\varphi(\ \overline{Z}\ )} \end{array} \tag{3.21}$$

где формула $\varphi(\ \overline{Z}\ )$ получается из φ заменой каждого свободного вхождения Z на $\overline{Z}$.

3.5.3 Ускоренное вычисление значений μ–формул

Определение значения $\zeta(\varphi)$ в пункте 3.5.2 является также и алгоритмом вычисления этого значения. Этот алгоритм имеет сложность $O(|S|^{|\varphi|})$, где S – СП, в которой вычисляется значение $\zeta(\varphi)$.

Вычисление значений формул вида $\mu Z.\psi$ и $\nu Z.\psi$ можно ускорить, если учесть, что в качестве первоначальной аппроксимации для них можно взять не только $\mathbf{0}$ или $\mathbf{1}$, а любое подмножество $Q' \subseteq Q$, обладающее соответственно свойством

$$Q' \subseteq \zeta(\mu Z.\psi) \quad \text{или} \quad Q' \supseteq \zeta(\nu Z.\psi)$$

Основанный на этой идее алгоритм вычисления значения $\zeta(\varphi)$ для формулы φ вида $\mu Z.\psi$ выглядит следующим образом.

1. Пронесём в φ все отрицания вниз, используя законы де Моргана и соотношения (3.21). Получившуюся формулу обозначим тем же символом φ.

   Из определения правильности следует, что после этого пронесения все отрицания будут располагаться только над утверждениями.

2. Обозначим символом M список всех начинающихся с μ подформул формулы φ (включая саму φ), которые не содержатся в подформулах, начинающихся с ν.

   Каждой формуле $\eta \in M$ сопоставим новые РП A_η и B_η, в которых будут храниться промежуточные результаты вычисления значения η, и инициализируем A_η значением $\mathbf{0}$. Каждый раз, когда значение РП A_η будет обновляться, её старое значение будет записываться в B_η.

3. Далее работает цикл

   $$\textbf{do} \left\{ \begin{array}{l} B_\varphi := A_\varphi \\ A_\varphi := \zeta[Z := A_\varphi](\psi) \end{array} \right\} \textbf{while} \begin{array}{l} \exists \eta \in M : \\ B_\eta \neq A_\eta \end{array}$$
   $$\textbf{return } A_\varphi$$

   причём на каждом шаге цикла вычисление значения

   $$\zeta[Z := A_\varphi](\psi) \qquad (3.22)$$

   делается не совсем так, как это предписывалось рекурсивным определением в пункте 3.5.2.

   Отличие заключается только в способе вычисления значений подформул $\eta \in M$, и выглядит следующим образом.

   Каждый раз, когда в процессе вычисления (3.22) дело доходит до необходимости вычислить значение вида $\zeta'(\eta)$ для некоторой формулы $\eta \in M$, и формула η имеет вид $\mu Z_\eta.\psi_\eta$, в качестве требуемого значения возвращается

   $$\zeta'[Z_\eta := A_\eta](\psi_\eta) \qquad (3.23)$$

   (вычисляемое точно таким же модифицированным алгоритмом), и именно это значение заносится в РП A_η.

   Отметим, что значения подформул, вычисляемые модифицированным алгоритмом, всегда являются подмножествами значений этих подформул, вычисляемых алгоритмом из пункта 3.5.2.

   Для доказательства корректности данного алгоритма следует учесть, что при каждом обновлении содержимого РП A_η старое значение содержится в новом.

Вычисление значения формул вида $\nu Z.\psi$ может быть произведено двойственным образом.

Данный алгоритм имеет сложность $O(|S|^d)$, где d – **глубина чередования** формулы φ, определяемая как максимальная длина последовательности $\psi_1, \ldots, \psi_k$ подформул формулы φ, каждая из которых начинается с μ или ν, и для каждого $i = 1, \ldots, k-1$

- ψ_{i+1} является подформулой формулы ψ_i
- если ψ_i начинается с μ, то ψ_{i+1} начинается с ν, и если ψ_i начинается с ν, то ψ_{i+1} начинается с μ.

3.5.4 Вложение CTL в μ–исчисление

Пусть a – некоторый переход из множества T.

Каждой CTL–формуле φ, в которую входят только CTL–операторы вида (3.6), можно сопоставить замкнутую μ–формулу φ_μ, получаемую из φ заменой в ней подформул, начинающихся с CTL–операторов, на замкнутые μ–формулы, по следующему правилу:

1. $\mathbf{EX}\psi$ заменяется на $\langle a \rangle \psi$
2. $\mathbf{EU}(\psi, \eta)$ заменяется на $\mu Z.\Big(\eta \vee \big(\psi \wedge \langle a \rangle Z\big)\Big)$
3. $\mathbf{EG}\psi$ заменяется на $\nu Z.\big(\psi \wedge \langle a \rangle Z\big)$

Пусть S – СП вида (2.6). Определим СП S_μ вида (3.18) как СП с такими же множеством состояний и оценкой, что и у S, и, кроме того, $\delta_a = \delta$. Нетрудно доказать, что для каждой CTL–формулы φ её значение Q_φ в S совпадает со значением φ_μ в S_μ.

Таким образом, задача MC-CTL может быть сведена к задаче вычисления значения μ-формул.

Глава 4

Символьные вычисления

4.1 Представление множеств булевозначными выражениями

Пусть множество Q состоит из означиваний вида

$$\xi : V \to \mathcal{D} \tag{4.1}$$

где V – некоторое множество переменных, и $\mathcal{D}$ – множество их значений.

Каждому булевозначному выражению e, зависящему от переменных из множества V, соответствует подмножество $Q_e \subseteq Q$, определяемое следующим образом:

$$Q_e \stackrel{\text{def}}{=} \{\xi \in Q \mid \xi(e) = 1\}$$

Если для подмножества $Q' \subseteq Q$ существует булевозначное выражение e, такое, что

$$Q' = Q_e \tag{4.2}$$

то мы будем говорить, что e является **символьным представлением** подмножества Q' (или просто что e **представляет** подмножество Q').

Представление подмножества $Q' \subseteq Q$ в виде (4.2) наиболее эффективно в том случае, когда размер выражения e существенно меньше числа элементов в Q'.

Некоторым операциям над множествами означиваний соответствуют операции над представляющими эти множества булевозначными выражениями, например,

$$\begin{aligned} Q_{e_1} \cap Q_{e_2} &= Q_{e_1 \wedge e_2} \\ Q_{e_1} \cup Q_{e_2} &= Q_{e_1 \vee e_2} \\ Q \setminus Q_e &= Q_{\neg e} \end{aligned}$$

Вычисления над множествами означиваний, в которых вместо теоретико-множественных операций производятся соответствующие им операции над представляющими эти множества булевозначными выражениями, мы будем называть **символьными вычислениями**.

4.2 Задача SMC-CTL

Задача **SMC-CTL** (Symbolic Model Checking), заключается том, чтобы по заданным CTL-формуле φ и СП S, в которой

- состояниями являются означивания вида (4.1)
- отношение перехода представлено некоторым булевозначным выражением $\delta(V, V')$, зависящим от переменных из V и их штрихованных дубликатов, и имеет вид

  $$\{(\xi, \xi') \in Q^2 \mid \delta(\xi, \xi') = 1\}$$

  (определение значения выражения $\delta(V, V')$ на паре означиваний было приведено в пункте 2.4.3)
- для каждого утверждения p множество Q_p представлено некоторым булевозначным выражением $e(p)$

вычислить булевозначное выражение $e(\varphi)$, представляющее множество Q_φ.

Как и раньше, без ограничения общности мы можем предполагать, что в φ могут входить лишь CTL–операторы вида (3.6).

Идея излагаемого ниже алгоритма решения задачи SMC-CTL заключается в том, что выбирается некоторый класс $\mathcal{C}$ булевозначных выражений, который обладает следующими свойствами:

- существует алгоритм проверки эквивалентности выражений из класса $\mathcal{C}$
  (напомним, что выражения эквивалентны, если они представляют одно и то же множество означиваний)
- $\mathcal{C}$ содержит выражения, эквивалентные константам 0 и 1
- на $\mathcal{C}$ реализованы булевские операции (т.е. имеются алгоритмы, вычисляющие по выражениям $e_1, e_2 \in \mathcal{C}$ выражения из класса $\mathcal{C}$, эквивалентные выражениям

  $$\overline{e}, \quad e_1 \wedge e_2, \quad e_1 \vee e_2$$

- на $\mathcal{C}$ реализована операция **EX**, которая вычисляет по выражению $e \in \mathcal{C}$ выражение

  $$\mathbf{EX}(e) \stackrel{\text{def}}{=} \exists V' \big(\delta(V, V') \wedge e'\big) \tag{4.3}$$

где выражение e' получается из e заменой каждой входящей в него переменной $x \in V$ на её штрихованный дубликат x'.

Если исходные данные задачи SMC-CTL представлены выражениями из класса $\mathcal{C}$, то для вычисления выражения $e(\varphi)$ можно использовать следующий рекурсивный алгоритм.

1. Если $\varphi = p \in \mathcal{P}$, то выражение $e(p)$ уже известно.

2. Если φ является булевой комбинацией своих подформул, то выражение $e(\varphi)$ получается применением соответствующей булевой операции к выражениям, которые соответствуют этим подформулам.

3. Если φ начинается с CTL-оператора, то $e(\varphi)$ вычисляется по следующим правилам:

$$e(\mathbf{EX}\psi) = \mathbf{EX}\big(e(\psi)\big)$$
$$e(\mathbf{EG}\psi) = \nu Z.\Big(e(\psi) \wedge \mathbf{EX}(Z)\Big)$$
$$e(\mathbf{EU}(\psi,\eta)) = \mu Z.\Big(e(\eta) \vee \big(e(\psi) \wedge \mathbf{EX}(Z)\big)\Big)$$

Второе и третье выражение вычисляются по алгоритму, изложенному в конце пункта 3.4.1, т.е.

$$e_1 := \left\{ \begin{array}{l} 1 \quad \text{(для второго выражения)} \\ 0 \quad \text{(для третьего выражения)} \end{array} \right.$$
$$\textbf{do} \left\{ \begin{array}{l} e_2 := e_1 \\ e_1 := \mathcal{F}(e_1) \end{array} \right\} \textbf{while } (e_2 \neq e_1)$$
$$\textbf{return } e_1$$

где $\mathcal{F}(Z)$ совпадает с записью в скобках после операторов μZ и νZ во втором и третьем выражениях.

4.3 Binary Decision Diagrams

4.3.1 Понятие BDD

Одним из классов $\mathcal{C}$ булевозначных выражений со свойствами, изложенными в пункте 4.2, является класс **двоичных решающих диаграмм**, сокращённо обозначаемых знакосочетанием **BDD** (Binary Decision Diagram).

BDD можно использовать для представления только таких множеств означиваний вида (4.1), в которых каждая переменная $x \in V$ имеет тип `bool`.

Если же тип некоторых переменных из V – не `bool`, то

- означивания вида (4.1) можно преобразовать в означивания вида

$$\hat{V} \to \{0,1\} \tag{4.4}$$

где $\hat{V}$ получается из V заменой каждой переменной $x \in V$, тип которой - не `bool`, на $|x|$ новых переменных типа `bool`, где $|x|$ – количество битов, необходимых для записи значений типа $\tau(x)$, и

- для представления множеств означиваний вида (4.1) можно использовать BDD, представляющие множества означиваний вида (4.4).

Ниже мы предполагаем, что каждая переменная x из V имеет тип `bool`.

BDD можно определить в том синтаксисе, который изложен в пункте 2.1.1. Однако наиболее наглядно BDD представляются в виде графов, поэтому мы будем определять BDD сразу в графовой форме.

BDD представляет собой ациклический граф e с выделенной вершиной $Root(e)$, называемой **корнем**. Вершины BDD делятся на два класса - терминальные и нетерминальные.

Каждая терминальная вершина имеет метку 0 или 1. Из терминальных вершин не выходит ни одного ребра.

Каждая нетерминальная вершина v помечена некоторой переменной $l(v) \in V$. Из неё выходят два ребра, одно из которых имеет метку 0, а другое - метку 1.

Вычисление значения BDD на означивании ξ происходит посредством прохода по этой BDD, начиная с корневой вершины. В каждый момент прохода

- если мы в этот момент находимся в нетерминальной вершине, то мы переходим к следующей вершине по ребру с меткой $\xi(l(v))$, и

- если мы в этот момент находимся в терминальной вершине, то вычисление заканчивается, и в качестве результата выдаётся метка этой вершины.

Две BDD называются **изоморфными**, если существует взаимно-однозначное соответствие между множествами их вершин, такое, что

- корневые вершины соответствуют друг другу,

- метки соответствующих вершин совпадают, и

- если в одной BDD пара вершин соединена ребром, то соответствуюшая ей пара в другой BDD тоже соединена ребром с той же меткой.

Очевидно, что если две BDD изоморфны, то они эквивалентны.

4.3.2 Редукция BDD

Редукцией BDD называется преобразование её в эквивалентную ей BDD меньшего размера.

Существуют три операции редукции.

1. Если BDD содержит пару v_1, v_2 вершин со следующими свойствами:
   - $l(v_1) = l(v_2)$, и
   - если v_1 и v_2 нетерминальны, то концы выходящих из v_1 и v_2 рёбер с одинаковыми метками совпадают.

   то можно
   - удалить v_1 и выходящие из неё рёбра, и
   - рёбра, входящие в v_1, перенаправить в v_2.
2. Если BDD содержит вершину v, такую, что выходящие из v рёбра имеют один и тот же конец v_1, то можно
   - удалить v и выходящие из неё рёбра, и
   - входящие в v рёбра перенаправить в v_1.
3. Если BDD содержит недостижимую вершину v (т.е. такую вершину v, в которую не существует пути из корня), то можно удалить эту вершину и все связанные с нею рёбра.

Редуцированием BDD называется применение к ней операций редукции до тех пор, пока это возможно.

BDD называется **нередуцируемой**, если к ней невозможно применить никакую операцию редукции.

4.3.3 Порядок переменных в BDD

Пусть на множестве V переменных задан некоторый линейный порядок R.

BDD называется **согласованной** с R, если для каждого ребра из одной нетерминальной вершины v_1 в другую нетерминальную вершину v_2 имеет место неравенство

$$l(v_1) < l(v_2)$$

Можно доказать, что если две нередуцируемые BDD согласованы с одним и тем же порядком на V и эквивалентны, то они изоморфны.

Пусть заданы некоторое множество означиваний M вида (4.1), и некоторый порядок R на V.

Наименьший возможный размер BDD, которая представляет M и согласована с R, обозначается знакосочетанием

$$size(M, R)$$

Например, если представляемое множество M состоит из всех означиваний, на которых истинно выражение

$$\bigwedge_{i=1}^{k} (x_i \leftrightarrow y_i)$$

то

- $size(M, R_1) = 3k + 2$, где порядок R_1 имеет вид

$$x_1 < y_1 < \ldots < x_k < y_k$$

- $size(M, R_2) = 3 \cdot 2^k - 1$, где порядок R_2 имеет вид

$$x_1 < \ldots < x_k < y_1 < \ldots < y_k$$

Порядок R на V называется **оптимальным** для представления M, если для любого порядка R' на V

$$size(M, R) \leq size(M, R')$$

Задача проверки оптимальности выбранного порядка является NP–полной.

Ниже все рассматриваемые BDD предполагаются согласованными с некоторым фиксированным порядком на множестве переменных.

Это, в частности, обеспечивает возможность проверки эквивалентности двух BDD, которая, ввиду вышесказанного, может быть произведена путём

- редуцирования обеих BDD, и
- проверке изоморфности получившихся BDD.

4.3.4 Операции на BDD

В этом пункте определяются булевские и некоторые другие операции на BDD. После выполнения действий, изложенных в определении каждой операции, необходимо редуцировать получившуюся BDD.

Константы 0 и 1

BDD, представляющая константу 0 или 1, состоит из одной вершины, помеченной этой константой.

Подстановка значения вместо переменной

Пусть e – некоторая BDD.

Для каждой её нетерминальной вершины v мы будем обозначать символом v_b (где $b = 0$ или 1) конец выходящего из v ребра с меткой b.

Знакосочетание

$$[x := b]e \tag{4.5}$$

(где $x \in V$ и $b = 0$ или 1) обозначает BDD, получаемую из e удалением всех вершин с меткой x и выходящих из них рёбер, причём перед удалением каждой такой вершины v каждое ребро, входящее в v, перенаправляется в v_b.

В том случае, когда удаляемая вершина является корнем в e, корнем в (4.5) будет v_b. В этом случае (4.5) является подграфом в e.

Нетрудно видеть, что для каждого означивания ξ имеет место соотношение

$$\xi\ ([x := b]e)\ =\ (\xi[x := b])\ (e)$$

где означивание $\xi[x := b]$ отличается от ξ лишь значением на переменной x, которой оно сопоставляет значение b.

Отрицание

Для каждой BDD e её отрицание $\overline{e}$ получается из e заменой меток терминальных вершин: 0 заменяется на 1, а 1 - на 0.

Бинарные булевы операции

Пусть $*$ обозначает операцию $\wedge$ или $\vee$. Для каждой пары BDD e_1, e_2 их булева композиция $e_1 * e_2$ определяется следующим образом.

Если одна из BDD e_1, e_2 является константой, то $e_1 * e_2$ совпадает либо с e_1, либо с e_2:

$$0 \wedge e = 0, \quad 0 \vee e = e, \quad 1 \wedge e = e, \quad 1 \vee e = 1$$

Пусть обе BDD e_1 и e_2 не константы. Обозначим метки вершин $Root(e_1)$ и $Root(e_2)$ символами x и y соответственно.

Если $x = y$, то искомая BDD имеет корень с меткой x, из которого выходит

- ребро с меткой 0 в BDD

$$([x := 0]e_1) \ * \ ([x := 0]e_2)$$

  и

- ребро с меткой 1 в BDD

$$([x := 1]e_1) \ * \ ([x := 1]e_2)$$

Если $x < y$, то искомая BDD имеет корень с меткой x, из которого выходит

- ребро с меткой 0 в BDD

$$([x := 0]e_1) \ * \ e_2$$

  и

- ребро с меткой 1 в корень BDD

$$([x := 1]e_1) \ * \ e_2$$

Если $y < x$, то искомая BDD определяется аналогично.

Заметим, что все участвующие в данном определении вспомогательные BDD (типа $[x := b]e_i$) являются подграфами исходных BDD, т.е. полностью определяются вершинами исходных BDD, являющимися корневыми в данных вспомогательных BDD. Это позволяет во всех выражениях, в которых участвуют вспомогательные BDD, вместо самих этих BDD записывать только определяющие их вершины. Используя данное соображение, нетрудно доказать, что сложность задачи вычисления бинарных булевских операций на BDD имеет верхнюю оценку $O(|e_1| \cdot |e_2|)$.

Реляционные произведения

Для вычисления BDD $\mathbf{EX}(e)$, определяемой соотношением (4.3), мы реализуем более общую операцию **реляционного произведения (РП)**, которая по паре BDD

$$f(X, Y) \quad \text{и} \quad g(Y, Z)$$

(где X, Y, Z – попарно непересекающиеся списки булевых переменных, f содержит переменные из X и Y, а g – переменные из Y и Z) строит BDD

$$\exists\, Y \left(f(X, Y) \ \wedge \ g(Y, Z) \right) \tag{4.6}$$

где $\exists\, Y$ является сокращением знакосочетания

$$\exists y_1 \ldots \exists y_k$$

если Y имеет вид $(y_1, \ldots, y_k)$. Для каждой BDD e знакосочетание $\exists y\, e$ является сокращённой записью BDD

$$[y := 0]e \ \vee \ [y := 1]e$$

Операция вычисления (4.6) обозначается знакосочетанием **Rel_Prod** и имеет аргументы f, g, Y. В определении этой операции используется вспомогательная переменная $Cache$, в которой хранятся четвёрки вида

$$(\ f,\ g,\ Y,\ \mathbf{Rel\_Prod}(f, g, Y)\ )$$

представляющие собой вычисленные значения функции **Rel_Prod** для некоторых значений её аргументов.

Операция **Rel_Prod** определяется рекурсивно следующим образом.

Rel_Prod (f, g, Y) :=

1. если $f = 0$ или $g = 0$, то **return** 0
2. если $f = 1$ и $g = 1$, то **return** 1
3. если $(f, g, Y, h) \in Cache$ то **return** h
4. иначе выполняем следующую последовательность действий:
   - a := maxvar (f) (переменная в f с максимальным номером)
   - b := maxvar (g)
   - c := max (a, b)
   - h_0 := **Rel_Prod** $([c := 0]f,\ [c := 0]g,\ Y)$
   - h_1 := **Rel_Prod** $([c := 1]f,\ [c := 1]g,\ Y)$
   - $h := \begin{cases} h_0 \vee h_1 & \text{если } c \in Y \\ (z \wedge h_1) \vee (\overline{z} \wedge h_0) & \text{иначе} \end{cases}$
   - добавляем (f, g, Y, h) в $Cache$
   - **return** (h)

В наихудшем случае этот алгоритм имеет экспоненциальную сложность.

В том случае, когда

- вычисляется РП вида

$$\exists Y \left\{ \begin{array}{l} \delta(X,Y) \\ e(Y) \end{array} \right\} \qquad (4.7)$$

(которое представляет основной интерес для задачи МС), и

- δ является комбинацией отношений δ_i, многие из которых зависят от небольшого числа переменных из Y

то вычисление BDD (4.7) может быть ускорено.

Например, если $\delta = \delta_1 \vee \ldots \vee \delta_n$ (что имеет место, например, в том случае, когда δ определяется по программной системе согласно правилам, изложенным в пункте 2.4.3), то вместо BDD (4.7) можно вычислять эквивалентную ей BDD

$$\bigvee_{i=1}^{n} \exists Y \left\{ \begin{array}{l} \delta_i(X,Y) \\ e(Y) \end{array} \right\} \qquad (4.8)$$

В частности, если моделируемая система представляет собой схему из функциональных элементов, компоненты которой работают последовательно, то

$$\delta_i \;=\; \left\{ \begin{array}{l} y_i \leftrightarrow f_i(X) \\ \bigwedge\limits_{j \neq i} (y_j = x_j) \end{array} \right\}$$

и (4.8) можно переписать в виде

$$\bigvee_{i=1}^{n} \exists y_i \left\{ \begin{array}{l} y_i \leftrightarrow f_i(X) \\ e(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n) \end{array} \right\} \qquad (4.9)$$

В другом случае, когда δ представляет собой конъюнкцию выражений, их надо сначала объединить в группы, относя в одну группу те выражения, которые связаны по смыслу. Выражение, соответствующее каждой группе, должно быть представлено единой BDD. Пусть выражения, соответствующие этим группам, имеют вид $\delta_1, \ldots, \delta_n$, тогда

$$\delta \;=\; \delta_1 \wedge \ldots \wedge \delta_n$$

Для каждого $i \in \{1, \ldots, n\}$ обозначим символом $Y(\delta_i)$ множество переменных из Y, которые входят в δ_i.

Если представить множество Y в виде разбиения $Y_1 \sqcup \ldots \sqcup Y_n$, где

$$\begin{array}{l} Y_1 \stackrel{\text{def}}{=} Y(\delta_1) \\ Y_2 \stackrel{\text{def}}{=} Y(\delta_2) \setminus Y_1 \\ Y_3 \stackrel{\text{def}}{=} Y(\delta_3) \setminus (Y_1 \cup Y_2) \\ \ldots \\ Y_n \stackrel{\text{def}}{=} Y(\delta_n) \setminus (Y_1 \cup \ldots \cup Y_{n-1}) \end{array}$$

то вместо BDD (4.7) можно вычислять эквивалентную ей BDD

$$\exists Y_1 \left\{ \begin{array}{l} \delta_1 \\ \exists Y_2 \left\{ \begin{array}{l} \delta_2 \\ \ldots \exists Y_{n-1} \left\{ \begin{array}{l} \delta_{n-1} \\ \exists Y_n \left\{ \begin{array}{l} \delta_n \\ e(Y) \end{array} \right\} \end{array} \right\} \end{array} \right\} \end{array} \right\}$$

Вычисление последней BDD производится путём вычисления последовательности BDD $e'_n, \ldots, e'_1$, где

$$\forall i = n, \ldots, 1 \qquad e'_i \stackrel{\text{def}}{=} \exists Y_i \left\{ \begin{array}{l} \delta_i \\ e'_{i+1} \end{array} \right\}, \qquad e'_{n+1} \stackrel{\text{def}}{=} e(Y)$$

Размеры BDD $e'_n, \ldots, e'_1$ определяются числом переменных в этих BDD и зависят от порядка конъюнктивных членов в выражении δ. Задача поиска наилучшего такого порядка является NP-полной. Порядок, близкий к оптимальному, можно найти, например, следующим образом.

1. Составляем список C конъюнктивных членов в δ.
2. Для каждой переменной y, входящей в какое-либо из множеств вида $Y(\delta')$, где $\delta' \in C$, вычисляем её "стоимость", которая может иметь вид

$$\max_{\delta' \in C,\ Y(\delta') \ni y} |Y(\delta')| \quad \text{или} \quad \sum_{\delta' \in C,\ Y(\delta') \ni y} |Y(\delta')|$$

3. В качестве первого конъюнктивного члена берём такое δ', что $Y(\delta')$ содержит переменную с наименьшей стоимостью.
4. Исключаем δ' из списка C.
5. Следующий по порядку конъюнктивный член ищем среди оставшихся в списке C по точно такому же принципу.

Глава 5

Логика LTL

5.1 Бесквантворные темпоральные формулы

Совокупность **бесквантворных темпоральных формул (БТФ)** представляет собой темпоральную логику со следующими дополнительными правилами:

- если ψ – БТФ, то $\mathbf{X}\psi$, $\mathbf{F}\psi$, $\mathbf{G}\psi$ – БТФ
- если ψ и η – БТФ, то $\mathbf{U}(\psi,\eta)$ и $\mathbf{R}(\psi,\eta)$ – БТФ

Жирные символы ($\mathbf{X}$, и т.д.) в данных формулах называются **темпоральными операторами**.

Пусть $\pi = (q_0, q_1, \ldots)$ – путь в некоторой СП S. Для каждого $i \geq 0$ символ π_i обозначает "хвост" пути π:

$$\pi_i \;=\; (q_i,\; q_{i+1},\; \ldots)$$

Для каждой БТФ φ её **значением** $\pi(\varphi)$ на пути π является булева константа 1 или 0, которая определяется индуктивно:

1. если $\varphi = p \in \mathcal{P}$, то $\pi(\varphi) \stackrel{\text{def}}{=} q_0(\varphi)$
   (данное значение уже определено в СП S)
2. $\pi(\mathbf{1}) = 1, \quad \pi(\mathbf{0}) = 0$
3. значения булевых комбинаций определяются стандартным образом
4. - $\pi(\mathbf{X}\psi) \stackrel{\text{def}}{=} \pi_1(\psi)$,
   - $\pi(\mathbf{F}\psi) = 1$, если $\exists i \geq 0 :\ \pi_i(\psi) = 1$
   - $\pi(\mathbf{G}\psi) = 1$, если $\forall i \geq 0 \quad \pi_i(\psi) = 1$
   - $\pi(\mathbf{U}(\psi,\eta)) = 1$, если $\exists i \geq 0$:
   $$\left\{ \begin{array}{l} \pi_i(\eta) = 1, \text{ и} \\ \forall\ j < i \quad \pi_j(\psi) = 1 \end{array} \right\} \qquad (5.1)$$
   - $\pi(\mathbf{R}(\psi,\eta)) = 1$, если $\forall i \geq 0$
   $$\left[\begin{array}{l} \pi_i(\eta) = 1, \quad \text{или} \\ \exists j < i :\ \pi_j(\psi) = 1 \end{array} \right]$$

Мы будем называть БТФ φ и ψ **эквивалентными** (и обозначать этот факт знакосочетанием $\varphi = \psi$), если для каждого пути π в каждой СП $\quad \pi(\varphi) = \pi(\psi)$.

Нетрудно доказать, что имеют место соотношения

- $\mathbf{F}\varphi = \mathbf{1U}\varphi$
- $\overline{\mathbf{X}\varphi} = \mathbf{X}\overline{\varphi}$
- $\overline{\mathbf{G}\varphi} = \mathbf{F}\overline{\varphi}$
- $\overline{\mathbf{U}(\varphi,\psi)} = \mathbf{R}(\overline{\varphi},\overline{\psi})$

Следовательно, для любой БТФ существует эквивалентная ей БТФ, в которую входят только связки $\neg, \vee, \mathbf{X}, \mathbf{U}$.

Кроме того, имеют место соотношения

$$\mathbf{U}(\psi,\eta) = \left[\begin{array}{l} \eta \\ \psi \wedge \mathbf{XU}(\psi,\eta) \end{array} \right]$$

$$\mathbf{R}(\psi,\eta) = \left\{ \begin{array}{l} \eta \\ \psi \vee \mathbf{XR}(\psi,\eta) \end{array} \right\}$$

5.2 LTL-формулы

LTL–формулой называется знакосочетание вида $\mathbf{A}\psi$ или $\mathbf{E}\psi$, где ψ – БТФ.

Для каждой СП S и каждого её состояния q **значение** LTL–формулы φ в q обозначается знакосочетанием $q(\varphi)$ и определяется следующим образом:

- $q(\mathbf{A}\psi) = 1$, если для каждого пути π из q
  $$\pi(\psi) = 1 \qquad (5.2)$$
- $q(\mathbf{E}\psi) = 1$, если существует путь π из q, такой, что имеет место (5.2).

Из этого определения следует, что для каждой БТФ ψ $\overline{\mathbf{A}\psi} = \mathbf{E}\overline{\psi}$.

Можно определить стандартным образом отношение эквивалентности CTL–формул и LTL–формул, и доказать, что

1. CTL–формула $\mathbf{AG}(\mathbf{EF}p)$ не эквивалентна ни одной LTL–формуле
2. LTL–формула $\mathbf{A}(\mathbf{FG}p)$ не эквивалентна ни одной CTL–формуле
3. дизъюнкция приведённых выше формул не эквивалентна ни одной LTL–формуле, и ни одной CTL–формуле.

5.3 Model checking для LTL

5.3.1 СП S_φ

Ниже мы предполагаем, что каждая рассматриваемая БТФ содержит связки только из множества $\{\neg, \vee, \mathbf{X}, \mathbf{U}\}$.

Для каждой БТФ φ знакосочетание $\langle\varphi\rangle$ обозначает наименьшее (по отношению включения) множество формул, удовлетворяющее следующим условиям:

1. $\langle\varphi\rangle$ содержит все подформулы формулы φ
2. если $\mathbf{U}(\psi,\eta) \in \langle\varphi\rangle$, то $\mathbf{XU}(\psi,\eta) \in \langle\varphi\rangle$

Множество $\langle\varphi\rangle$ называется **замыканием** φ. Ниже множество $\mathcal{P} \cap \langle\varphi\rangle$ будет обозначаться символом $\mathcal{P}_\varphi$.

Для каждой БТФ φ символ S_φ обозначает СП,

- состояниями которой являются функции вида

$$K : \langle\varphi\rangle \to \{0,1\} \tag{5.3}$$

удовлетворяющие следующим условиям:

$$\begin{aligned} &K(\overline{\psi}) = \overline{K(\psi)} \\ &K(\psi \vee \eta) = K(\psi) \vee K(\eta) \\ &K(\mathbf{U}(\psi,\eta)) = \left[\begin{array}{l} K(\eta) \\ K(\psi) \wedge K(\mathbf{XU}(\psi,\eta)) \end{array}\right] \end{aligned} \tag{5.4}$$

- для каждой пары K, K' состояний СП S_φ имеется ребро $K \to K'$, если для каждой формулы вида $\mathbf{X}\psi$ из $\langle\varphi\rangle$

$$K(\mathbf{X}\psi) = K'(\psi) \tag{5.5}$$

- для каждого $p \in \mathcal{P}_\varphi$ и каждого состояния K истинность p в K равна $K(p)$,
- начальными состояниями являются такие функции K, для которых $K(\varphi) = 1$
- в СП S_φ дополнительно задан список условий fairness, который имеет вид

$$(F_{\mathrm{U}(\psi,\eta)} \mid \mathbf{U}(\psi,\eta) \in \langle\varphi\rangle)$$

где для каждой формулы из $\langle\varphi\rangle$ вида $\mathbf{U}(\psi,\eta)$

$$F_{\mathrm{U}(\psi,\eta)} \stackrel{\mathrm{def}}{=} \{K : \langle\varphi\rangle \to \{0,1\} \mid K(\mathbf{U}(\psi,\eta)) \le K(\eta)\}$$

Нетрудно доказать, что путь $\kappa = (K_0, \ldots)$ в S_φ является fair тогда и только тогда, когда для каждого $i \ge 0$ и каждой формулы вида $\mathbf{U}(\psi,\eta)$ из $\langle\varphi\rangle$

$$K_i(\mathbf{U}(\psi,\eta)) \le \bigvee_{j \ge i} K_j(\eta)$$

Лемма.

Для каждого состояния K СП S_φ и каждого fair пути κ, выходящего из K, имеет место равенство

$$K(\varphi) = \kappa(\varphi)$$

Доказательство.

Докажем более общее утверждение: для каждого fair пути $\kappa = (K_0, \ldots)$ в S_φ имеет место соотношение

$$\forall\psi \in \langle\varphi\rangle, \quad \forall i \ge 0 \qquad K_i(\psi) = \kappa_i(\psi) \tag{5.6}$$

Доказательство этого соотношения мы проведём индукцией по структуре ψ.

1. $\forall p \in \mathcal{P}_\varphi \quad K_i(p) = \kappa_i(p)$ по определению значения БТФ на пути
2. $K_i(\overline{\psi}) = \overline{K_i(\psi)} = \overline{\kappa_i(\psi)} = \kappa_i(\overline{\psi})$
3. $K_i(\psi \vee \eta) = K_i(\psi) \vee K_i(\eta) =$
   $= \kappa_i(\psi) \vee \kappa_i(\eta) = \kappa_i(\psi \vee \eta)$
4. $K_i(\mathbf{X}\psi) = K_{i+1}(\psi) = \kappa_{i+1}(\psi) = \kappa_i(\mathbf{X}\psi)$
5. для доказательства равенства

$$K_i(\mathbf{U}(\psi,\eta)) = \kappa_i(\mathbf{U}(\psi,\eta))$$

мы отдельно докажем два неравенства:

$$K_i(\mathbf{U}(\psi,\eta)) \le \kappa_i(\mathbf{U}(\psi,\eta)) \tag{5.7}$$

и

$$K_i(\mathbf{U}(\psi,\eta)) \ge \kappa_i(\mathbf{U}(\psi,\eta)) \tag{5.8}$$

Пусть (5.7) неверно, т.е.

$$K_i(\mathbf{U}(\psi,\eta)) = 1 \tag{5.9}$$

$$\kappa_i(\mathbf{U}(\psi,\eta)) = 0 \tag{5.10}$$

Согласно определению значения БТФ на пути, из (5.10) следуют соотношения

$$\kappa_i(\eta) = 0 \tag{5.11}$$

$$\kappa_i(\psi \wedge \mathbf{XU}(\psi,\eta)) = 0 \tag{5.12}$$

из которых, учитывая индуктивное предположение, мы получаем соотношения

$$K_i(\eta) = 0 \tag{5.13}$$

$$K_i(\psi) \wedge \kappa_{i+1}(\mathbf{U}(\psi,\eta)) = 0 \tag{5.14}$$

Из (5.4), (5.9), и (5.13) следует, что

$$K_i(\psi) \wedge K_{i+1}(\mathbf{U}(\psi,\eta)) = 1 \tag{5.15}$$

т.е.

$$K_i(\psi) = 1 \tag{5.16}$$

и

$$K_{i+1}(\mathbf{U}(\psi,\eta)) = 1 \tag{5.17}$$

Из (5.14) и (5.16) следует, что

$$\kappa_{i+1}(\mathbf{U}(\psi,\eta)) = 0 \tag{5.18}$$

Соотношения (5.17) и (5.18) представляют собой исходные соотношения (5.9) и (5.10) в которых вместо i написано $i+1$.

Следовательно, будут верны соотношения (5.9) и (5.10), в которых вместо i написано произвольное $j \geq i$. В частности, поскольку из (5.9) и (5.10) следует (5.13), то для каждого $j \geq i$ будет верно (5.13), в котором вместо i написано j. Учитывая (5.9), мы получаем противоречие с предположением о том, что путь κ – fair.

Теперь докажем обратное неравенство (5.8).

Выражение $\kappa_i(\mathbf{U}(\psi,\eta))$ по определению является дизъюнкцией выражений вида

$$\kappa_i(\psi) \wedge \ldots \wedge \kappa_{i+k-1}(\psi) \wedge \kappa_{i+k}(\eta) \tag{5.19}$$

где $k \geq 0$.

Для доказательства (5.8) достаточно доказать, что каждое из выражений (5.19) не превосходит

$$K_i(\mathbf{U}(\psi,\eta)) \tag{5.20}$$

По индуктивному предположению, (5.19) совпадает с

$$K_i(\psi) \wedge \ldots \wedge K_{i+k-1}(\psi) \wedge K_{i+k}(\eta) \tag{5.21}$$

Поскольку

$$K_{i+k}(\eta) \leq K_{i+k}(\mathbf{U}(\psi,\eta)) = K_{i+k-1}(\mathbf{X}\mathbf{U}(\psi,\eta))$$

то

$$\begin{array}{ll} & K_{i+k-1}(\psi) \wedge K_{i+k}(\eta) \leq \\ \leq & K_{i+k-1}(\psi) \wedge K_{i+k-1}(\mathbf{X}\mathbf{U}(\psi,\eta)) \leq \\ \leq & K_{i+k-1}(\mathbf{U}(\psi,\eta)) \end{array}$$

т.е. последние два члена в выражении (5.21) можно промажорировать выражением

$$K_{i+k-1}(\mathbf{U}(\psi,\eta))$$

Производя и далее подобные мажорирующие замены, в конце концов придём к желаемому выражению (5.20).

5.3.2 СП $S \times S_\varphi$

Для каждой СП $S = (\mathcal{P}, Q, \delta, L, Q^0)$ и каждой БТФ φ знакосочетание $S \times S_\varphi$ обозначает СП,

- состояниями которой являются пары вида (q, K), где $q \in Q$, K – состояние S_φ, и

$$\forall p \in \mathcal{P}_\varphi \quad K(p) = q(p)$$

- в СП $S \times S_\varphi$ имеется ребро $(q,K) \to (q',K')$, если $q \to q'$ и $K \to K'$
- для каждого состояния (q,K) СП $S \times S_\varphi$ и каждого $p \in \mathcal{P}$ значение p в (q,K) равно $K(p) \quad (= q(p))$.

Пусть $S = (\mathcal{P}, Q, \delta, L, Q^0)$ – некоторая СП.

Каждый путь $\pi = (q_0, \ldots)$ в S определяет fair путь $\kappa_\pi = (K_0, \ldots)$ в S_φ, где

$$\forall \psi \in \langle\varphi\rangle, \quad \forall i \geq 0 \qquad K_i(\psi) = \pi_i(\psi) \tag{5.22}$$

Если бы κ_π был не fair, то

$$\exists i_0 \geq 0 : \ \forall\, i \geq i_0 \quad K_i(\mathbf{U}(\psi,\eta)) \not\leq K_i(\eta)$$

т.е.

$$\forall\, i \geq i_0 \quad \pi_i(\mathbf{U}(\psi,\eta)) \not\leq \pi_i(\eta)$$

т.е.

$$\forall\, i \geq i_0 \quad \pi_i(\mathbf{U}(\psi,\eta)) = 1, \quad \pi_i(\eta) = 0$$

Это неверно, т.к. из $\pi_{i_0}(\mathbf{U}(\psi,\eta)) = 1$ следует, что

$$\exists\, i \geq i_0 : \quad \pi_i(\eta) = 1$$

что противоречит соотношению $\forall\, i \geq i_0 \quad \pi_i(\eta) = 0$.

Для каждого пути $\pi = (q_0, \ldots)$ в S обозначим символом σ_π путь $\pi \times \kappa_\pi$ в $S \times S_\varphi$, т.е. путь

$$((q_0, K_0), (q_1, K_1), \ldots) \tag{5.23}$$

Произвольный путь σ в $S \times S_\varphi$ мы будем называть fair, если последовательность его вторых компонентов является fair путём в S_φ.

Соответствие $\pi \mapsto \sigma_\pi$ между путями в S и fair путями в $S \times S_\varphi$ биективно. Это следует из того, что для каждого fair пути σ вида (5.23) имеет место соотношение (5.22) (в котором $\pi = (q_0, \ldots)$). Доказательство (5.22) мы проведём индукцией по структуре ψ.

1. $\forall p \in \mathcal{P}_\varphi \quad K_i(p) = q_i(p) = \pi_i(p)$
2. $K_i(\overline{\psi}) = \overline{K_i(\psi)} = \overline{\pi_i(\psi)} = \pi_i(\overline{\psi})$
3. $K_i(\psi \vee \eta) = K_i(\psi) \vee K_i(\eta) =$
   $= \pi_i(\psi) \vee \pi_i(\eta) = \pi_i(\psi \vee \eta)$
4. $K_i(\mathbf{X}\psi) = K_{i+1}(\psi) = \pi_{i+1}(\psi) = \pi_i(\mathbf{X}\psi)$
5. $K_i(\mathbf{U}(\psi,\eta)) =$

$$= \left[\begin{array}{l} K_i(\eta) \\ K_i(\psi) \wedge K_i(\mathbf{X}\mathbf{U}(\psi,\eta)) \end{array}\right] =$$

$$= \left[\begin{array}{l} K_i(\eta) \\ K_i(\psi) \wedge K_{i+1}(\mathbf{U}(\psi,\eta)) \end{array}\right] =$$

$$= \left[\begin{array}{l} K_i(\eta) \\ K_i(\psi) \wedge \left[\begin{array}{l} K_{i+1}(\eta) \\ K_{i+1}(\psi) \wedge K_{i+1}(\mathbf{X}\mathbf{U}(\psi,\eta)) \end{array}\right] \end{array}\right] =$$

$$= \left[\begin{array}{l} K_i(\eta) \\ K_i(\psi) \wedge K_{i+1}(\eta) \\ K_i(\psi) \wedge K_{i+1}(\psi) \wedge K_{i+1}(\mathbf{X}\mathbf{U}(\psi,\eta)) \end{array}\right] =$$

$$= \left[\begin{array}{l} K_i(\eta) \\ K_i(\psi) \wedge K_{i+1}(\eta) \\ K_i(\psi) \wedge K_{i+1}(\psi) \wedge K_{i+2}(\mathbf{U}(\psi,\eta)) \end{array} \right] =$$

$$= \ldots =$$

$$= \left[\begin{array}{l} \pi_i(\eta) \\ \pi_i(\psi) \wedge \pi_{i+1}(\eta) \\ \pi_i(\psi) \wedge \pi_{i+1}(\psi) \wedge \pi_{i+2}(\eta) \\ \ldots \\ \pi_i(\psi) \wedge \ldots \wedge \pi_{i+k-1}(\psi) \wedge \pi_{i+k}(\eta) \\ \pi_i(\psi) \wedge \ldots \wedge \pi_{i+k}(\psi) \wedge K_{i+k+1}(\mathbf{U}(\psi,\eta)) \end{array} \right]$$

где k – произвольное неотрицательное число.

Таким образом, имеет место неравенство

$$\pi_i(\mathbf{U}(\psi,\eta)) \leq K_i(\mathbf{U}(\psi,\eta))$$

Для обоснования обратного неравенства заметим, что из того, что σ – fair, следует соотношение

$$K_i(\mathbf{U}(\psi,\eta)) \leq \bigvee_{j \geq i} K_j(\eta) = \bigvee_{j \geq i} \pi_j(\eta) \qquad \blacksquare$$

Сильно связная компонента C в $S \times S_\varphi$ называется **fair**, если для каждого $(q,K) \in C$ и каждой формулы вида $\mathbf{U}(\psi,\eta) \in \langle\varphi\rangle$

$$K(\mathbf{U}(\psi,\eta)) \leq \bigvee_{(q',K') \in C} K'(\eta) \qquad (5.24)$$

Докажем, что для каждого состояния (q,K) системы $S \times S_\varphi$ следующие условия эквивалентны.

1. Существует fair путь из (q,K).
2. Существует конечный путь из (q,K) в состояние (q',K'), принадлежащее некоторой fair SCC.

Сначала докажем, что из (1) следует (2). Если существует fair путь σ из (q,K), то существует состояние $(q',K') \in \sigma$, такое, что все состояния из совокупности

$$\{(q'',K'') \in \sigma \mid (q'',K'') \underset{\sigma}{\geq} (q',K')\} \qquad (5.25)$$

входят в σ бесконечно много раз.

(5.25) является сильно связным подмножеством, и, следовательно, содержится в некоторой SCC C. Докажем, что C - fair SCC, т.е. для каждого $(q_1,K_1) \in C$ и каждой формулы $\mathbf{U}(\psi,\eta) \in \langle\varphi\rangle$

$$K_1(\mathbf{U}(\psi,\eta)) \leq \bigvee_{(q_2,K_2) \in C} K_2(\eta) \qquad (5.26)$$

Выберем произвольное состояние (q_1',K_1') из (5.25). Поскольку C сильно связна, то существует конечный путь $\rho \subseteq C$ из (q_1,K_1) в (q_1',K_1'). Обозначим символом σ' "хвост" σ, начинающийся с (q_1',K_1').

Учитывая (5.4) и (5.5), а также то, что σ' – fair путь, заключаем, что $\rho \cdot \sigma'$ – тоже fair путь, поэтому

$$K_1(\mathbf{U}(\psi,\eta)) \leq \bigvee_{(q_2,K_2) \in \rho \cdot \sigma'} K_2(\eta) \qquad (5.27)$$

Поскольку $\rho \cdot \sigma' \subseteq C$, то из (5.27) следует (5.26).

Теперь докажем, что из (2) следует (1). Пусть существует конечный путь σ_0 из (q,K) в $(q',K') \in C$, где C – fair SCC. Обозначим символом σ_1 путь вида $(q',K') \to (q',K')$, содержащий все состояния из C. Нетрудно видеть, что конкатенация $\sigma_0 \cdot \sigma_1^\omega$ является fair путём из (q,K). $\blacksquare$

5.3.3 Задача MC-LTL

Задача **MC-LTL** заключается в вычислении множества $Q_{\mathrm{E}\varphi} = \{q \in Q \mid q(\mathbf{E}\varphi) = 1\}$, где Q – множество состояний некоторой СП, и $\mathbf{E}\varphi$ – заданная LTL–формула.

Решение данной задачи даёт возможность вычислять также множества вида $Q_{\mathrm{A}\varphi}$, поскольку $\mathbf{A}\varphi = \overline{\mathbf{E}\overline{\varphi}}$.

Согласно определению, $q(\mathbf{E}\varphi) = 1$ означает, что

$$\exists \pi \ \text{ из } \ q: \ \pi(\varphi) = 1 \qquad (5.28)$$

Как было установлено выше, пути π соответствует fair путь σ_π в $S \times S_\varphi$ из некоторого состояния (q,K_0). Полагая в (5.22) $i = 0$ и $\psi = \varphi$, получаем:

$$\pi(\varphi) = \pi_0(\varphi) = K_0(\varphi)$$

Поэтому соотношение (5.28) эквивалентно следующему: существует состояние (q,K) системы $S \times S_\varphi$, такое, что

1. $K(\varphi) = 1$, и
2. из (q,K) выходит fair путь.

Как было доказано в конце предыдущего пункта, второе из этих условий равносильно существованию пути из (q,K) в некоторую fair SCC. Используя это соображение, можно построить алгоритм вычисления множества $Q_{\mathrm{E}\varphi}$, аналогичный алгоритму решения задачи MC-CTL из пункта 3.3.1. Сложность данного алгоритма имеет вид $O(|S| \cdot 2^{|\varphi|})$.

Можно доказать, что задачи MC-LTL и fair-MC-LTL являются PSPACE–полными.

Отметим, что на задачу проверки соотношения (5.28) можно смотреть и как на задачу fair-MC-CTL (и решать её символьными методами), поскольку истинность данного соотношения равносильна существованию такого состояния (q,K) системы $S \times S_\varphi$, что

1. $K(\varphi) = 1$, и
2. $(q,K)^F(\mathbf{EG1}) = 1$.

5.4 Автоматы Бюхи

5.4.1 Понятие автомата Бюхи

Автомат Бюхи (называемый ниже просто **автоматом**) – это пятёрка

$$\mathcal{B} = (A, Q, \delta, Q^0, F) \tag{5.29}$$

компоненты которой имеют следующий смысл:

1. A – множество, называемое **алфавитом**
2. Q – множество, элементы которого называются **состояниями**
3. δ – подмножество множества $Q \times A \times Q$, называемое **отношением перехода**
4. $Q^0 \subseteq Q$ – множество **начальных состояний**
5. $F = (F_1, \ldots, F_n)$ – список **fair множеств**, где для каждого $i = 1, \ldots, n \quad F_i \subseteq Q$.

Элементы множества δ называются **переходами**. Произвольный переход $(q, a, q') \in \delta$ обозначается знакосочетанием $q \xrightarrow{a} q'$.

Автомат можно представить в виде графа, вершинами которого являются состояния. Для каждого перехода $q \xrightarrow{a} q'$ граф содержит ребро с меткой a из q в q'.

Для каждого пути π в этом графе символ $L(\pi)$ обозначает последовательность меток рёбер, из которых состоит данный путь, т.е. если π имеет вид

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \ldots \tag{5.30}$$

то $L(\pi) = (a_0, a_1, a_2, \ldots)$.

Знакосочетание $\inf(\pi)$ обозначает множество всех состояний, которые встречаются на пути π бесконечное число раз. Путь π называется **fair**, если для каждого $i = 1, \ldots, n \quad \inf(\pi) \cap F_i \neq \emptyset$.

Язык автомата $\mathcal{B}$ – это множество $L(\mathcal{B})$ бесконечных цепочек символов алфавита A, соответствующих всевозможным fair путям из начальных состояний, т.е.

$$L(\mathcal{B}) = \{ L(\pi) \mid \pi - \text{fair путь из некоторого } q \in Q^0 \}$$

Автоматы $\mathcal{B}_1$ и $\mathcal{B}_2$ называются **эквивалентными**, если $L(\mathcal{B}_1) = L(\mathcal{B}_2)$.

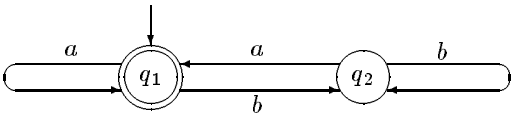
Для каждого автомата $\mathcal{B}$ вида (5.29) существует эквивалентный ему автомат $\mathcal{B}_1 = (A, Q_1, \delta_1, Q_1^0, F_1)$, такой, что список F_1 состоит только из одного множества. Компоненты $\mathcal{B}_1$ можно определить, например, так:

- $Q_1 = Q \times \{0, 1, \ldots, n\}, \quad Q_1^0 = Q^0 \times \{0\}$, $F_1 = (Q \times \{n\})$
- δ_1 состоит из переходов $(q, j) \xrightarrow{a} (q', j')$, таких, что $q \xrightarrow{a} q'$ и

$$j' = \begin{cases} k & \text{если } q' \in F_k \text{ и } j = k - 1 \\ 0 & \text{если } j = n \\ j & \text{в остальных случаях} \end{cases}$$

5.4.2 Пример автомата

Рассмотрим в качестве примера автомат



Данный автомат имеет следующие компоненты:

- $A = \{a, b\}$
- $Q = \{q_1, q_2\}$
- $Q^0 = \{q_1\}$
  (начальные состояния выделяются дополнительными стрелочками, ведущими в них)
- $F = (\{q_1\})$
  (состояния из fair множества обозначаются двойными кружочками)

Язык данного автомата имеет вид $(b^* \cdot a)^\omega$.

Ниже, если у автомата не указывается вид списка F его fair множеств, то предполагается, что этот список состоит только из одного множества, которое будет обозначаться тем же символом F.

5.4.3 Пересечение автоматов

Для каждой пары автоматов $\mathcal{B}_1, \mathcal{B}_2$, где

$$\mathcal{B}_i = (A, Q_i, \delta_i, Q_i^0, F_i) \qquad (i = 1, 2)$$

существует автомат $\mathcal{B}_1 \cap \mathcal{B}_2$, называемый **пересечением** $\mathcal{B}_1$ и B_2, и обладающий следующим свойством:

$$L(\mathcal{B}_1 \cap \mathcal{B}_2) = L(\mathcal{B}_1) \cap L(\mathcal{B}_2) \tag{5.31}$$

Компоненты автомата $\mathcal{B}_1 \cap \mathcal{B}_2$ можно определить, например, так:

- $Q = Q_1 \times Q_2 \times \{0, 1, 2\}, \quad Q^0 = Q_1^0 \times Q_2^0 \times \{0\}$, $F = Q_1 \times Q_2 \times \{2\}$,
- δ состоит из переходов $(q_1, q_2, j) \xrightarrow{a} (q_1', q_2', j')$, таких, что $q_i \xrightarrow{a} q_i' \quad (i = 1, 2)$, и

$$j' = \begin{cases} 1 & \text{если } j = 0 \text{ и } q_1' \in F_1 \\ 2 & \text{если } j = 1 \text{ и } q_2' \in F_2 \\ 0 & \text{если } j = 2 \\ j & \text{в остальных случаях} \end{cases}$$

Если $F_1 = Q_1$, то $\mathcal{B}_1 \cap \mathcal{B}_2$ можно определить проще:

- $Q = Q_1 \times Q_2, \quad Q^0 = Q_1^0 \times Q_2^0, \quad F = Q_1 \times F_2$.
- δ состоит из переходов $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$, таких, что $q_i \xrightarrow{a} q_i' \quad (i = 1, 2)$

5.4.4 Использование автоматов в задаче MC-LTL

Одна из возможных форм задачи **MC-LTL** заключается в том, чтобы для заданной СП $S = (\mathcal{P}, Q, \delta, L, Q^0)$ и заданной LTL–формулы вида $\mathbf{A}\varphi$ доказать, что

$$\forall q \in Q^0 \qquad q(\mathbf{A}\varphi) = 1 \tag{5.32}$$

т.е. для каждого пути π, выходящего из какого-либо начального состояния СП S, имеет место соотношение

$$\pi(\varphi) = 1. \tag{5.33}$$

Как было установлено в пункте 5.3.2, путь π определяет fair путь κ_π из некоторого состояния K СП S_φ, обладающего свойством $K(\varphi) = \pi(\varphi)$. Так как $\pi(\varphi) = 1$, то, следовательно, состояние K, из которого выходит путь κ_π, является начальным.

Сопоставим пути $\pi = (q_0, \ldots)$ последовательность $L(\pi) = (L(q_0), \ldots)$ функций вида $\mathcal{P}_\varphi \to \{0, 1\}$, в которой для каждого $i \geq 0$ и каждого $p \in \mathcal{P}_\varphi$ имеет место равенство $L(q_i)(p) = q_i(p)$. Пути $\kappa_\pi = (K_0, \ldots)$ можно сопоставить аналогичную последовательность $L(\kappa_\pi) = (L(K_0), \ldots)$ функций вида $\mathcal{P}_\varphi \to \{0, 1\}$, в которой для каждого $i \geq 0$ и каждого $p \in \mathcal{P}_\varphi$ имеет место равенство $L(K_i)(p) = K_i(p)$. Из (5.22) следует, что $L(\pi) = L(\kappa_\pi)$.

Для каждой СП S обозначим символом $L(S)$ множество всех последовательностей вида $L(\pi)$, где π – произвольный путь в S из некоторого начального состояния (если в S присутствуют условия fairness, то рассматриваются только fair пути).

Мы доказали, что из (5.32) следует включение

$$L(S) \subseteq L(S_\varphi) \tag{5.34}$$

Обратно, из (5.34) следует (5.32), т.к. (5.34) означает, что для каждого пути $\pi = (q_0, \ldots)$ из произвольного начального состояния $q_0 \in Q^0$ существует fair путь $\kappa = (K_0, \ldots)$ в S_φ, такой, что $K_0(\varphi) = 1$ и

$$\forall i \geq 0, \ \ \forall p \in \mathcal{P}_\varphi \qquad q_i(p) = K_i(p) \tag{5.35}$$

Из (5.35) следует, что последовательность $((q_0, K_0), \ldots)$ является fair путём в $S \times S_\varphi$, и, как было установлено в пункте 5.3.2, отсюда следует соотношение (5.22), из которого следует (5.33). ■

Соотношение (5.34) эквивалентно соотношению

$$L(S) \cap L(S_{\overline{\varphi}}) = \emptyset \tag{5.36}$$

потому что $L(S_\varphi) \cap L(S_{\overline{\varphi}}) = \emptyset$, и для каждой последовательности $(a_0, \ldots)$ функций вида $\mathcal{P}_\varphi \to \{0, 1\}$ имеет место одно из двух соотношений:

$$(a_0, \ldots) \in L(S_\varphi) \quad \text{или} \quad (a_0, \ldots) \in L(S_{\overline{\varphi}}).$$

Действительно, рассмотрим СП, множество состояний которой имеет вид $\{q_0, \ldots\}$, отношение перехода состоит из пар вида (q_i, q_{i+1}), и для каждого $p \in \mathcal{P}_\varphi$ и каждого $i \geq 0 \quad q_i(p) \stackrel{\text{def}}{=} a_i(p)$. Обозначим символом π путь $(q_0, \ldots)$ в этой СП. Ему соответствует fair путь κ'_π в S_φ и fair путь κ''_π в $S_{\overline{\varphi}}$. Согласно определению, $(a_0, \ldots) = L(\kappa'_\pi) = L(\kappa''_\pi)$, и если $\pi(\varphi) = 1$, то $(a_0, \ldots) \in L(S_\varphi)$, а если $\pi(\varphi) = 0$, то $(a_0, \ldots) \in L(S_{\overline{\varphi}})$.

Таким образом, (5.32) эквивалентно (5.36).

Один из возможных способов проверки соотношения (5.36) заключается в построении автоматов $\mathcal{B}_S$ и $\mathcal{B}_\varphi$, обладающих свойствами $L(\mathcal{B}_S) = L(S)$, $L(\mathcal{B}_\varphi) = L(S_{\overline{\varphi}})$, и проверке пустоты языка автомата $\mathcal{B}_S \cap \mathcal{B}_\varphi$.

Для каждой СП $S = (\mathcal{P}, Q, \delta, L, Q^0)$ автомат $\mathcal{B}_S$, обладающий свойством $L(S) = L(\mathcal{B}_S)$, можно построить, например, путём добавления к множеству состояний этой СП нового состояния *init* (которое будет начальным состоянием автомата $\mathcal{B}_S$), и рёбер из *init* во все состояния из Q^0. Если ребро автомата имеет вид $q \to q'$, то его метка равна $L(q')$ (т.е. алфавит всех рассматриваемых в данном пункте автоматов состоит из функций вида $\mathcal{P}_\varphi \to \{0, 1\}$). Список F fair множеств автомата $\mathcal{B}_S$ совпадает с аналогичным списком СП S. Если данный список состоит более чем из одного множества, то автомат $\mathcal{B}_S$ преобразуется в эквивалентный ему автомат с одним fair множеством. Если в S fair множества не указаны (т.е. все состояния СП S являются fair), то F состоит из множества всех состояний автомата $\mathcal{B}$.

Нетрудно доказать эквивалентность условий:

1. язык автомата (5.29) непуст
2. существует путь из некоторого его начального состояния $q \in Q^0$ в состояние $q' \in C$, где C – некоторая SCC множества Q, обладающая свойством $C \cap F \neq \emptyset$
3. существует путь из некоторого состояния $q \in Q^0$ в состояние $q' \in F$, через которое проходит цикл.

Таким образом, для проверки соотношения (5.36) можно проверять либо условие 2, либо условие 3.

Если проверяется условие 2, то для построения всех SCC со свойствами, указанными в условии 2, можно использовать алгоритм Тарьяна.

Если проверяется условие 3, то автомат $\mathcal{B}_S \cap \mathcal{B}_\varphi$ можно строить "на лету" ("on-the-fly"). Данный способ построения заключается в том, что сначала строится автомат $\mathcal{B}_\varphi$, который используется в процессе построения автомата $\mathcal{B}_S$. Пусть состояниями $\mathcal{B}_S \cap \mathcal{B}_\varphi$ являются пары (q_s, q_φ), где q_s – состояние $\mathcal{B}_S$, и q_φ – состояние $\mathcal{B}_\varphi$. Если уже построено некоторое состояние q_s автомата $\mathcal{B}_S$, то к тому фрагменту автомата $\mathcal{B}_S$, который уже построен, добавляются только такие состояния q'_s, для которых существует элемент a алфавита, такой, что $q_s \stackrel{a}{\to} q'_s$ и $q_\varphi \stackrel{a}{\to} q'_\varphi$ для некоторого состояния q'_φ автомата $\mathcal{B}_\varphi$.

Если построена часть автомата $\mathcal{B}_S \cap \mathcal{B}_\varphi$, содержащая путь, упомянутый в условии 3, то в построении всего автомата $\mathcal{B}_S \cap \mathcal{B}_\varphi$ уже нет необходимости.

5.4.5 Оптимизация построения $\mathcal{B}_\varphi$

Автомат $\mathcal{B}_\varphi$ можно строить не по СП $S_{\overline{\varphi}}$, а по более компактной СП S', задающей тот же самый язык.

СП S' строится следующим образом. Пронесём в $\overline{\varphi}$ все отрицания вниз, чтобы они располагались только над утверждениями, и обозначим получившуюся формулу символом α. Далее мы строим граф, каждая вершина q которого является подмножеством множества $\langle\alpha\rangle$, причём q разбито на два непересекающихся класса $\mathrm{New}(q)$ и $\mathrm{Old}(q)$.

На каждом шаге построения данный граф является аппроксимацией системы S'. В конце построения данный граф будет представлять собой искомую СП S'. Для каждой вершины q графа и каждого $p \in \mathcal{P}_\varphi$ значение $q(p)$ равно 1, если $p \in q$, и 0 – если $\overline{p} \in q$. Каждой формуле из $\langle\alpha\rangle$ вида $\mathbf{U}(\psi,\eta)$ соответствует fair множество $F_{\mathbf{U}(\psi,\eta)}$, состоящее из всех состояний q, для которых верна импликация $\mathbf{U}(\psi,\eta) \in q \;\Rightarrow\; \eta \in q$.

Каждый шаг построения осуществляется в соответствии со следующим замыслом: для каждой вершины q, и каждой формулы $\psi \in q$, ψ должна быть истинной на всех fair путях, выходящих из q.

Сначала строим вершину $q_0 \stackrel{\text{def}}{=} \{\alpha\} = \mathrm{New}(q_0)$.

Для очередного шага построения выбирается произвольная вершина q, у которой $\mathrm{New}(q) \neq \emptyset$ (если таких вершин нет, то построение закончено).

1. Если в $\mathrm{New}(q)$ есть хоть одна формула β, не начинающаяся с $\mathbf{X}$, то она переносится из $\mathrm{New}(q)$ в $\mathrm{Old}(q)$, после чего
   (a) если $\beta = \psi \wedge \eta$, то ψ и η добавляются к $\mathrm{New}(q)$
   (b) если $\beta = \psi \vee \eta$, то
   - создаётся дубликат q' вершины q с теми же классами New и Old, и для каждого ребра, ведущего в q, создаётся новое ребро с тем же началом, но с концом в q',
   - ψ добавляется к $\mathrm{New}(q)$,
   - η добавляется к $\mathrm{New}(q')$.

   (c) если $\beta = \mathbf{U}(\psi,\eta)$, то выполняем те же операции, что и в предыдущем пункте, применительно к формуле $\eta \vee \Big(\psi \wedge \mathbf{X}\beta\Big)$, т.е.
   - создаются дубликат q' вершины q и новые рёбра, ведущие в q',
   - η добавляется к $\mathrm{New}(q)$
   - ψ и $\mathbf{X}\beta$ добавляются к $\mathrm{New}(q')$

   (d) если $\beta = \psi\mathbf{R}\eta$, то обрабатываем её так же, как $\eta \wedge \Big(\psi \vee \mathbf{X}\beta\Big)$, т.е.
   - создаются дубликат q' вершины q и новые рёбра, ведущие в q',
   - η и ψ добавляются к $\mathrm{New}(q)$
   - η и $\mathbf{X}\beta$ добавляются к $\mathrm{New}(q')$

   Затем проверяется непротиворечивость и неизбыточность модифицированной вершины q:
   - если в q входит формула $\mathbf{0}$, или пара формул вида $p, \overline{p}$, то q и все ведущие в неё рёбра удаляются
   - если в q входит формула $\mathbf{1}$, то эта формула удаляется
   - если в q входит пара одинаковых формул, то удаляется та из них, которая входит в $\mathrm{New}(q)$

   В случаях (b), (c), (d) те же действия выполняются для q'.

2. Если в $\mathrm{New}(q)$ все формулы начинаются с $\mathbf{X}$, то
   (a) в том случае, когда $\exists\, q' \neq q$:
   $$\mathrm{New}(q) \subseteq \mathrm{New}(q') \quad \text{и} \quad \mathrm{Old}(q) = \mathrm{Old}(q')$$
   вершина q удаляется, а каждое ведущее в неё ребро перенаправляется в q'
   (b) иначе создаются новая вершина $q' = \{\psi \mid \mathbf{X}\psi \in \mathrm{New}(q)\} = \mathrm{New}(q')$, и ребро из q в q', а все формулы из $\mathrm{New}(q)$ удаляются.

Начальными состояниями СП S' являются такие вершины, которые содержат формулу α.

5.4.6 Проверка включения языков

Поскольку задача MC-LTL в форме проверки условия (5.32) может быть сведена к проверке условия (5.34), то исследование проблемы включения языков также представляет большой интерес. Эту проблему можно рассмотреть в следующей постановке: по заданным автоматам $\mathcal{B}_1$ и $\mathcal{B}_2$ над одним и тем же алфавитом проверить условие $L(\mathcal{B}_1) \subseteq L(\mathcal{B}_2)$. Известно, что данная проблема PSPACE-полна.

Пусть автоматы $\mathcal{B}_i \quad (i = 1,2)$ имеют вид

$$(\ A,\ Q_i,\ \delta_i,\ Q_i^0,\ F_i\ )$$

Обозначим символом S СП

$$(\ \{p_1,p_2\},\ Q_1 \times Q_2,\ \delta,\ L,\ Q_1^0 \times Q_2^0\ )$$

где

- $(q_1,q_2) \to (q_1',q_2')$, если существует a, такой, что $q_i \stackrel{a}{\to} q_i' \quad (i = 1,2)$
- $(q_1,q_2)(p_i) = 1 \quad \Leftrightarrow \quad q_i \in F_i \quad (i = 1,2)$.

Можно доказать, что условие $L(\mathcal{B}_1) \subseteq L(\mathcal{B}_2)$ эквивалентно каждому из следующих условий:

- в каждом начальном состоянии S истинна LTL–формула $\mathbf{A}(\mathbf{GF}p_1 \to \mathbf{GF}p_2)$
- в каждом начальном состоянии S истинна CTL–формула $\mathbf{AGAF}p_2$ с ограничениями fairness, задающимися CTL–формулой $\mathbf{AGAF}p_1$.

Глава 6

Системы с передачей сообщений

6.1 Программные системы с передачей сообщений

Программная система с передачей сообщений (называемая в этой главе более коротко **системой**) представляет собой конечную совокупность Σ программ, взаимодействующих друг с другом посредством передачи сообщений. Каждое сообщение представляет собой список значений. Множество всех возможных значений будет обозначаться символом $\mathcal{D}$, а множество всех списков значений – символом $\mathcal{D}^*$.

6.1.1 Понятие программы

Программа представляется некоторой блок-схемой, каждая вершина n которой помечена некоторым оператором $Op(n)$. Вид этих операторов определяется ниже.

С каждой программой P связано некоторое множество $Var(P)$ её **переменных**, причём для разных программ их множества переменных не пересекаются. Каждой переменной x сопоставлено множество её возможных значений $\mathcal{D}_x \subseteq \mathcal{D}$, которые может принимать переменная x. Если X – список переменных вида

$$(x_1, \ldots, x_n)$$

то символ $\mathcal{D}_X$ обозначает декартово произведение

$$D_{x_1} \times \ldots \times D_{x_n}.$$

Кроме того, считается заданным некоторое множество функциональных символов

$$+, \quad -, \quad \textbf{head}, \quad \textbf{tail}, \quad \ldots$$

при помощи которых можно стандартным образом строить выражения с переменными. Если каждой переменной x, входящей в выражение e, сопоставлено некоторое значение $\xi(x)$, то всё выражение e принимает значение $\xi(e)$, определяемое стандартным образом. Если E – список выражений вида

$$(e_1, \ldots, e_n) \tag{6.1}$$

и каждой переменной x, входящей в какое-либо из выражений списка E, сопоставлено некоторое значение $\xi(x)$, то символ $\xi(E)$ обозначает список соответствующих значений

$$(\xi(e_1), \ldots, \xi(e_n)).$$

Ниже для каждой программы P

- символ $Tm(P)$ будет обозначать совокупность всех выражений с переменными из $Var(P)$
- символ $Fm(P)$ будет обозначать совокупность тех выражений из $Tm(P)$, которые принимают только булевские значения (0 или 1). Выражения из $Fm(P)$ называются **формулами**.

Операторы, входящие в блок-схему, могут иметь следующий вид.

начало:

$$Op(n) = \left(\begin{array}{c} \textbf{start} \\ Init(P) \end{array} \right) \tag{6.2}$$

где $Init(P)$ – формула, называемая **начальным условием** программы P.

присваивание:

$$Op(n) = \left(\; x := e \; \right) \tag{6.3}$$

где

- x – некоторая переменная, и
- e – некоторое выражение.

проверка условия:

$$Op(n) = \left(\; b \; \right) \tag{6.4}$$

где b – некоторая формула.

посылка сообщения:

$$Op(n) = \left(\; P' \leftarrow E \; \right) \tag{6.5}$$

где

- P' – имя программы, которой посылается сообщение, и

- E – список выражений, значения которых посылаются программе P'.

получение сообщения:

$$Op(n) = \quad \left(\; X \leftarrow P' \; \right) \tag{6.6}$$

где

- P' – имя программы, от которой приходит сообщение, и
- X – список различных переменных, в которые записываются полученные значения.

выбор:

$$Op(n) = \quad \left(\quad \right) \tag{6.7}$$

(пустая метка).

остановка:

$$Op(n) = \quad \left(\; \textbf{halt} \; \right) \tag{6.8}$$

Из вершин с меткой вида (6.2), (6.3), (6.5), (6.6) выходит только одно ребро. Из вершин с меткой вида (6.4) выходят два ребра: одно имеет метку "+", другое – метку "−". Из вершин с меткой (6.7) может выходить произвольное количество рёбер. Из вершин с меткой (6.8) не выходит ни одного ребра.

6.1.2 Функционирование программы

Функционирование программы P происходит обычным образом, и заключается в обходе вершин блок-схемы, с выполнением операторов, сопоставленных проходимым вершинам. После выполнения оператора, соответствующего текущей вершине, происходит переход по выходящему из неё ребру к следующей вершине. На каждом шаге t функционирования ($t = 0, 1, \ldots$) каждая переменная x программы содержит некоторое значение $\xi_t(x)$. Значения переменных в начальный момент должны удовлетворять начальному условию.

Операторы выполняются следующим образом.

- Оператор (6.3) заносит значения выражения e в переменную x.
- Оператор (6.4) вычисляет значение формулы b, и
  - если оно равно 1, то происходит переход к следующей вершине по ребру с меткой "+",
  - иначе - по ребру с меткой "−".
- Оператор (6.5) выполняется путём
  - ожидания момента, когда программа P' будет готова принять сообщение от P,
  - как только наступает такой момент, программе P' посылается список значений выражений, входящих в E.
- Оператор (6.6) выполняется путём
  - ожидания момента, когда от программы P' поступит некоторое сообщение,
  - как только наступает такой момент, значения из поступившего сообщения записываются в переменные из списка X.
- Если текущая вершина помечена оператором (6.7), то
  - из рёбер, которые из неё выходят, выбирается ребро, ведущее в вершину, помеченную таким оператором, который возможно выполнить, и
  - происходит переход в эту вершину.
- Оператор (6.8) завершает выполнение всей программы.

6.2 Пример системы

В этом пункте мы рассмотрим пример системы, представляющей собой реализацию протокола передачи данных через ненадёжную среду.

Эта система будет обозначаться символом UBP (Unbounded Buffer Protocol).

6.2.1 Описание системы UBP

Задача UBP заключается в том, чтобы получать сообщения от программы In, являющейся внешней по отношению к системе UBP, и доставлять их в том же порядке и без искажений программе Out (тоже являющейся внешней по отношению к системе UBP). Предполагается, что в процессе доставки передаваемые сообщения проходят через среду (мы будем считать её частью системы UBP), в которой может произойти частичное искажение передаваемых сообщений. Система UBP должна уметь распознавать искажения и переносылать те сообщения, которые были искажены.

UBP состоит из четырёх программ:

$$Sender, \quad Receiver, \quad Buffer_1 \quad \text{и} \quad Buffer_2$$

взаимодействующих друг с другом так, как показано на рис. 1. Программы $Buffer_1$ и $Buffer_2$ являются моделью среды, через которую передаются сообщения.

Для программ, входящих в UBP, мы будем использовать наряду с их полными именами также сокращённые имена: S, R, B_1 и B_2 соответственно.

6.2.2 Программа $Sender$

Программа $Sender$ изображена на рис. 2.

$Sender$ имеет массив s, в который он записывает сообщения, поступающие к нему от программы In. Получив очередное сообщение g, программа $Sender$

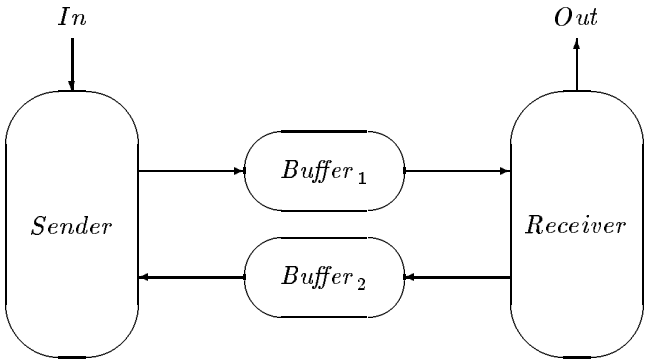


Рис. 1: *UBP*.

- сопоставляет ему порядковый номер in $(= 1, 2, \ldots)$,
- записывает в $s[in]$ полученное сообщение,
- посылает программе $Buffer_1$ пакет, представляющий собой пару (g, in).

Также *Sender* может получать пакеты от программы $Buffer_2$, которые являются подтверждениями, посылаемыми ему программой *Receiver*:

- если полученный пакет имеет вид (g, k), и g совпадает с $s[k]$, то это значит, что *Receiver* получил сообщение с номером k без искажения, и в этом случае *Sender* посылает программе $Buffer_1$ пакет

  $$(\$, k)$$

  где \$ – специальный символ, который не может быть искажён в процессе передачи,
- если же полученный пакет имеет вид (g, k), и g не совпадает с $s[k]$, то это означает, что сообщение с номером k возможно было искажено в процессе передачи, и в этом случае *Sender* посылает программе $Buffer_1$ пакет $(s[k], k)$ ещё раз.

6.2.3 Программа *Receiver*

Программа *Receiver* изображена на рис. 3.

Receiver имеет массив r, в который он записывает сообщения, поступающие к нему от программы $Buffer_1$:

- если *Receiver* получил пакет (m, l), где m не совпадает с "\$" (это означает, что ещё пока нет уверенности в том, что сообщение с номером l дошло без искажений), то *Receiver*
  - пересылает этот пакет (в качестве подтверждения) программе $Buffer_2$
  - записывает значение m в $r[l]$,
- если же полученный пакет имеет вид $(\$, l)$ то это означает, что сообщение с номером l было получено программой *Receiver* без искажения, и его можно переслать программе Out.

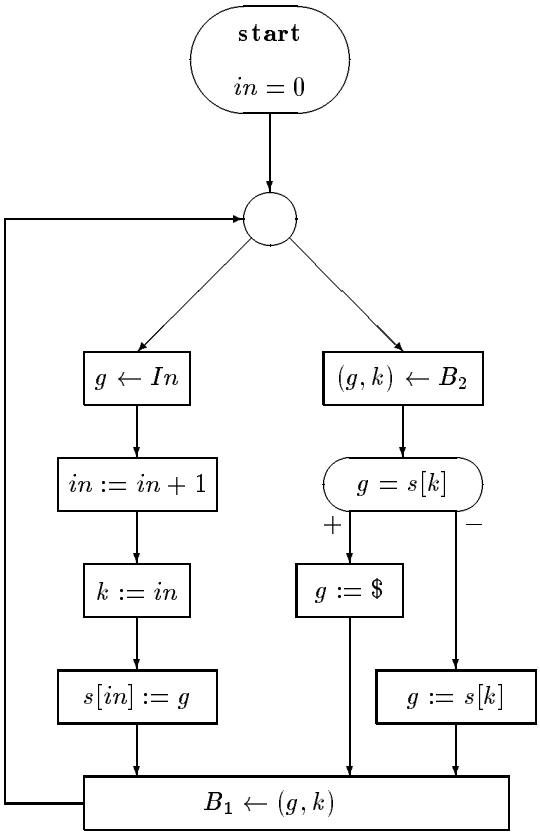


Рис. 2: *Sender*.

Receiver имеет булевозначный массив h, где для каждого $i \geq 1 \quad h[i] = 1$ если i–е сообщение было передано от программы *Sender* программе *Receiver* без искажения.

6.2.4 Программы $Buffer_1$ и $Buffer_2$

Программа $Buffer_1$ изображена на рис. 4. Данная программа изображает среду, через которую передаются пакеты.

$Buffer_1$ представляет собой неограниченный буфер, который может содержать произвольную совокупность пакетов вида (d, k), где

- d – **тело** пакета (некоторое значение),
- k – **номер** пакета (натуральное число).

Пакеты, содержащиеся в буфере, могут переупорядочиваться. Кроме того, тела этих пакетов могут искажаться, но их номера искажаться не могут. Мы предполагаем, что имеется **искажающая функция**

$$f : \mathcal{D} \to \mathcal{D}$$

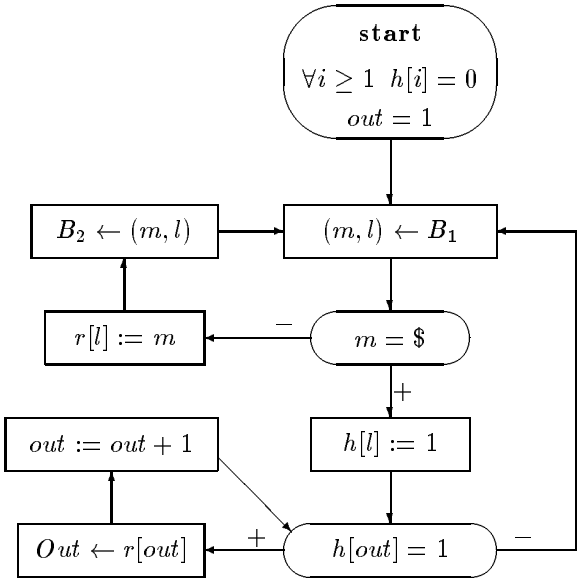


Рис. 3: *Receiver*.

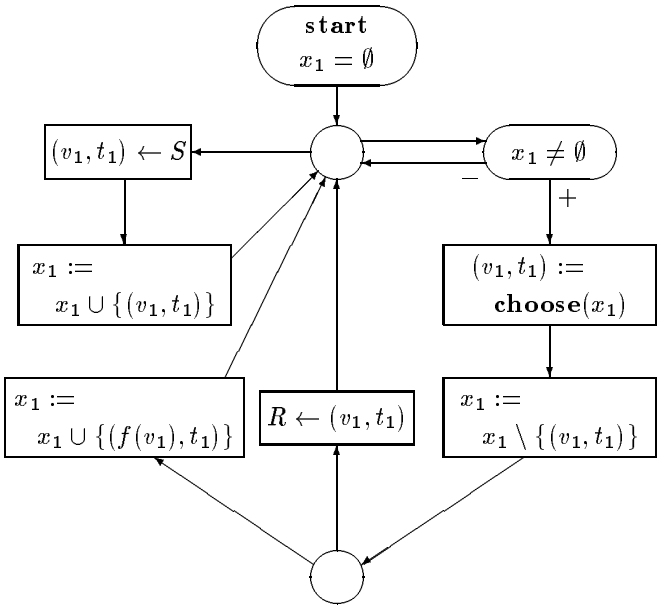


Рис. 4: $Buffer_1$.

такая, что $f(\$) = \$$, и

$$\forall d \in \mathcal{D} \qquad (\; f(d) \neq d \quad \Rightarrow \quad \forall k \geq 1 \quad f^k(d) \neq d \;). \tag{6.9}$$

$Buffer_1$ имеет переменную x_1, в которой содержатся пакеты, полученные от программы *Sender*, и пока ещё не посланные программе *Receiver* (т.е. каждое возможное значение переменной x_1 представляет собой *множество* пакетов.

Функция **choose** в программе $Buffer_1$ является недетерминированной, и если значение x_1 в текущий момент времени является непустым множеством, то значение **choose**(x_1) есть произвольный пакет из x_1.

Программа $Buffer_2$ полностью идентична программе $Buffer_1$, за исключением того, что она принимает пакеты от *Receiver*а, а посылает их *Sender*у. Её переменные мы будем обозначать символами x_2, v_2, t_2.

6.2.5 Спецификация

Одно из свойств системы UBP, которое можно выразить в виде формальной спецификации, заключается в том, что она работает как очередь типа FIFO ("First Input - First Output"), что означает следующее: последовательность сообщений, передаваемых от UBP программе Out, совпадает с последовательностью сообщений, которую UBP получает от программы In.

Более формально данное свойство можно выразить следующим образом: внешнее поведение UBP эквивалентно поведению программы *Spec*, изображённой на рис. 5.

Spec представляет собой неограниченный FIFO буфер, который

- не искажает полученные сообщения, и
- выдаёт их программе Out в том же порядке, в котором они в него поступают от программы In.

Spec имеет переменную q, содержащую список сообщений, которые были получены, но пока ещё не выданы.

После получения от программы In нового сообщения, *Spec* добавляет его в конец списка q.

В программе *Spec* используются следующие обозначения:

- символ "$\cdot$" обозначает операцию конкатенации,
- символ "ε" обозначает пустой список,
- знакосочетание **first**(q) обозначает первый элемент списка q, и
- знакосочетание **tail**(q) обозначает список, получаемый из q путём удаления его первого элемента.

6.3 Формальные модели систем и программ

6.3.1 Модель системы

Модель системы представляет собой список Σ вида

$$\Sigma = (P_1, \ldots, P_n) \tag{6.10}$$

состоящий из графовых моделей программ, входящих в данную систему.

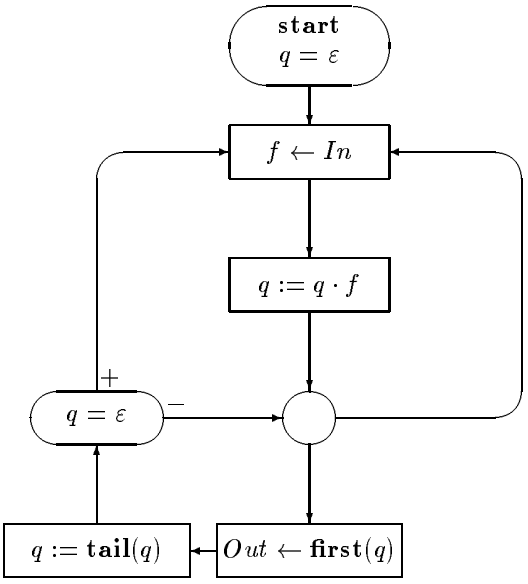


Рис. 5: *Spec*.

6.3.2 Графовые модели программ

Пусть P – некоторая программа.

Графовая модель программы P представляет собой ориентированный граф с выделенной вершиной $Start(P)$.

Множество вершин данного графа обозначается символом $N(P)$, а множество его рёбер – $F(P)$.

Каждое ребро $f \in F(P)$ имеет метку $\langle f \rangle$, которая называется **действием**. Действие может содержать следующие компоненты (каждая из которых может отсутствовать):

1. **Условие:**
$$\varphi\ ? \tag{6.11}$$
где φ – некоторая формула из $Fm(P)$.

2. **Посылка или приём сообщения**:
$$P' \ \leftarrow \ E \qquad \text{(посылка сообщения)} \tag{6.12}$$
или
$$X \ \leftarrow \ P' \qquad \text{(приём сообщения)} \tag{6.13}$$
где
   - P' – некоторая программа,
   - E – список выражений из $Tm(P)$,
   - X – список различных переменных из $Var(P)$.

3. **Подстановка**, т.е. отображение вида
$$\theta : Var(P) \to Tm(P) \tag{6.14}$$

Мы будем предполагать, что для каждой программы P

- множество её переменных $Var(P)$ содержит специальную переменную at_P,
- в каждой графовой модели программы P переменная at_P принимает значения в множестве $N(P)$ вершин данной модели, и
- для каждого ребра $f \in F(P)$ компонента (6.14) метки $\langle f \rangle$ удовлетворяет следующему условию:
$$\theta(at_P) = n$$
где вершина n является концом ребра f.

Подстановку (6.14) мы будем обозначать также записью вида
$$(e_1/x_1, \ldots, e_n/x_n) \tag{6.15}$$
где

- список
$$x_1, \ldots, x_n \tag{6.16}$$
состоит из всех тех переменных из множества
$$Var(P) \setminus \{at_P\} \tag{6.17}$$
для которых $x_i \neq \theta(x_i)$, и
- для каждого $i = 1, \ldots, n$
$$e_i = \theta(x_i)$$

Ниже для каждой подстановки θ вида (6.14) и каждого выражения $e \in Tm(P)$ мы будем обозначать символом $\theta(e)$ выражение, получаемое заменой каждой переменной x в e на выражение $\theta(x)$. Если E – список выражений вида (6.1), то символ $\theta(E)$ обозначает список
$$(\theta(e_1), \ldots, \theta(e_n)).$$

Если у действия отсутствует компонента, связанная с приёмом или передачей сообщения, то такое действие называется **внутренним**.

Если компоненты "условие" или "подстановка" не указаны, то они по умолчанию имеют следующий вид:

- компонента "условие" есть тавтология (например, формула $1 = 1$),
- компонента "подстановка" действует тождественно на все переменные из (6.17) (т.е. список (6.16) является пустым).

6.3.3 Функционирование графовой модели

Понятие **функционирования** графовой модели аналогично понятию функционирования блок-схемы, и заключается в обходе вершин графовой модели (начиная с вершины $Start(P)$), с выполнением действий, являющихся метками проходимых рёбер.

На каждом шаге t функционирования ($t = 0, 1, \ldots$) каждой переменной x из $Var(P)$ сопоставлено некоторое значение $\xi_t(x)$. Значения переменных в начальный момент времени должны удовлетворять начальному условию.

На каждом шаге t функционирования графовой модели значение переменной at_P равно той вершине, в которой на данном шаге мы находимся. В частности,

$$\xi_0(at_P) = Start(P).$$

Соответствие ξ_t между переменными и их значениями на шаге t можно рассматривать как отображение

$$\xi_t : Var(P) \to \mathcal{D}$$

Данное отображение называется **состоянием** программы P в момент времени t. Совокупность всех возможных состояний программы P обозначается символом $S(P)$.

Ниже для каждого состояния ξ и каждого выражения e из $Tm(P)$ знакосочетание $\xi(e)$ обозначает значение, получаемое

- заменой каждой переменной x в e на значение $\xi(x)$, и
- применением в получившемся выражении операций, соответствующих функциональным символам, входящим в e.

Каждый шаг t функционирования графовой модели P начинается с выбора такого ребра f с началом в $\xi_t(at_P)$, что действие $\langle f \rangle$ возможно выполнить. После этого выполняется действие $\langle f \rangle$, и происходит переход в вершину, являющуюся концом ребра f. В этой вершине мы будем находиться в момент $t + 1$.

Действие $\langle f \rangle$ выполняется путём последовательного выполнения всех его компонентов: сначала выполнятся первая компонента, потом - другая (если она есть), затем - третья (если она есть).

Компонента (6.11) выполняется путём вычисления значения формулы φ, и

- если оно равно 0, то выполнение всего действия является невозможным,
- иначе - выполнение действия продолжается.

Выполнение компонентов (6.12) и (6.13) происходит аналогично выполнению соответствующих операторов в блок-схемах.

Выполнение компоненты (6.15) происходит путём

- вычисления значения выражений $e_1, \ldots, e_n$, и
- занесения этих значений в переменные $x_1, \ldots, x_n$ соответственно, т.е.

$$\xi_{t+1}(x_1) \stackrel{\text{def}}{=} \xi_t(e_1), \quad \ldots, \quad \xi_{t+1}(x_n) \stackrel{\text{def}}{=} \xi_t(e_n)$$

6.3.4 Преобразование программ в их графовые модели

Алгоритм построения по блок-схеме программы P её графовой модели имеет следующий вид.

1. На каждом ребре блок-схемы рисуется точка.
2. Для каждой вершины n блок-схемы, и каждой пары F_1, F_2 рёбер блок-схемы, таких что F_1 входит в n, а F_2 - выходит из n
   (a) рисуется ребро f, соединяющее точку на F_1 с точкой на F_2,
   (b) на этом ребре f рисуется метка $\langle f \rangle$, определяемая следующим образом:
      i. если $Op(n)$ имеет вид (6.3), то
      $$\langle f \rangle \stackrel{\text{def}}{=} \left(\; e/x \; \right)$$
      ii. если $Op(n)$ имеет вид (6.4), и ребро, выходящее из n, помечено символом "+", то
      $$\langle f \rangle \stackrel{\text{def}}{=} \left(\; b \; ? \; \right)$$
      iii. если $Op(n)$ имеет вид (6.4), и ребро, выходящее из n, помечено символом "−", то
      $$\langle f \rangle \stackrel{\text{def}}{=} \left(\; \neg b \; ? \; \right)$$
      iv. если $Op(n)$ имеет вид (6.6) или (6.5), то $\langle f \rangle$ совпадает с $Op(n)$,
      v. если $Op(n)$ имеет вид (6.7), то действие $\langle f \rangle$ пусто (не содержит ни одной компоненты).
3. Если имеется хотя бы одно ребро f с пустым действием $\langle f \rangle$, то
   (a) обозначим начало и конец ребра f символами ν_1 и ν_2 соответственно,
   (b) каждое из рёбер, выходящее из ν_2, преобразуем в ребро с той же меткой, выходящее из ν_1,
   (c) удалим ребро f и точку ν_2.

   Данная операция повторяется до тех пор пока не останется ни одного ребра f с пустым действием.
4. Оставшиеся точки являются вершинами графовой модели.
5. $Start(P)$ есть точка, нарисованная на ребре, выходящем из начальной вершины блок-схемы.

Например, графовые модели программ из системы UBP, а также программы $Spec$, имеют вид, изображённый на рис. 6, 7, 8, и 9.

Начиная со следующего пункта, мы будем называть графовые модели программ просто **программами**.

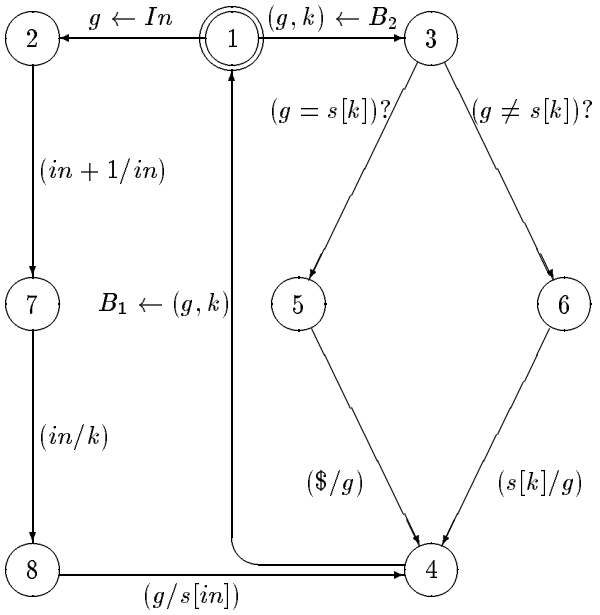


Рис. 6: Графовая модель программы *Sender*.

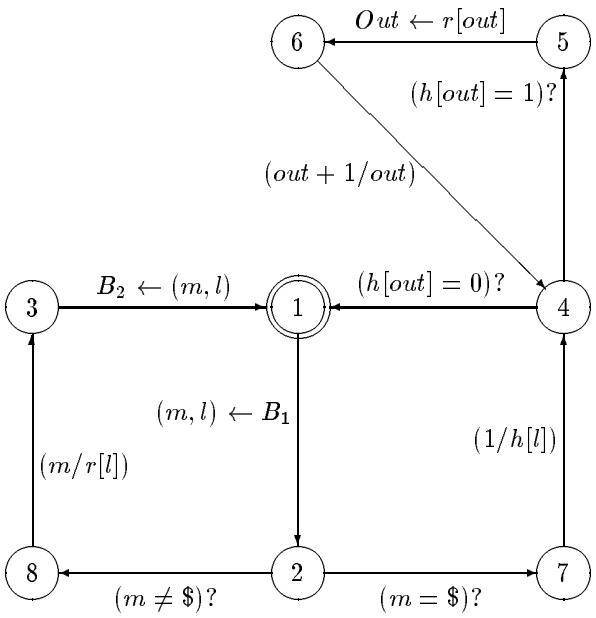


Рис. 7: Графовая модель программы *Receiver*.

6.4 Редукция программ

Сложность процедуры верификации систем зависит от сложности их моделей: чем меньше размер модели, тем проще её верифицировать.

В данном пункте мы излагаем один метод преобразования программ, который позволяет уменьшать их размер.

6.4.1 Композиция действий

Пусть P – некоторая программа, и α_1 и α_2 – пара её действий.

В том случае, когда α_1 и α_2 удовлетворяют некоторым условиям, возможно определить действие, называемое **композицией** α_1 и α_2, которое заключается в последовательном выполнении α_1 и α_2.

Одно из этих условий - хотя бы одно из действий α_1, α_2 должно быть внутренним.

Обозначим

- символами φ_1 и φ_2 – компоненты "условие" действий α_1 и α_2, и
- символами θ_1 и θ_2 – компоненты "подстановка" действий α_1 и α_2.

Композицией α_1 и α_2 называется действие, обозначаемое знакосочетанием

$$\alpha_1\alpha_2 \tag{6.18}$$

и определяемое следующим образом:

1. компонента "условие" действия (6.18) имеет вид

$$\varphi_1 \ \wedge \ \theta_1(\varphi_2)$$

2. компонента "подстановка" имеет вид

$$\theta_1\theta_2 \tag{6.19}$$

где знакосочетание (6.19) обозначает подстановку, называемую **композицией** подстановок θ_1 и θ_2, и определяемую следующим образом:

$$\forall x \in Var(P) \qquad (\theta_1\theta_2)(x) \stackrel{\text{def}}{=} \theta_1(\theta_2(x))$$

3. если
   - α_1 содержит компоненту $X \leftarrow P'$, и
   - формула $\theta_1(\varphi_2)$ не зависит от переменных из X,

   то (6.18) содержит такую же компоненту,

4. если
   - α_2 содержит компоненту $X \leftarrow P'$, и
   - для всех $y \in Var(P) \setminus X \quad \theta(y)$ не зависит от переменных из X,

   то (6.18) содержит такую же компоненту,

5. если α_1 содержит компоненту $P' \leftarrow E$, то (6.18) содержит такую же компоненту,

6. если α_2 содержит компоненту $P' \leftarrow E$, то (6.18) содержит компоненту

$$P' \leftarrow \theta_1(E)$$

В тех случаях, когда условия в пунктах 3 и 4 не выполнены, действие (6.18) не определено.

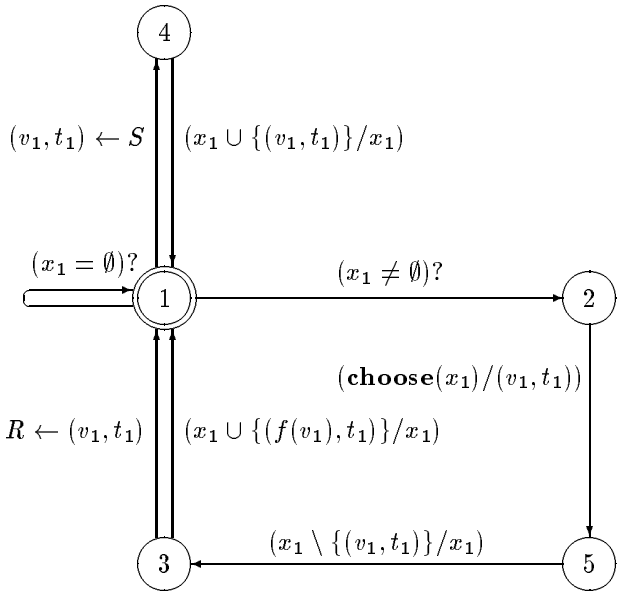


Рис. 8: Графовая модель программы $Buffer_1$.

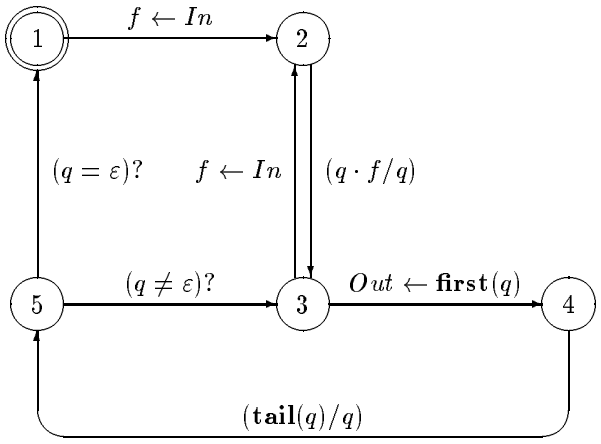


Рис. 9: Графовая модель программы *Spec*.

6.4.2 Определение редукции программ

Операция редукции программы заключается в удалении из неё одной вершины и преобразовании некоторых рёбер.

Данную операцию можно применить в том случае, когда в программе P имеется ребро f_0, обладающее перечисленными ниже свойствами.

Для формулировки этих свойств мы введём следующие обозначения.

- Начало и конец ребра f_0 обозначим символами n_0 и n соответственно.
- Список всех рёбер, выходящих из n, обозначим знакосочетанием
$$f_1, \ldots, f_k \tag{6.20}$$
- Концы рёбер из списка (6.20) обозначим символами
$$n_1, \ldots, n_k \tag{6.21}$$
- Условия действий $\langle f_0 \rangle, \langle f_1 \rangle, \ldots, \langle f_k \rangle$ обозначим символами
$$\varphi_0, \varphi_1, \ldots, \varphi_k$$

Необходимые условия для применения операции редукции заключаются в следующем:

1. $n_0 \neq n$,
2. $n \neq Start(P)$,
3. существуют композиции действий
$$\langle f_0 \rangle \langle f_1 \rangle, \quad \ldots, \quad \langle f_0 \rangle \langle f_k \rangle \tag{6.22}$$
4. если действие $\langle f_0 \rangle$ – внутреннее, то для каждого ребра f, выходящего из n_0, за исключением f_0, следующая формула является тавтологией:
$$\neg(\varphi \wedge \varphi_0)$$
где φ – условие действия $\langle f \rangle$,
5. если действие $\langle f_0 \rangle$ – не внутреннее, то следующие формулы являются тавтологиями:
$$\bigvee_{i=1}^{k} \varphi_i \quad \text{и} \quad \bigwedge_{1 \leq i < j \leq k} \neg(\varphi_i \wedge \varphi_j)$$

Если данные условия выполнены, то к программе P можно применить **операцию редукции**, которая заключается в выполнении следующих действий:

- удаление вершины n и рёбер $f_0, f_1, \ldots, f_k$,
- добавление новых рёбер из n_0 в вершины из списка (6.21) с метками из списка (6.22) соответственно,
- удаление вершин, недостижимых из начальной вершины,
- удаление цикловых рёбер с однокомпонентными действиями вида φ ?,
- удаление бесполезных присваиваний (после которых новые значения переменных не используются).

Операция редукции уменьшает число вершин. Может оказаться так, что к редуцированной программе опять можно применить операцию редукции. Применение данной операции несколько раз может существенно уменьшить число вершин анализируемой программы. Например, применение операции редукции к программам из системы UBP уменьшает их размер в среднем в 3 раза.

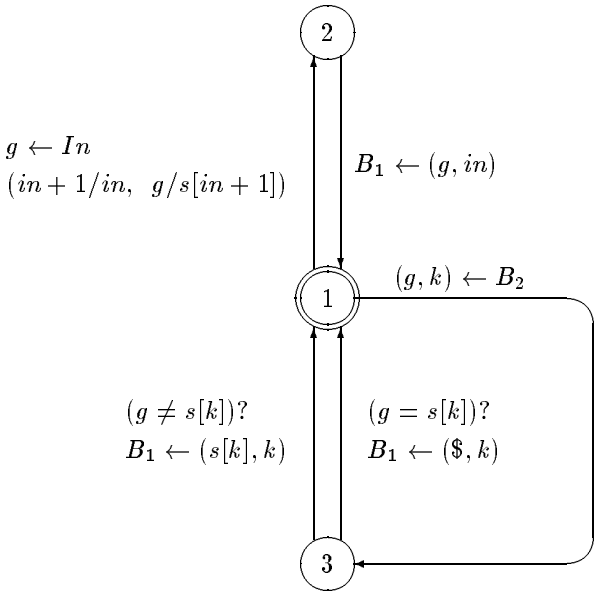


Рис. 10: $Sender'$.

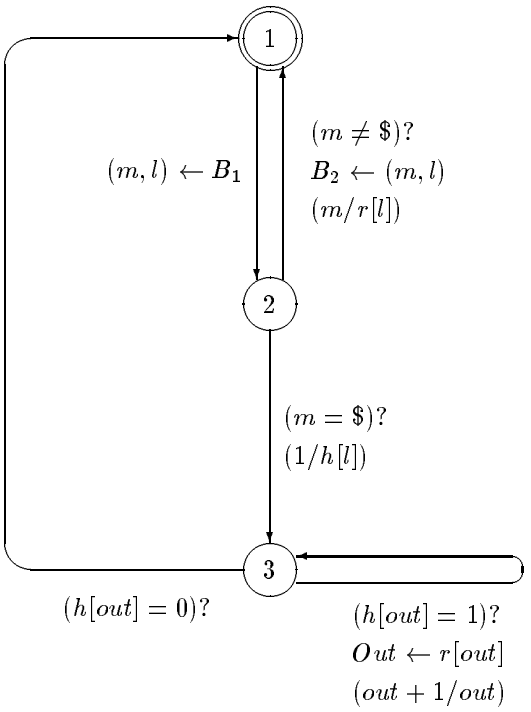


Рис. 11: $Receiver'$.

6.4.3 Пример редукции программ

В результате применения операции редукции к программам из системы UBP получаются программы

$$Sender',\ \ Receiver',\ \ Buffer'_i\ \ (i \in \{1,2\}),\ \ Spec'$$

изображённые на рис. 10 – 13.

6.5 Отношения перехода

6.5.1 События

Событием называется знакосочетание одного из следующих видов:

- $P \leftarrow D$ (передача списка значений D программе P),
- $D \leftarrow P$ (приём списка значений D от программы P),
- τ (внутреннее событие).

где P – некоторая программа, D – некоторый список значений из $\mathcal{D}^*$, и τ – некоторый специальных символ.

Множество всех событий обозначается символом $\mathcal{M}$.

6.5.2 Отношение перехода на состояниях программы

Пусть P – некоторая программа.

В данном пункте мы определяем **отношение перехода** "$\rightarrow$" вида

$$\rightarrow \ \subseteq \ S(P) \times \mathcal{M} \times S(P)$$

Если тройка (ξ, μ, ξ') принадлежит "$\rightarrow$", то данный факт будет обозначаться знакосочетанием

$$\xi \xrightarrow{\ \mu\ } \xi'. \qquad (6.23)$$

Соотношение (6.23) имеет место в том и только в том случае, когда существует ребро f из $F(P)$, обладающее перечисляемыми ниже свойствами. В формулировке данных свойств символы φ и θ обозначают компоненты "условие" и "подстановка" действия $\langle f \rangle$.

1. Вершины $\xi(at_P)$ и $\xi'(at_P)$ являются соответственно началом и концом ребра f.
2. $\xi(\varphi) = 1$.
3. Если $\langle f \rangle$ внутреннее действие, то
$$\mu = \tau \quad \text{и} \quad \xi' = \xi\theta$$
где символ $\xi\theta$ обозначает состояние, сопоставляющее переменной $x \in Var(P)$ значение $\xi(\theta(x))$.
4. Если $\langle f \rangle$ содержит компоненту $P' \leftarrow E$, то
$$\mu = P' \leftarrow \xi(E) \quad \text{и} \quad \xi' = \xi\theta$$
5. Если $\langle f \rangle$ содержит компоненту $X \leftarrow P'$, то
$$\mu = D \leftarrow P' \quad \text{и} \quad \xi' = \xi\ (D/X)\ \theta$$
для некоторого $D \in \mathcal{D}_X$.

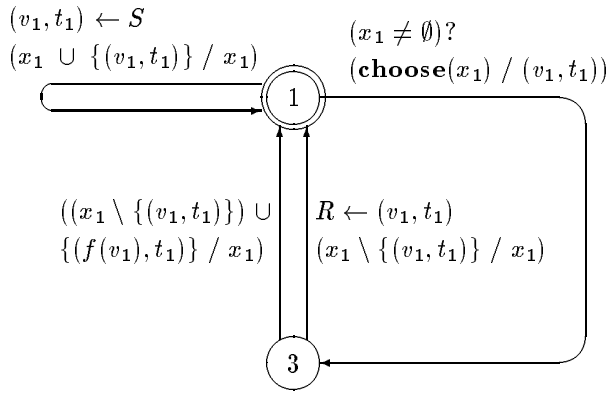


Рис. 12: $Buffer'_1$.

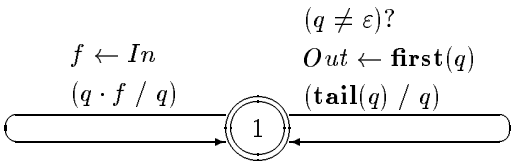


Рис. 13: $Spec'$.

6.5.3 Отношение перехода на состояниях системы

Состоянием системы Σ вида (6.10) называется список Ξ вида

$$\Xi = (\xi_1, \ldots, \xi_n) \qquad (6.24)$$

такой, что для каждого $i \in \{1, \ldots, n\}$ $\quad \xi_i \in S(P_i)$.

Состояние системы Σ можно рассматривать как функцию вида

$$\Xi : Var(\Sigma) \to \mathcal{D}$$

где

$$Var(\Sigma) \stackrel{\text{def}}{=} Var(P_1) \cup \ldots \cup Var(P_n)$$

Множество всевозможных состояний системы Σ обозначается символом $S(\Sigma)$.

Состояние (6.24) называется **начальным**, если для каждого $i \in \{1, \ldots, n\}$ имеет место равенство

$$\xi_i(Init(P_i)) = 1.$$

В данном пункте мы определяем **отношение перехода** "$\to$" вида

$$\to \quad \subseteq \quad S(\Sigma) \times \mathcal{M} \times S(\Sigma)$$

Если тройка (Ξ, μ, Ξ') принадлежит данному отношению, то этот факт будет обозначаться знакосочетанием

$$\Xi \xrightarrow{\mu} \Xi'. \qquad (6.25)$$

Пусть Ξ и Ξ' имеют вид

$$\Xi = (\xi_1, \ldots, \xi_n), \qquad \Xi' = (\xi'_1, \ldots, \xi'_n)$$

Соотношение (6.25) имеет место в одном из следующих случаев.

Внешнее событие:

В данном случае μ имеет вид

$$D \leftarrow P \quad \text{или} \quad P \leftarrow D \qquad (\text{где } P \notin \Sigma)$$

и для некоторого индекса i

- $\xi_i \xrightarrow{\mu} \xi'_i$, и
- $\xi_j = \xi'_j$ для всех $j \neq i$,

Внутреннее событие:

В данном случае $\mu = \tau$, и

1. либо для некоторого индекса i
   - $\xi_i \stackrel{\tau}{\to} \xi'_i$, и
   - $\xi_j = \xi'_j$ для всех $j \neq i$,
2. либо для некоторой пары i, j различных индексов
   - $$\xi_i \xrightarrow{P_j \leftarrow D} \xi'_i \quad \text{и} \quad \xi_j \xrightarrow{D \leftarrow P_i} \xi'_j$$
     для некоторого $D \in \mathcal{D}^*$
   - $\xi_k = \xi'_k$ для всех $k \neq i, j$.

6.6 Эквивалентность систем

Понятие эквивалентности систем позволяет дать точное определение соответствия системы своей спецификации. Мы будем считать, что система Σ соответствует спецификации $Spec$, если имеется эквивалентность между системой Σ и системой (или программой), представляющей спецификацию $Spec$.

6.6.1 Понятие эквивалентности систем

Пусть Σ – некоторая система.

На множестве $S(\Sigma)$ можно ввести структуру ориентированного размеченного графа: в том случае, когда имеет место соотношение (6.25), мы будем считать, что данный граф содержит ребро из Ξ в Ξ' с меткой μ.

Ниже мы будем использовать следующие обозначения: для всех $\Xi_1, \Xi_2 \in S(\Sigma)$

- знакосочетание
  $$\Xi_1 \xrightarrow{\tau^*} \Xi_2$$
  означает, что либо $\Xi_1 = \Xi_2$, либо в определённом выше графе есть путь из Ξ_1 в Ξ_2, каждое ребро которого имеет метку τ.

- знакосочетание

$$\Xi_1 \xrightarrow{\mu_\tau} \Xi_2 \qquad (6.26)$$

означает, что

– или $\mu = \tau$ и $\Xi_1 = \Xi_2$,

– или имеют место соотношения

$$\Xi_1 \xrightarrow{\tau^*} \Xi_1' \xrightarrow{\mu} \Xi_2' \xrightarrow{\tau^*} \Xi_2$$

где Ξ_1', Ξ_2' – некоторые состояния из $S(\Sigma)$.

Системы Σ_1 и Σ_2 называются **эквивалентными** (данный факт обозначается знакосочетанием $\Sigma_1 \sim \Sigma_2$), если существует бинарное отношение

$$R \subseteq S(\Sigma_1) \times S(\Sigma_2)$$

(называемое **бимоделированием**), такое, что

1. для каждого начального состояния Ξ_1 системы Σ_1 существует начальное состояние Ξ_2 системы Σ_2, такое, что
$$(\Xi_1, \Xi_2) \in R \qquad (6.27)$$
2. для каждого начального состояния Ξ_2 системы Σ_2 существует начальное состояние Ξ_1 системы Σ_1, такое, что имеет место (6.27)
3. для
   - каждой пары $(\Xi_1, \Xi_2) \in R$, и
   - каждого состояния $\Xi_1' \in S(\Sigma_1)$, такого, что
$$\Xi_1 \xrightarrow{\mu} \Xi_1'$$
   для некоторого $\mu \in \mathcal{M}$

   существует состояние $\Xi_2' \in S(\Sigma_2)$ такое, что
$$(\Xi_1', \Xi_2') \in R \quad \text{и} \quad \Xi_2 \xrightarrow{\mu_\tau} \Xi_2'$$
4. для
   - каждой пары $(\Xi_1, \Xi_2) \in R$, и
   - каждого состояния $\Xi_2' \in S(\Sigma_2)$, такого, что
$$\Xi_2 \xrightarrow{\mu} \Xi_2'$$
   для некоторого $\mu \in \mathcal{M}$

   существует состояние $\Xi_1' \in S(\Sigma_1)$ такое, что
$$(\Xi_1', \Xi_2') \in R \quad \text{и} \quad \Xi_1 \xrightarrow{\mu_\tau} \Xi_1'$$

Аналогичным образом формулируется понятие эквивалентности программ.

Нетрудно доказать, что "$\sim$" действительно является отношением эквивалентности (т.е. оно рефлексивно, симметрично и транзитивно).

6.6.2 Свойства отношения эквивалентности систем

Ниже мы приводим без доказательства два утверждения, относящиеся к понятию эквивалентности систем. Данные утверждения будут использоваться в пункте 6.7 для верификации системы UBP.

Утверждение 1.
Пусть программа P' получается из программы P при помощи редукции. Тогда $P \sim P'$.

Для доказательства данного утверждения нужно построить бимоделирование R между P и P'.

Пусть P' получается из P преобразованиями, описанными в пункте 6.4.2. Мы будем использовать те обозначения для удаляемых рёбер, которые приведены в этом пункте.

Рассмотрим два случая.

1. $\langle f_0 \rangle$ – внутреннее действие. В этом случае
$$R \stackrel{\text{def}}{=} \{(\xi, \xi) \mid \xi \in S(P)\} \cup \{(\xi, \xi') \mid f_0 : \xi \xrightarrow{\tau} \xi'\}$$
2. $\langle f_0 \rangle$ содержит компоненту, связанную с приёмом или передачей сообщения (и, следовательно, все действия $\langle f_1 \rangle, \ldots, \langle f_k \rangle$ – внутренние). В этом случае
$$R \stackrel{\text{def}}{=} \{(\xi, \xi) \mid \xi \in S(P)\} \cup \{(\xi, \xi') \mid f_i : \xi \xrightarrow{\tau} \xi',\ i = 1, \ldots, k\}.$$

Утверждение 2.
Пусть Σ и Σ' – системы вида

$$\Sigma = (P_1, \ldots, P_n), \quad \Sigma' = (P_1', \ldots, P_n')$$

и для каждого $i = 1, \ldots, n$ $\quad P_i \sim P_i'$. Тогда $\Sigma \sim \Sigma'$.

6.7 Верификация системы UBP

6.7.1 Задача верификации

Задача верификации системы UBP заключается в том, чтобы доказать эквивалентность

$$(Sender, Receiver, Buffer_1, Buffer_2) \sim Spec. \qquad (6.28)$$

Согласно утверждениям 1 и 2 из пункта 6.6.2, соотношение (6.28) эквивалентно соотношению

$$(Sender', Receiver', Buffer'_1, Buffer'_2) \sim Spec'. \qquad (6.29)$$

Ниже мы будем обозначать символом Σ систему в левой части соотношения (6.29), а входящие в неё программы – символами S, R, B_1 и B_2 соответственно.

Для доказательства соотношения (6.29) необходимо определить бимоделирование между Σ и $Spec'$.

6.7.2 Неформальное обсуждение функционирования системы Σ

Функционирование системы Σ можно интерпретировать как создание и обработку пакетов.

Для каждого $j \geq 1$ пакет с номером j проходит несколько стадий при своём движении в системе Σ:

1. создание j–го пакета (в программе S, в момент поступления в систему j–го сообщения от программы In),
2. пересылка j–го пакета от одной программы системы Σ к другой программе системы Σ по рёбрам на рис. 1,
3. модификация j–го пакета (в программах B_1, B_2 или S),
4. уничтожение j–го пакета (в программе R, в момент выдачи из системы j–го сообщения программе Out).

Отметим, что для каждого состояния Ξ системы Σ имеет место эквиваленция

$$\left(\begin{array}{l}\text{в состоянии } \Xi \\ \text{в системе имеется} \\ \text{пакет с номером } j\end{array}\right) \quad \Leftrightarrow \quad \Xi \models out \leq j \leq in. \tag{6.30}$$

Каждый возможный вариант функционирования системы Σ соответствует некоторому пути π

$$\pi = \quad (\Xi_1 \xrightarrow{\mu_1} \Xi_2 \xrightarrow{\mu_2} \ldots) \tag{6.31}$$

в ориентированном графе с множеством вершин $S(\Sigma)$, который был определён в пункте 6.6.1.

Поскольку каждый оператор в программах из системы Σ не уменьшает значения переменных in и out, то, следовательно, справедливы неравенства

$$\begin{array}{l}\Xi_1(in) \leq \Xi_2(in) \leq \ldots \\ \Xi_1(out) \leq \Xi_2(out) \leq \ldots\end{array} \tag{6.32}$$

Следовательно,

- номер первого состояния в последовательности (6.31), в котором j–й пакет присутствует в какой-либо из программ системы Σ, является наименьшим из i, удовлетворяющих неравенству

$$j \leq \Xi_i(in)$$

- номер последнего состояния в последовательности (6.31), в котором j–й пакет присутствует в какой-либо из программ системы Σ, является наибольшим из k, удовлетворяющих неравенству

$$\Xi_k(out) \leq j$$

Таким образом, те состояния из последовательности (6.31), в которых j–й пакет присутствует в какой-либо из программ системы Σ, образуют связную подпоследовательность $\pi(j)$, которая

- является пустой, в том случае, когда при данном варианте функционирования j–е сообщение вообще не поступало в систему,
- имеет вид

$$\pi(j) = (\Xi_i \xrightarrow{\mu_i} \Xi_{i+1} \xrightarrow{\mu_{i+1}} \ldots \xrightarrow{\mu_{k-1}} \Xi_k) \tag{6.33}$$

  в том случае, когда j–е сообщение поступило в систему в момент времени i и выдаётся из системы программе Out в момент времени k,
- является бесконечной

$$\pi(j) = \quad (\Xi_i \xrightarrow{\mu_i} \Xi_{i+1} \xrightarrow{\mu_{i+1}} \ldots) \tag{6.34}$$

  в том случае, когда j–е сообщение поступило в систему в момент времени i, но никогда не выдаётся программе Out.

6.7.3 Определение бимоделирования между Σ и $Spec'$

Искомое бимоделирование R мы определяем как совокупность пар (Ξ_1, Ξ_2) из $S(\Sigma) \times S(Spec')$, удовлетворяющих условию

$$\Xi_1(u) = \Xi_2(q)$$

где

- $u \stackrel{\text{def}}{=} (s[out], \ldots, s[in])$
  (список сообщений, которые поступили в систему, но ещё пока не выданы)
- q – переменная программы $Spec'$.

Для доказательства того, что данное отношение действительно является бимоделированием, будут использоваться некоторые инварианты системы Σ, которые определяются в пункте 6.7.4

6.7.4 Инварианты системы Σ

Понятие инварианта

Инвариантом системы называется произвольная формула с переменными из множества $Var(\Sigma)$, которая является истинной в каждом состоянии данной системы, достижимом из начального состояния. Формула φ называется **истинной** в состоянии Ξ, если

$$\Xi(\varphi) = 1 \tag{6.35}$$

Соотношение (6.35) мы будем также записывать в виде знакосочетания

$$\Xi \models \varphi$$

Доказательство того, что некоторая формула φ является инвариантом системы, обычно состоит из доказательства следующих двух утверждений:

- (6.35) верно для каждого начального состояния Ξ,
- для каждой пары Ξ, Ξ' состояний, таких, что

$$\Xi \xrightarrow{\mu} \Xi' \quad \text{для некоторого } \mu \in \mathcal{M}$$

из (6.35) следует соотношение $\Xi'(\varphi) = 1$.

Вспомогательные выражения и формулы

В рассуждениях о системе Σ будут использоваться следующие выражения с переменными из множества $Var(\Sigma)$:

- $x_i(j) \stackrel{\text{def}}{=} x_i \cap \{(d,j) \mid d \in \mathcal{D}\} \qquad (j \geq 1,\ i = 1,2)$,
  т.е. $x_i(j)$ есть множество пакетов с номером j в буфере B_i,
- $f^*(d) \stackrel{\text{def}}{=} \{f^k(d) \mid k \geq 0\} \qquad (d \in \mathcal{D})$,
  т.е. $f^*(d)$ есть множество, состоящее из d и всех возможных искажений значения d.

Отметим, что из (6.9) следует импликация:

$$(d' \in f^*(d)) \ \wedge \ (d \in f^*(d')) \qquad \Rightarrow \qquad d = d'. \quad (6.36)$$

Ниже для каждого списка $e_1, \ldots, e_n$ формул их конъюнкция $e_1 \wedge \ldots \wedge e_n$ будет обозначаться также знакосочетанием

$$\left\{ \begin{array}{l} e_1 \\ \ldots \\ e_n \end{array} \right\}.$$

Определим вспомогательные формулы, которые будут использоваться при определении инвариантов системы Σ.

- $PASSING(j) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} out \leq j \leq in \\ h[j] = 0 \end{array} \right\}$
  (j–е сообщение находится в системе, но пока ещё нет гарантии, что оно было передано программе R успешно).
- $S(j) \stackrel{\text{def}}{=} S_2(j) \vee S_3(j)$, где

$$S_2(j) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} at_S = 2 \\ in = j \end{array} \right\}, \qquad S_3(j) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} at_S = 3 \\ k = j \end{array} \right\}$$

  (либо в программе S создан пакет с номером j, либо в программу S пришёл от B_2 пакет с номером j)
- $R(j) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} at_R = 2 \\ l = j \end{array} \right\}$
  (в программу R пришёл пакет с номером j)
- $B_i(j) \stackrel{\text{def}}{=} (x_i(j) \neq \emptyset) \qquad (i = 1,2)$
  (в буфере B_i имеется пакет с номером j)
- $\Sigma(j) \stackrel{\text{def}}{=} \bigvee\limits_{P \in \Sigma} P(j)$
  (пакет с номером j присутствует в одной из программ из системы Σ)
- $MUTEX(j) \stackrel{\text{def}}{=} \bigwedge\limits_{P \neq P' \in \Sigma} \neg \left\{ \begin{array}{l} P(j) \\ P'(j) \end{array} \right\}$
  (пакет с номером j не может присутствовать одновременно в двух разных программах из системы Σ).

Определение инвариантов

Инварианты системы Σ имеют следующий вид:

F_1: $(out \leq j \leq in) \ \rightarrow \left\{ \begin{array}{l} \Sigma(j) \\ MUTEX(j) \end{array} \right\}$

F_2: $out \leq in + 1$

F_3: $(d,j) \in x_1 \ \rightarrow \left\{ \begin{array}{l} PASSING(j) \\ (d = \$) \rightarrow (s[j] = r[j]) \\ (d \neq \$) \rightarrow (d \in f^*(s[j])) \end{array} \right\}$

F_4: $(d,j) \in x_2 \ \rightarrow \left\{ \begin{array}{l} PASSING(j) \\ d \in f^*(r[j]) \\ r[j] \in f^*(s[j]) \end{array} \right\}$

F_5: $(h[j] = 1) \ \rightarrow \left\{ \begin{array}{l} r[j] = s[j] \\ j \leq in \end{array} \right\}$

F_6: $|x_i(j)| \leq 1 \qquad (i = 1,2)$

F_7: $(at_S = 2) \ \rightarrow \left\{ \begin{array}{l} PASSING(in) \\ g = s[in] \end{array} \right\}$

F_8: $(at_S = 3) \ \rightarrow \left\{ \begin{array}{l} PASSING(k) \\ g \in f^*(r[k]) \\ r[k] \in f^*(s[k]) \end{array} \right\}$

F_9: $(at_R = 2) \ \rightarrow \left\{ \begin{array}{l} PASSING(l) \\ (m = \$) \rightarrow (r[l] = s[l]) \\ (m \neq \$) \rightarrow (m \in f^*(s[l])) \end{array} \right\}$

F_{10}: $(h[out] = 1) \ \rightarrow \ (at_R = 3)$

Доказательство того, что формулы F_1–F_{10} действительно являются инвариантами системы Σ, производится непосредственной проверкой: сначала устанавливается, что формула

$$\bigwedge_{i=1}^{10} F_i$$

истинна во всех начальных состояниях системы Σ, а затем рассматриваются индуктивные переходы. В данном доказательстве используется импликация (6.36).

Интерпретация инвариантов

Инвариант F_1 можно интерпретировать как утверждение о том, что для каждого $j \geq 1$ и для каждого пути (6.31) совокупность всех состояний, входящих в подпоследовательность $\pi(j)$, можно разбить на 4 непересекающихся класса (некоторые из которых могут быть пустыми), в которых соответственно истинны формулы

$$S(j),\ R(j),\ B_1(j),\ B_2(j) \qquad (6.37)$$

Мы будем обозначать данные классы теми же символами, что и формулы из списка (6.37), которые в них истинны.

В свою очередь, каждый из перечисленных классов можно поделить на несколько непересекающихся подклассов (некоторые из которых могут быть пустыми), с учётом более детальной информации о состояниях:

1. класс $S(j)$ разбивается на 3 подкласса, в которых истинны формулы

$$S_2(j) \qquad (A_1(j))$$

$$\left\{ \begin{array}{l} S_3(j) \\ g \neq s[j] \end{array} \right\} \qquad (A_2(j))$$

$$\left\{ \begin{array}{l} S_3(j) \\ g = s[j] \end{array} \right\} \qquad (A_3(j))$$

соответственно

2. класс $R(j)$ разбивается на 3 подкласса, в которых истинны формулы

$$\left\{ \begin{array}{l} R(j) \\ m \neq s[j], \$ \end{array} \right\} \qquad (A_4(j))$$

$$\left\{ \begin{array}{l} R(j) \\ m = s[j] \end{array} \right\} \qquad (A_5(j))$$

$$\left\{ \begin{array}{l} R(j) \\ m = \$ \end{array} \right\} \qquad (A_6(j))$$

соответственно

3. класс $B_1(j)$ (который, на основании инварианта F_6, или пуст, или состоит из одного элемента) разбивается на 3 подкласса, в которых истинны формулы

$$\left\{ \begin{array}{l} x_1(j) = \{(d, j)\} \\ d \in f^*(s[j]) \setminus \{s[j], \$\} \end{array} \right\} \qquad (A_7(j))$$

$$x_1(j) = \{(s[j], j)\} \qquad (A_8(j))$$

$$x_1(j) = \{(\$, j)\} \qquad (A_9(j))$$

соответственно

4. класс $B_2(j)$ (который тоже или пуст, или состоит из одного элемента) разбивается на 2 подкласса, в которых истинны формулы

$$\left\{ \begin{array}{l} x_2(j) = \{(d, j)\} \\ d \in f^*(s[j]) \setminus \{s[j]\} \end{array} \right\} \qquad (A_{10}(j))$$

$$x_2(j) = \{(s[j], j)\} \qquad (A_{11}(j))$$

соответственно

Получившиеся 11 классов состояний из подпоследовательности $\pi(j)$ мы будем обозначать теми же символами, что и определяющие их формулы, т.е. $A_1(j)$–$A_{11}(j)$. Данные классы можно интерпретировать как **состояния j–го пакета.**

Состояния $A_1(j)$–$A_{11}(j)$ могут быть представлены в виде диаграммы, изображённой на рис. 14. Стрелки на этой диаграмме изображают изменение состояния и места j–го пакета в системе Σ. Состояние j–пакета может изменяться в соответствии со стрелками на диаграмме (при таких переходах система Σ выполняет только внутренние действия). Для каждой пары Ξ_i, Ξ_{i+1} соседних состояний из $\pi(j)$

- либо Ξ_i и Ξ_{i+1} находятся в одном и том же классе,
- либо классы, в которых находятся Ξ_i и Ξ_{i+1}, соединены стрелкой.

Ниже мы будем пользоваться следующей леммой.

Лемма.

Пусть пара $A_p(j)$, $A_q(j)$ состояний j–го пакета такова, что в диаграмме на рис. 14 существует стрелка из $A_p(j)$ в $A_q(j)$.

Тогда для каждого состояния Ξ системы Σ, такого, что

$$\Xi \models A_p(j)$$

существует состояние Ξ' системы Σ, такое, что

$$\Xi' \models A_q(j) \quad \text{и} \quad \Xi \xrightarrow{\tau} \Xi'.$$

6.7.5 Проверка условий 1 и 2

Условия 1 и 2 из определения бимоделирования верны по причине того, что

- равенство
$$\Xi_1(u) = \varepsilon$$
имеет место для каждого начального состояния Ξ_1 системы Σ, и
- равенство
$$\Xi_2(q) = \varepsilon$$
имеет место для каждого начального состояния Ξ_2 системы $Spec'$.

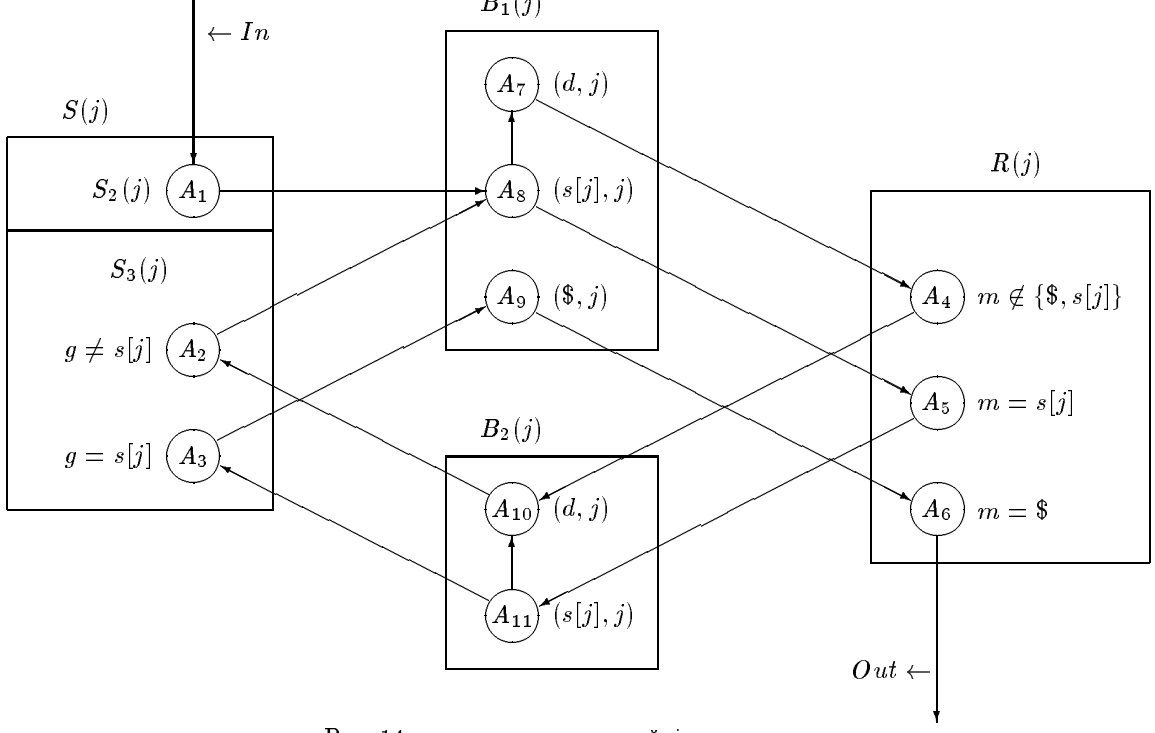


Рис. 14: диаграмма состояний j–го пакета.

6.7.6 Проверка условия 3

Пусть имеются состояния Ξ_1, Ξ_2, Ξ_1' и событие μ, обладающие свойствами, изложенными в формулировке условия 3 в пункте 6.6.1.

Для определения искомого состояния Ξ_2' мы рассмотрим три возможных вида события μ:

$$\mu = \tau, \quad \mu = (d \leftarrow In), \quad \mu = (Out \leftarrow d). \tag{6.38}$$

Ниже знакосочетание $P : i \to j$ (где P – произвольная программа из Σ) означает, что

$$\Xi_1(at_P) = i \quad \text{и} \quad \Xi_1'(at_P) = j$$

1. $\mu = \tau$.

   В этом случае $\Xi_1'(u) = \Xi_1(u)$.

   Определим $\Xi_2' \stackrel{\text{def}}{=} \Xi_2$.

2. $\mu = (d \leftarrow In)$.

   Это возможно только в случае $S : 1 \to 2$. В этом случае

   $$\Xi_1'(in) = \Xi_1(in) + 1, \qquad \Xi_1'(s[in]) = d$$

   Таким образом,

   $$\Xi_1'(u) = \Xi_1'(s[out], \ldots, s[in]) =$$
   $$(\Xi_1'(s[out]), \ldots, \Xi_1'(s[in-1]), \Xi_1'(s[in])) =$$
   $$(\Xi_1(s[out]), \ldots, \Xi_1(s[in]), d) = \Xi_1(u) \cdot d$$

   Определим

   $$\Xi_2'(q) \stackrel{\text{def}}{=} \Xi_2(q) \cdot d, \quad \Xi_2'(f) \stackrel{\text{def}}{=} d$$

   Переход от Ξ_2 к Ξ_2' осуществляется по левому ребру на рис. 13.

3. $\mu = (Out \leftarrow d)$.

   Это возможно только в случае $R : 3 \to 3$. В этом случае

   $$\Xi_1 \models (h[out] = 1) \tag{6.39}$$

   $$\Xi_1(r[out]) = d \tag{6.40}$$

   $$\Xi_1'(out) = \Xi_1(out) + 1 \tag{6.41}$$

   Из (6.39) на основании F_5 следует соотношение

   $$\Xi_1 \models \left\{ \begin{array}{l} r[out] = s[out] \\ out \leq in \end{array} \right\} \tag{6.42}$$

   Равенство $\Xi_1(u) = \Xi_2(q)$ можно записать следующим образом:

   $$\Xi_1(s[out], s[out+1], \ldots, s[in]) =$$
   $$= \Xi_2(\mathbf{first}(q)) \cdot \Xi_2(\mathbf{tail}(q))$$

   откуда следуют равенства

   $$\Xi_1(s[out]) = \Xi_2(\mathbf{first}(q)) \tag{6.43}$$

   и

   $$\Xi_1(s[out+1], \ldots, s[in]) = \Xi_2(\mathbf{tail}(q)) \tag{6.44}$$

   Из (6.43), (6.42) и (6.40) следует равенство

   $$\Xi_2(\mathbf{first}(q)) = d \tag{6.45}$$

   Из (6.41) и (6.44) следует равенство

   $$\Xi_1'(u) = \Xi_2(\mathbf{tail}(q)) \tag{6.46}$$

Определим

$$\Xi_2'(q) \stackrel{\text{def}}{=} \Xi_2(\mathbf{tail}(q)), \quad \Xi_2'(f) \stackrel{\text{def}}{=} \Xi_2(f)$$

Переход от Ξ_2 к Ξ_2' осуществляется по правому ребру на рис. 13.

6.7.7 Проверка условия 4

Пусть имеются состояния Ξ_1, Ξ_2, Ξ_2' и событие μ, обладающие свойствами, изложенными в формулировке условия 4 в пункте 6.6.1.

Для определения искомого состояния Ξ_1' мы рассмотрим возможные виды события μ (см. (6.38)). Заметим, что случай $\mu = \tau$ невозможен ввиду структуры программы $Spec'$.

Случай $\mu = (d \leftarrow In)$

Данный вид μ возможен только в том случае, когда переход от Ξ_2 к Ξ_2' осуществляется по левому ребру на рис. 13. Следовательно,

$$\Xi_2'(f) = d, \qquad \Xi_2'(q) = \Xi_2(q) \cdot d \quad (= \Xi_1(u) \cdot d)$$

Нетрудно доказать, что существует состояние Ξ системы Σ, обладающее следующими свойствами:

$$\Xi_1 \xrightarrow{\tau^*} \Xi \tag{6.47}$$

и

$$\Xi \models (at_S = 1)$$

Из (6.47) следует, что $\Xi_1(u) = \Xi(u)$, откуда получаем:

$$\Xi(s[out], \ldots, s[in]) = \Xi_2(q).$$

Определим Ξ_1' следующим образом:

$$\begin{aligned}
&\Xi_1'(at_S) \stackrel{\text{def}}{=} 2\\
&\Xi_1'(in) \stackrel{\text{def}}{=} \Xi_1(in) + 1\\
&\Xi_1'(g) \stackrel{\text{def}}{=} d\\
&\Xi_1'(s[in]) \stackrel{\text{def}}{=} d
\end{aligned}$$

(значение Ξ_1' на неуказанных переменных совпадает со значением Ξ_1 на этих переменных).

Имеем:

$$\Xi \xrightarrow{\mu} \Xi_1'$$

и

$$\begin{aligned}
&\Xi_1'(u) = \Xi_1'(s[out], \ldots, s[in]) =\\
&= (\Xi_1'(s[out]), \ldots, \Xi_1'(s[in-1]), \Xi_1'(s[in])) =\\
&= (\Xi_1(s[out]), \ldots, \Xi_1(s[in]), d) =\\
&= \Xi_1(s[out], \ldots, s[in]) \cdot d = \Xi_1(u) \cdot d =\\
&= \Xi_2(q) \cdot d = \Xi_2'(q)
\end{aligned}$$

т.е. $(\Xi_1', \Xi_2') \in R$.

Случай $\mu = (Out \leftarrow d)$

Данный вид μ возможен только в том случае, когда переход от Ξ_2 к Ξ_2' осуществляется по правому ребру на рис. 13. Следовательно,

$$\begin{aligned}
&\Xi_2(q) \neq \varepsilon\\
&\mathbf{first}(\Xi_2(q)) = d\\
&\mathbf{tail}(\Xi_2(q)) = \Xi_2'(q)
\end{aligned} \tag{6.48}$$

Из (6.48) следует, что

$$\Xi_1(s[out], \ldots, s[in]) = \Xi_1(u) = \Xi_2(q) = d \cdot \Xi_2'(q)$$

Поэтому

$$\Xi_1 \models (out \leq in) \tag{6.49}$$

$$\Xi_1(s[out]) = d \tag{6.50}$$

$$\Xi_1(s[out+1], \ldots, s[in]) = \Xi_2'(q) \tag{6.51}$$

Рассмотрим два подслучая.

1. $\Xi_1 \models (h[out] = 1)$.

   Из инвариантов (F_{10}) и (F_5) следует, что в этом случае

   $$\Xi_1(at_R) = 3$$

   и

   $$\Xi_1 \models r[out] = s[out] \quad (= d)$$

   Определим Ξ_1' следующим образом:

   $$\begin{aligned}
   &\Xi_1'(at_R) \stackrel{\text{def}}{=} 3\\
   &\Xi_1'(out) \stackrel{\text{def}}{=} \Xi_1(out) + 1
   \end{aligned}$$

   (значение Ξ_1' на неуказанных переменных совпадает со значением Ξ_1 на этих переменных).

   Имеем:

   $$\Xi_1 \xrightarrow{\mu} \Xi_1'$$

   и

   $$\begin{aligned}
   &\Xi_1'(u) = \Xi_1'(s[out], \ldots, s[in]) =\\
   &= (\Xi_1'(s[out]), \ldots, \Xi_1'(s[in])) =\\
   &= (\Xi_1(s[out+1]), \ldots, \Xi_1(s[in])) = \Xi_2'(q)
   \end{aligned}$$

   т.е. $(\Xi_1', \Xi_2') \in R$.

2. $$\Xi_1 \models h[out] = 0 \tag{6.52}$$

   Из (6.49) и из рассуждений в пункте 6.7.4 следует, что существует индекс $i \in \{1, \ldots, 11\}$, такой, что

   $$\Xi_1 \models A_i(out)$$

   Для каждого $j \geq 1$ из любого состояния j–го пакета существует путь в диаграмме на рис. 14 в состояние $A_6(j)$. В частности, существует путь из $A_i(out)$ в $A_6(out)$. Поэтому из леммы в конце

пункта 6.7.4 следует, что существует состояние Ξ системы Σ, такое, что

$$\Xi_1 \xrightarrow{\tau^*} \Xi \quad \text{и} \quad \Xi \models A_6(out)$$

Нетрудно доказать, что существует состояние Ξ' системы Σ со следующими свойствами:

$$\Xi \xrightarrow{\tau^*} \Xi' \quad \text{и} \quad \Xi' \models (h[out] = 1)$$

Очевидно, что имеют место соотношения

$$\Xi' \models (out \leq in)$$
$$\Xi'(s[out]) = d$$
$$\Xi'(s[out+1], \ldots, s[in]) = \Xi'_2(q)$$

Оставшаяся часть доказательства для подслучая 2 повторяет доказательство для подслучая 1, с заменой Ξ_1 на Ξ'.

Глава 7

Процессная алгебра

7.1 Действия

Мы будем предполагать, что задано некоторое множество Act, имеющее вид дизъюнктного объединения

$$Act \;=\; \Delta \sqcup \overline{\Delta} \sqcup \{\tau\}$$

причём между множествами Δ и $\overline{\Delta}$ имеется взаимно однозначное соответствие, при котором каждому $a \in \Delta$ соответствует элемент $\overline{a} \in \overline{\Delta}$.

Элементы множества Act называются **действиями**. Для каждого $a \in \Delta$ действия a и $\overline{a}$ называются **комплементарными**. Действие τ называется **невидимым**.

7.2 Процессные графы

Процессные графы предназначены для наглядного изображения дискретных процессов, работа которых заключается в исполнении действий из множества Act.

Процессный граф (ПГ) представляет собой граф P с выделенной начальной вершиной $Start(P)$, каждое ребро α которого помечено действием $\langle\alpha\rangle \in Act$.

Работа процесса, изображаемого графом P, происходит путём обхода P, начиная с вершины $Start(P)$. В каждый такт времени процесс выбирает ребро α, выходящее из той вершины, в которой он в данный момент находится, и переходит в вершину, являющуюся концом α, выполняя при этом действие $\langle\alpha\rangle$.

На множестве всех ПГ можно определить алгебраические операции. Во всех бинарных операциях множества вершин графов-аргументов предполагаются непересекающимися.

пустой процесс:

ПГ $\mathbf{0}$ состоит из одной вершины, и не имеет рёбер.

префиксное действие:

Для каждого $a \in Act$ и каждого ПГ P знакосочетание $a.P$ обозначает ПГ, получаемый добавлением к P

- новой вершины, являющейся корнем в $a.P$, и
- ребра с меткой a из $Start(a.P)$ в $Start(P)$

выбор:

ПГ $P + Q$ получается добавлением к $P \cup Q$

- новой вершины, являющейся корнем в $P+Q$,
- новых рёбер: для каждого ребра из начальной вершины P или Q

  $$Start(P) \xrightarrow{\alpha} N \quad \text{или} \quad Start(Q) \xrightarrow{\alpha} N$$

  мы добавляем ребро

  $$Start(P+Q) \xrightarrow{\alpha} N$$

Таким образом, процесс $P+Q$, работает либо как P, либо как Q.

параллельная композиция:

Вершинами $P \mid Q$ являются пары вида (N_1, N_2), где N_1 – вершина P, и N_2 – вершина Q.

Начальной вершиной $P \mid Q$ является пара

$$(Start(P), Start(Q))$$

Для каждой вершины (N_1, N_2) граф $P|Q$ содержит следующие рёбра, выходящие из (N_1, N_2):

- $(N_1, N_2) \xrightarrow{a} (N_1', N_2)$, если P содержит ребро $N_1 \xrightarrow{a} N_1'$
- $(N_1, N_2) \xrightarrow{a} (N_1, N_2')$, если Q содержит ребро $N_2 \xrightarrow{a} N_2'$
- $(N_1, N_2) \xrightarrow{\tau} (N_1', N_2')$, если P содержит ребро вида $N_1 \xrightarrow{a} N_1'$, а Q – ребро вида $N_2 \xrightarrow{\overline{a}} N_2'$, где $a \neq \tau$, и если a имеет вид $\overline{b}$, где $b \in \Delta$, то $\overline{a} \stackrel{\text{def}}{=} b$.

Таким образом, функционирование $P|Q$ заключается в поочерёдном функционировании P и Q, с возможностью одновременного исполнения комплементарных действий, при котором эти действия "гасят" друг друга, порождая невидимое совместное действие τ.

переименование:

Для каждой функции $f : \Delta \to \Delta$ ПГ $f(P)$ получается из P заменой метки a каждого ребра на $f(a)$, где знакосочетание $f(a)$ для $a \notin \Delta$ обозначает

- действие $\overline{f(b)}$, если $a = \overline{b}$, где $b \in \Delta$, и
- действие τ, если $a = \tau$

блокировка:

Для каждого $L \subseteq \Delta$ ПГ $P \setminus L$ получается из P удалением каждого ребра, метка a которого не равна τ и удовлетворяет условию

$$name(a) \in L$$

где для каждого $a \in \Delta \quad name(a) \stackrel{\text{def}}{=} name(\overline{a}) \stackrel{\text{def}}{=} a$

7.3 Процессные выражения

Процессные выражения являются алгебраической формой описания процессов, работа которых заключается в исполнении действий из множества Act.

Множество $\mathcal{E}$ **процессных выражений (ПВ)** определяется индуктивно следующим образом.

процессные переменные:

Мы предполагаем, что задано счётное множество ПВ V, элементы которого называются **процессными переменными**. Каждая процессная переменная принадлежит $\mathcal{E}$.

пустой процесс:

Символ **0** принадлежит $\mathcal{E}$.

префиксное действие:

Если $a \in Act$ и $P \in \mathcal{E}$, то $a.P \in \mathcal{E}$.

выбор:

Если P и $Q \in \mathcal{E}$, то $P + Q \in \mathcal{E}$.

параллельная композиция:

Если P и $Q \in \mathcal{E}$, то $P \mid Q \in \mathcal{E}$.

переименование:

Если f – функция из Δ в Δ, и $P \in \mathcal{E}$, то $f(P) \in \mathcal{E}$.

блокировка:

Если $L \subseteq \Delta$ и $P \in \mathcal{E}$, то $P \setminus L \ \in \mathcal{E}$.

рекурсивное определение:

Если S – имя системы уравнений вида

$$S = \{X_i = P_i \mid i = 1, \ldots, n,\ X_i \in \Pi V,\ P_i \in \mathcal{E}\} \quad (7.1)$$

то для каждого $i \in \{1, \ldots, n\}$ символ S_i принадлежит $\mathcal{E}$.

7.4 Отношения перехода на $\mathcal{E}$

На множестве $\mathcal{E}$ можно определить совокупность бинарных отношений

$$\{\delta_a \subseteq \mathcal{E}^2 \mid a \in Act\}$$

называемых **отношениями перехода**.

Если $(P, P') \in \delta_a$ для некоторого $a \in Act$, то этот факт мы будем записывать в виде знакосочетания

$$P \xrightarrow{a} P' \quad (7.2)$$

и интерпретировать его как следующее высказывание: процесс P может выполнить действие a, и после этого вести себя так же, как процесс P'.

Для каждого $P \in \mathcal{E}$ символ $\delta_a(P)$ обозначает множество

$$\{P' \in \mathcal{E} \mid (P, P') \in \delta_a\}$$

Отношения перехода определяются при помощи излагаемых ниже правил, в которых символы P, P', Q, Q' обозначают ПВ, $a \in Act$, $L \subseteq \Delta$.

1. $a.P \xrightarrow{a} P$

2. если $P \xrightarrow{a} P'$, то

$$P + Q \xrightarrow{a} P' \quad \text{и} \quad Q + P \xrightarrow{a} P'$$

3. если $P \xrightarrow{a} P'$, то

$$P \mid Q \xrightarrow{a} P' \mid Q \quad \text{и} \quad Q \mid P \xrightarrow{a} Q \mid P'$$

4. если $P \xrightarrow{a} P'$, $Q \xrightarrow{\overline{a}} Q'$ и $a \neq \tau$, то

$$P \mid Q \xrightarrow{\tau} P' \mid Q' \quad \text{и} \quad Q \mid P \xrightarrow{\tau} Q' \mid P'$$

5. если $P \xrightarrow{a} P'$ и f - функция из Δ в Δ, то

$$f(P) \xrightarrow{f(a)} f(P')$$

6. если $P \xrightarrow{a} P'$ и $a = \tau$ или $name(a) \notin L$, то

$$P \setminus L \xrightarrow{a} P' \setminus L$$

7. для каждой системы (7.1) и каждого $i \in \{1, \ldots, n\}$, если $P_i[\vec{X} := \vec{S}] \xrightarrow{a} P'$, то

$$S_i \xrightarrow{a} P'$$

где $P_i[\vec{X} := \vec{S}]$ получается из P_i подстановкой ПВ из списка $\vec{S} = (S_1, \ldots, S_n)$ вместо соответствующих переменных из списка $\vec{X} = (X_1, \ldots, X_n)$.

7.5 Бинарные отношения на $\mathcal{E}$

7.5.1 Сильная конгруэнция

Обозначим символом $\mathcal{H}$ функцию вида

$$\mathcal{H} : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathcal{E}^2)$$

которая сопоставляет каждому отношению $R \subseteq \mathcal{E}^2$ отношение $\mathcal{H}(R)$, состоящее из всех пар $(P, Q) \in \mathcal{E}^2$, таких, что для каждого $a \in Act$

- $\forall\ P' \in \delta_a(P) \quad \exists\ Q' \in \delta_a(Q) : \quad (P', Q') \in R$
- $\forall\ Q' \in \delta_a(Q) \quad \exists\ P' \in \delta_a(P) : \quad (P', Q') \in R$

Нетрудно доказать, что функция $\mathcal{H}$ – монотонна, т.е. имеет место импликация

$$R_1 \subseteq R_2 \quad \Rightarrow \quad \mathcal{H}(R_1) \subseteq \mathcal{H}(R_2)$$

Отношение $R \subseteq \mathcal{E}^2$ называется **бимоделированием (б.м.)**, если $R \subseteq \mathcal{H}(R)$. Например, отношение $Id_{\mathcal{E}}$, состоящее из всех пар вида (P, P), является б.м.

Сильной конгруэнцией на $\mathcal{E}$ называется отношение $\sim$, определяемое как объединение всех б.м.:

$$\sim \stackrel{\text{def}}{=} \bigcup_{R_i - \text{б.м.}} R_i$$

Отношение $\sim$ является

1. б.м., т.к. для каждого б.м. R_i

$$R_i \subseteq \mathcal{H}(R_i) \subseteq \mathcal{H}(\bigcup_{R_i - \text{б.м.}} R_i)$$

и следовательно

$$\bigcup_{R_i - \text{б.м.}} R_i \subseteq \mathcal{H}(\bigcup_{R_i - \text{б.м.}} R_i)$$

2. наибольшей неподвижной точкой функции $\mathcal{H}$, т.к.

$$\sim \subseteq \mathcal{H}(\sim) \quad \Rightarrow \quad \mathcal{H}(\sim) \subseteq \mathcal{H}(\mathcal{H}(\sim))$$

т.е. $\mathcal{H}(\sim)$ – б.м., и, следовательно, $\mathcal{H}(\sim) \subseteq \sim$

3. отношением эквивалентности, т.к.

   (a) *рефлексивность* следует из того, что $Id_{\mathcal{E}}$ – б.м., т.е. $Id_{\mathcal{E}} \subseteq \sim$

   (b) *симметричность* следует из того, что

$$R - \text{б.м.} \quad \Rightarrow \quad R^{-1} - \text{б.м.}$$

   (c) *транзитивность* следует из того, что $\sim \circ \sim$ – б.м., т.е. $\sim \circ \sim \subseteq \sim$

7.5.2 Характеризация сильной конгруэнции модальными формулами

Обозначим символом Fm множество модальных формул, которые строятся из

- элементарных формул **1** и **0**
- булевых связок $\wedge, \vee, \neg$, и
- модальных унарных связок $\Box_a$ и $\Diamond_a \quad (a \in Act)$.

Для каждого $P \in \mathcal{E}$ и каждой формулы $\varphi \in Fm$ **значение** $P(\varphi) \in \{0, 1\}$ определяется индуктивно:

- $P(\mathbf{1}) = 1, \quad P(\mathbf{0}) = 0$
- $P(\varphi_1 \wedge \varphi_2) = P(\varphi_1) \wedge P(\varphi_2)$
- $P(\varphi_1 \vee \varphi_2) = P(\varphi_1) \vee P(\varphi_2)$
- $P(\neg\varphi) = \neg P(\varphi)$
- $P(\Box_a \varphi) = \bigwedge\limits_{P' \in \delta_a(P)} P'(\varphi)$
- $P(\Diamond_a \varphi) = \bigvee\limits_{P' \in \delta_a(P)} P'(\varphi)$

Можно доказать, что

$$P_1 \sim P_2 \quad \Leftrightarrow \quad \forall\, \varphi \in Fm \quad P_1(\varphi) = P_2(\varphi)$$

Импликация "$\Rightarrow$" доказывается индукцией по структуре φ, а для доказательства обратной импликации рассмотрим отношение $R \subseteq \mathcal{E}^2$:

$$(P, Q) \in R \quad \Leftrightarrow \quad \forall\, \varphi \in Fm \quad P(\varphi) = Q(\varphi)$$

Отметим, что $(P_1, P_2) \in R$. Докажем, что R – б.м. Пусть это неверно, т.е.

$$\exists\, (P, Q) \in R : \quad (P, Q) \notin \mathcal{H}(R)$$

Пусть например для пары (P, Q) нарушается свойство (1) из определения отношения $\mathcal{H}(R)$, т.е. для некоторого $a \in Act$ существует $P' \in \delta_a(P)$, такое, что

$$\begin{array}{l} \forall\, Q' \in \delta_a(Q) \quad (P', Q') \notin R \\ \qquad\qquad (\text{т.е. } \exists \varphi \in Fm : P'(\varphi) \neq Q'(\varphi)) \end{array} \tag{7.3}$$

Обозначим символом φ формулу

- $\Diamond_a \mathbf{1}$, если $\delta_a(Q) = \emptyset$,
- $\Diamond_a(\varphi_1 \wedge \ldots \wedge \varphi_n)$, если множество $\delta_a(Q)$ имеет вид

$$\{Q_1, \ldots, Q_n\}$$

и $\forall\, i \in \{1, \ldots, n\} \quad P'(\varphi_i) = 1, \quad Q_i(\varphi_i) = 0$
(существование таких формул φ_i следует из (7.3)).

В обоих случаях $P(\varphi) = 1$ и $Q(\varphi) = 0$, что противоречит предположению $(P, Q) \in R$.

7.5.3 Наблюдаемая эквивалентность и наблюдаемая конгруэнция

Для каждого $a \in Act$

- символ τ_a обозначает отношение

$$(\delta_\tau)^* \circ \delta_a \circ (\delta_\tau)^* \quad \subseteq \mathcal{E}^2$$

  где $(\delta_\tau)^*$ – это рефлексивно-транзитивное замыкание отношения δ_τ

- $\hat{\tau}_a \stackrel{\text{def}}{=} \begin{cases} \tau_a & \text{если } a \neq \tau \\ (\delta_\tau)^* & \text{если } a = \tau \end{cases}$

Обозначим символом $\mathcal{H}_\tau$ функцию вида

$$\mathcal{H}_\tau : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathcal{E}^2)$$

которая сопоставляет каждому отношению $R \subseteq \mathcal{E}^2$ отношение $\mathcal{H}_\tau(R)$, состоящее из всех пар $(P,Q) \in \mathcal{E}^2$, таких, что для каждого $a \in Act$

1. $\forall \ P' \in \tau_a(P) \quad \exists\, Q' \in \hat{\tau}_a(Q) : \quad (P',Q') \in R$
2. $\forall \ Q' \in \tau_a(Q) \quad \exists\, P' \in \hat{\tau}_a(P) : \quad (P',Q') \in R$

Нетрудно доказать, что функция $\mathcal{H}_\tau$ монотонна.

Отношение $R \subseteq \mathcal{E}^2$ называется **наблюдаемым бимоделированием (н.б.м.)**, если $R \subseteq \mathcal{H}_\tau(R)$. Например, отношение $Id_\mathcal{E}$ является н.б.м.

Наблюдаемой эквивалентностью на $\mathcal{E}$ называется объединение всех наблюдаемых бимоделирований:

$$\approx \stackrel{\text{def}}{=} \bigcup_{R_i - \text{н.б.м.}} R_i$$

Нетрудно доказать, что $\sim$ содержится в $\approx$, и $\approx$ является н.б.м., наибольшей неподвижной точкой функции $\mathcal{H}_\tau$, и отношением эквивалентности.

Наблюдаемая конгруэнция – это бинарное отношение $\stackrel{c}{=} \subseteq \mathcal{E}^2$, состоящее из всех пар $(P,Q) \in \mathcal{E}^2$, таких что для каждого $a \in Act$

1. $\forall\, P' \in \tau_a(P) \quad \exists Q' \in \tau_a(Q) : \quad P' \approx Q'$
2. $\forall\, Q' \in \tau_a(Q) \quad \exists P' \in \tau_a(P) : \quad P' \approx Q'$

Бинарное отношение $R \subseteq \mathcal{E} \times \mathcal{E}$ называется **конгруэнцией**, если оно является отношением эквивалентности и сохраняет все операции на $\mathcal{E}$, т.е.

- если $(P,Q) \in R$, то
  - $\forall\, a \in Act \qquad (a.P, a.Q) \in R$
  - $\forall\, f : \Delta \to \Delta \qquad (f(P), f(Q)) \in R$
  - $\forall\, L \subseteq \Delta \qquad (P \setminus L, Q \setminus L) \in R$
- если $(P_1,Q_1) \in R$ и $(P_2,Q_2) \in R$, то
  - $(P_1 + P_2, Q_1 + Q_2) \in R$
  - $(P_1 \mid P_2,\ Q_1 \mid Q_2) \in R$
- если S и T – системы процессных уравнений вида

$$\begin{aligned} S = &\quad \{X_i = P_i \mid i = 1,\ldots,n\} \\ T = &\quad \{X_i = Q_i \mid i = 1,\ldots,n\} \end{aligned}$$

  где $\forall\, i \in \{1,\ldots,n\} \quad (P_i,Q_i) \in R$, то

$$\forall\, i \in \{1,\ldots,n\} \quad (S_i,T_i) \in R$$

Можно доказать, что

1. $\sim$ и $\stackrel{c}{=}$ являются конгруэнциями.
2. $\approx$ сохраняет все операции, за исключением $+$: например, если $a \neq \tau$, то

$$\mathbf{0} \approx \tau.\mathbf{0}, \ \text{ но } \ (\mathbf{0} + a.\mathbf{0}) \not\approx (\tau.\mathbf{0} + a.\mathbf{0})$$

3. Для любых ПВ P и Q
   - $P \stackrel{c}{=} Q \quad \Leftrightarrow \quad \forall\, R \in \mathcal{E} \quad P + R \approx Q + R$
   - $P \approx Q \quad \Leftrightarrow \quad \begin{cases} P \stackrel{c}{=} Q & \text{или} \\ P \stackrel{c}{=} \tau.Q & \text{или} \\ \tau.P \stackrel{c}{=} Q \end{cases}$
4. каждая конгруэнция θ на $\mathcal{E}$ обладает свойством

$$\theta \subseteq \approx \quad \Leftrightarrow \quad \theta \subseteq \stackrel{c}{=}$$

7.6 Процессная алгебра

Процессной алгеброй называется алгебра, носителем которой является фактор–множество множества $\mathcal{E}$ по отношению наблюдаемой конгруэнции, а операции индуцированы соответствующими операциями на процессных выражениях.

В излагаемых ниже утверждениях мы будем использовать следующие обозначения:

- для каждого подмножества $L \subseteq \Delta \cup \overline{\Delta}$

$$\begin{aligned} names(L) &\stackrel{\text{def}}{=} \{name(\lambda) \mid \lambda \in L\} \\ \overline{L} &\stackrel{\text{def}}{=} \{\overline{\lambda} \mid \lambda \in L\} \end{aligned}$$

  где для каждого $a \in \Delta \quad \overline{\overline{a}} \stackrel{\text{def}}{=} a$

- для каждого ПВ P знакосочетание $L(P)$ обозначает множество всех действий $a \neq \tau$, имеющих хотя бы одно **свободное вхождение** в P, т.е. такое вхождение, которое не находится в области действия никакой операции блокировки вида $\setminus L$, где $name(a) \in L$.

Ниже символы P, Q, R обозначают элементы процессной алгебры, f и g – функции из Δ в Δ, L и M – подмножества Δ, a – элемент Act.

Можно доказать, что в процессной алгебре имеют место следующие соотношения.

1. (a) $(P + Q) + R = P + (Q + R)$
   (b) $P + Q = Q + P$
   (c) $P + \mathbf{0} = P$
   (d) $P + P = P$
   (e) $(P \mid Q) \mid R = P \mid (Q \mid R)$
   (f) $P \mid Q = Q \mid P$
   (g) $P \mid \mathbf{0} = P$

2. (a) $a.\tau.P = a.P$
   (b) $P + \tau.P = \tau.P$
   (c) $a.(P + \tau.Q) + a.Q = a.(P + \tau.Q)$
   (d) $P + \tau.(P + Q) = \tau.(P + Q)$

3. (a) $(P + Q) \setminus L = (P \setminus L) + (Q \setminus L)$
   (b) $(a.P) \setminus L =$
   $$= \begin{cases} a.(P \setminus L), & \text{если } a = \tau \text{ или } name(a) \notin L \\ \mathbf{0}, & \text{если } name(a) \in L \end{cases}$$
   (c) $(P \setminus L) \setminus M = P \setminus (L \cup M)$
   (d) $(P \mid Q) \setminus L = (P \setminus L) \mid (Q \setminus L)$, если
   $$L \cap names(L(P) \cap \overline{L(Q)}) = \emptyset$$
   (e) $P \setminus L = P$, если $L \cap names(L(P)) = \emptyset$
   (f) $\mathbf{0} \setminus L = \mathbf{0}$

4. (a) $f(P + Q) = f(P) + f(Q)$
   (b) $f(a.P) = f(a).f(P)$
   (c) $id(P) = P$ (где $id : \Delta \to \Delta$ – тождественная функция)
   (d) если для каждого $a \in L(P)$ $f(a) = g(a)$, то $f(P) = g(P)$
   (e) $f(g(P)) = (f \circ g)(P)$
   (f) $f(P \mid Q) = f(P) \mid f(Q)$, если для всех $a, b \in names(L(P) \cup L(Q))$
   $$a \neq b \quad \Rightarrow \quad f(a) \neq f(b)$$
   (g) $f(\mathbf{0}) = \mathbf{0}$
   (h) $f(P \setminus L) = f(P) \setminus f(L)$
   (i) $f(P) \setminus L = f(P \setminus f^{-1}(L))$

5. Для каждой системы (7.1) и каждого $i \in \{1, \ldots, n\}$
   $$S_i = P_i[\vec{X} := \vec{S}]$$
   где $\vec{X}$ и $\vec{S}$ имеют тот же смысл, что и в пункте (7.4).

6. **Теорема о разложении:**
   Пусть заданы
   (a) список ПВ $P_1, \ldots, P_n$, причём $\forall\, i \in \{1, \ldots, n\}$ совокупность всех выходящих из P_i рёбер имеет вид
   $$\{\ P_i \xrightarrow{a_{ij}} P'_{ij} \ \mid j \in I_i\}$$
   (b) функции $f_1, \ldots, f_n$ из Δ в Δ, и
   (c) подмножество $L \subseteq \Delta$.

   Можно доказать, что элемент процессной алгебры
   $$(f_1(P_1) \mid \ldots \mid f_n(P_n)) \setminus L$$
   равен сумме всех элементов вида
   $$f_i(a_{ij}).(\left\{ \begin{array}{l} f_1(P_1) \mid \ldots \mid f_{i-1}(P_{i-1}) \\ f_i(P'_{ij}) \\ f_{i+1}(P_{i+1}) \mid \ldots \mid f_n(P_n) \end{array} \right\} \setminus L)$$
   где $i \in \{1, \ldots, n\}$, $j \in I_i$, $name(f_i(a_{ij})) \notin L$, и фигурные скобки изображают операцию параллельной композиции, а также всех элементов вида
   $$\tau.(\left\{ \begin{array}{l} f_1(P_1) \mid \ldots \mid f_{i-1}(P_{i-1}) \\ f_i(P'_{ik}) \\ f_{i+1}(P_{i+1}) \mid \ldots \mid f_{j-1}(P_{j-1}) \\ f_j(P'_{jl}) \\ f_{j+1}(P_{j+1}) \mid \ldots \mid f_n(P_n) \end{array} \right\} \setminus L)$$
   где $1 \leq i < j \leq n$, $k \in I_i$, $l \in I_j$, $f_i(a_{ik}) = \overline{f_j(a_{jl})}$.
   Фигурные скобки в этом и предыдущем выражениях изображают операцию параллельной композиции.

7.7 Пример верификации

Глава 8

Морфизмы СП

При анализе сложных систем их часто заменяют на более простые системы, с целью того, чтобы вместо анализа свойств исходной системы анализировать эти свойства на упрощённой системе.

Главное условие допустимости такой замены заключается в том, что исходная система должна обладать всеми теми свойствами, которыми обладает её упрощение.

Понятие **морфизма** систем предназначено для точного описания взаимосвязи между системой и её упрощением.

8.1 Понятие морфизма систем

Пусть заданы две системы

$$\begin{array}{rcl} S & = & (Q,\ \delta,\ L,\ Q^0,\ \mathcal{P}\,) \\ S' & = & (Q',\delta',L',Q'^0,\mathcal{P}') \end{array} \qquad (8.1)$$

причём $\mathcal{P} \supseteq \mathcal{P}'$.

Морфизм из S в S' – это произвольное бинарное отношение H вида

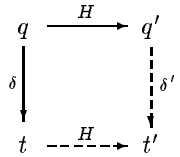
$$H \subseteq Q \times Q'$$

удовлетворяющее следующим условиям:

1. $\forall\, q \in Q^0\ \ \exists\, q' \in Q'^0 :\ \ (q,q') \in H$
2. если $(q,q') \in H$, то $\forall\, p \in \mathcal{P}' \quad q(p) = q'(p)$
3. если $(q,q') \in H$ и $q \xrightarrow{\delta} t$, то

$$\exists\, t' \in Q' : \quad (t,t') \in H \quad \text{и} \quad q' \xrightarrow{\delta'} t'$$

Последнее условие удобно представлять графически в виде диаграммы



Пути $\pi = (q_0, \ldots)$ в S и $\pi' = (q'_0, \ldots)$ в S' называются **соответствующими** относительно H, если

$$\forall i \geq 0 \quad (q_i, q'_i) \in H$$

Из определения морфизма непосредственно следует, что если $(q,q') \in H$, то для каждого пути π из q существует путь π' из q', соответствующий π относительно H.

Обозначим символос Φ_A совокупность формул, не содержащих квантора **E**. Очевидно, что $\mathrm{LTL} \subseteq \Phi_A$. Это включение является строгим.

Нетрудно доказать, что если существует морфизм из S в S', то для каждой формулы $\varphi \in \Phi_A$, такой, что $\mathcal{P}_\varphi \subseteq \mathcal{P}'$, имеет место импликация

$$S' \models \varphi \quad \Rightarrow \quad S \models \varphi$$

Система $S = (Q, \delta, L, Q^0)$ называется **детерминированной**, если

- $|Q^0| = 1$, и
- для каждого состояния $q \in Q$, и для всех $q', q'' \in \delta(q)$ имеет место импликация

$$q' \neq q'' \quad \Rightarrow \quad L(q') \neq L(q'')$$

Если системы S и S' – детерминированные, то

$$S \to S' \quad \Leftrightarrow \quad L(S) \subseteq L(S')$$

где $L(S) \stackrel{\text{def}}{=} \{L(\pi) \mid \pi - \text{ из } q_0\}$

8.2 Наибольший морфизм

Пусть заданы системы S и S' вида (8.1).

Определим последовательность отношений

$$\{H_i \subseteq Q \times Q' \mid i \geq 0\} \qquad (8.2)$$

следующим образом:

- $(q,q') \in H_0 \quad \Leftrightarrow \quad \forall\, p \in \mathcal{P}' \quad q(p) = L'_p(q')$
- $(q,q') \in H_{n+1} \quad \Leftrightarrow \quad \left\{ \begin{array}{l} (q,q') \in H_n \\ \text{если } q \xrightarrow{\delta} t, \text{ то } \exists\, t' \in Q' : \\ \quad (t,t') \in H_n \quad \text{и} \quad q' \xrightarrow{\delta'} t' \end{array} \right.$

Обозначим символом H пересечение отношений из (8.2)

$$H \stackrel{\text{def}}{=} \bigcap_{i \geq 0} H_i \tag{8.3}$$

Из конечности Q и невозрастания последовательности (8.2) следует, что $H = H_{i_0}$ для некоторого номера i_0.

Можно доказать, что

1. существование морфизма из S в S' эквивалентно условию

$$\forall\, q \in Q^0 \;\; \exists\, q' \in Q'^0 : \;\; (q, q') \in H \tag{8.4}$$

2. если имеет место (8.4), то отношение (8.3) является наибольшим (по включению) морфизмом из S в S'.

Истинность второго утверждения вытекает из того, что для каждого морфизма H' из S в S' имеют место соотношения

$$\begin{array}{l} H' \subseteq H_0 \\ \forall i \geq 0 \qquad H' \subseteq H_i \;\Rightarrow\; H' \subseteq H_{i+1} \end{array}$$

т.е. для каждого $i \geq 0 \quad H' \subseteq H_i$, откуда следует, что

$$H' \subseteq \bigcap_{i \geq 0} H_i = H$$

8.3 Fair морфизмы

Если системы (8.1) содержат компоненты fairness, то можно определить понятие **fair морфизма** из S в S', которое отличается от понятия обычного морфизма только условием 3.

Условие 3 для fair морфизма выглядит так: если $(q, q') \in H$, то для каждого fair пути $\pi = (q_0, \ldots)$ из q существует fair путь $\pi' = (q'_0, \ldots)$ из q', такой, что

$$\forall i \geq 0 \quad (q_i, q'_i) \in H$$

Можно доказать, что если существует fair морфизм из S в S', то

$$\forall \varphi \in \Phi_A \qquad (S' \models_F \varphi \quad \Rightarrow \quad S \models_F \varphi)$$

8.4 Изоморфизм систем

Пусть заданы две системы вида

$$\begin{array}{rcl} S & = & (Q,\ \delta,\ L,\ Q^0, \mathcal{P}) \\ S' & = & (Q', \delta', L', Q'^0, \mathcal{P}) \end{array} \tag{8.5}$$

причём множества утверждений у S и S' совпадают.

Изоморфизм между S и S' – это произвольное бинарное отношение $B \subseteq Q \times Q'$, такое, что

- B является морфизмом из S в S', и
- B^{-1} является морфизмом из S' в S.

Пути $\pi = (q_0, \ldots)$ в S и $\pi' = (q'_0, \ldots)$ в S' называются **соответствующими** относительно B, если

$$\forall i \geq 0 \quad (q_i, q'_i) \in B$$

Нетрудно доказать, что если существует изоморфизм B между S и S', то

- если путь π' соответствует пути π относительно B, то

$$\forall \varphi \in \Phi \qquad \pi(\varphi) = \pi'(\varphi)$$

- если $(q, q') \in B$, то

$$\forall \varphi \in \Phi_s \qquad q(\varphi) = q'(\varphi)$$

Если системы (8.5) – детерминированные, то

$$S \sim S' \quad \Leftrightarrow \quad L(S) = L(S')$$

Теорема.
Следующие утверждения эквивалентны.

1. $S \sim S'$
2. $\forall \varphi \in \Phi_s \quad S(\varphi) = S'(\varphi)$
3. $\forall \varphi \in \mathrm{CTL} \quad S(\varphi) = S'(\varphi)$

8.5 Наибольший изоморфизм

Пусть заданы системы S и S' вида (8.5).

Определим последовательность отношений

$$\{B_i \subseteq Q \times Q' \mid i \geq 0\} \tag{8.6}$$

следующим образом:

- $(q, q') \in B_0 \quad \Leftrightarrow \quad \forall\, p \in \mathcal{P} \quad q(p) = q'(p)$
- $(q, q') \in B_{n+1} \quad \Leftrightarrow \quad \left\{ \begin{array}{l} (q, q') \in B_n \\ \text{если } q \stackrel{\delta}{\to} t, \text{ то } \exists\, t' \in Q' : \\ \quad (t, t') \in B_n \quad \text{и} \quad q' \stackrel{\delta'}{\to} t' \\ \text{если } q' \stackrel{\delta}{\to} t', \text{ то } \exists\, t \in Q : \\ \quad (t, t') \in B_n \quad \text{и} \quad q \stackrel{\delta}{\to} t \end{array} \right.$

Обозначим символом B пересечение отношений из (8.6)

$$B \stackrel{\text{def}}{=} \bigcap_{i \geq 0} B_i \tag{8.7}$$

Из конечности Q и невозрастания последовательности (8.6) следует, что $B = B_{i_0}$ для некоторого номера i_0.

Можно доказать, что

1. существование изоморфизма между S и S' эквивалентно условию

$$\begin{array}{l} \forall\, q \in Q^0 \;\; \exists\, q' \in Q'^0 : \;\; (q, q') \in B \\ \forall\, q' \in Q'^0 \;\; \exists\, q \in Q^0 : \;\; (q, q') \in B \end{array} \tag{8.8}$$

2. если имеет место (8.8), то отношение (8.7) является наибольшим (по включению) изоморфизмом между S и S'.

8.6 Fair изоморфизмы

Изоморфизм B между системами (8.5) называется fair, если

- B является fair морфизмом из S в S', и
- B^{-1} является fair морфизмом из S' в S.

Можно доказать, что если fair системы S и S' – детерминированные, то следующие утверждения эквивалентны:

- существует fair изоморфизм между S и S'
- $L_F(S) = L_F(S')$

Отметим, что проблема проверки равенства

$$L_F(S) = L_F(S')$$

является PSPACE–полной.

Можно доказать, что если если существует fair изоморфизм между S и S', то для каждой формулы φ

$$S \models_F \varphi \quad \Leftrightarrow \quad S' \models_F \varphi$$

8.7 Канонические модели ACTL-формул

8.7.1 Определение ACTL

Символ ACTL обозначает множество всех CTL–формул,

- не содержащих квантора $\mathbf{E}$, и
- все отрицания в которых располагаются только над утверждениями.

Нетрудно доказать, что

- CTL–формулы $\mathbf{AGEF}p$ и $\mathbf{AG}\overline{\mathbf{AF}p}$ не эквивалентны ни одной ACTL–формуле
- ACTL–формулы $\mathbf{AFAG}p$ и $\mathbf{AGAX}p$ не эквивалентны ни одной LTL–формуле.

8.7.2 Замыкание ACTL-формулы

Замыканием формулы $\varphi \in$ ACTL называется наименьшее (по отношению включения) множество $\langle\varphi\rangle$ формул, которое мы будем обозначать тем же символом $\langle\varphi\rangle$, что и замыкание, и которое удовлетворяет следующим условиям:

1. каждая state–подформула формулы φ принадлежит $\langle\varphi\rangle$
2. если формула ψ вида $\mathbf{A}(\psi\mathbf{U}\eta)$ или $\mathbf{A}(\psi\mathbf{R}\eta)$ принадлежит $\langle\varphi\rangle$, то $\mathbf{AX}\psi$ принадлежит $\langle\varphi\rangle$
3. формула $\mathbf{AX0}$ принадлежит $\langle\varphi\rangle$

Подмножество $\langle\varphi\rangle_X \subseteq \langle\varphi\rangle$ состоит из

- всех утверждений, входящих в φ, и
- всех формул вида $\mathbf{AX}\psi$, входящих в $\langle\varphi\rangle$

8.7.3 Система S_φ

1. Состояниями системы S_φ являются все функции вида
$$K : \langle\varphi\rangle_X \to \{0,1\}$$
удовлетворяющие следующим условиям:
   (a) $K(\overline{\psi}) = \overline{K(\psi)}$
   (b) $K(\psi \vee \eta) \stackrel{\text{def}}{=} K(\psi) \vee K(\eta)$
   (c) если $\psi = \mathbf{A}(\mathbf{U}(\psi,\eta))$, то
$$K(\psi) = \left[\begin{array}{l} K(\eta) \\ K(\psi) \wedge K(\mathbf{AX}\psi) \end{array} \right] \vee K(\mathbf{AX0})$$
   (d) если $\psi = \mathbf{A}(\psi\mathbf{R}\eta)$, то
$$K(\psi) = \left\{ \begin{array}{l} K(\eta) \\ K(\psi) \vee K(\mathbf{AX}\psi) \end{array} \right\} \vee K(\mathbf{AX0})$$
2. Отношение перехода:
$$K \to K' \quad \stackrel{\text{def}}{=} \bigwedge_{\mathrm{AX}\psi \in \langle\varphi\rangle_X} K(\mathbf{AX}\psi) \to K'(\psi)$$
3. Оценка утверждений
$$L(K,p) \stackrel{\text{def}}{=} K(p)$$
4. Начальные состояния
$$Q^0(K) \stackrel{\text{def}}{=} K(\varphi)$$
5. Условия fairness
$$F = \{F_{\mathrm{A}(\psi\mathrm{U}\eta)} \mid \mathbf{A}(\psi\mathbf{U}\eta) \in \langle\varphi\rangle\}$$
где
$$F_{\mathrm{A}(\psi\mathrm{U}\eta)}(K) \quad \stackrel{\text{def}}{=} \quad K(\mathbf{AX}(\mathbf{A}(\psi\mathbf{U}\eta))) \to K(\eta)$$

Можно доказать, что для каждой подформулы ψ формулы φ
$$K(\psi) = (K \models_F \psi)$$

8.7.4 Теорема

Для каждой системы S следующие условия эквивалентны:

1. $S \models_F \varphi$
2. существует fair морфизм из S в S_φ

Доказательство.

Если верно (1), то морфизм $H : S \to S_\varphi$ можно определить следующим образом:

$$H(q,K) \stackrel{\text{def}}{=} \bigwedge_{\psi \in \langle\varphi\rangle_X} q(\psi) \leftrightarrow K(\psi)$$

Если верно (2), то (1) следует из того, что $S_\varphi \models \varphi$.

8.8 Абстракция

Глава 9

Редукция СП

в том числе симметрия

Глава 10

Дедуктивные рассуждения и построение инвариантов

блок-схемы, в том числе параллельные
статья с Жуковым

Глава 11

Функциональные и логические программы

Библиография

[1] **International Standard ISO/IEC 9126.**
Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for their Use. *International Organization for Standardization, International Electrotechnical Commission, Geneva, 1991.*

[2] **E. Clarke, O. Grumberg, D. Peled:** Model checking. *MIT Press, 2001.*

[3] **R.W.Floyd:** Assigning meanings to programs, *Proc. Symp. Appl. Math.,* **19***; in: J.T.Schwartz (ed.), Mathematical Aspects of Computer Science, pp. 19-32, American Mathematical Society, Providence, R.I., 1967.*

[4] **R.Milner:** A Calculus of Communicating Systems, *Lecture Notes in Computer Science, vol. 92, Springer, 1980.*

[5] **J.Esparza:** Decidability of model-checking for infinite-state concurrent systems, *Acta Informatica, 34:85-107, 1997.*

[6] **P.A.Abdulla, A.Annichini, S.Bensalem, A.Bouajjani, P.Habermehl, Y.Lakhnech:** Verification of Infinite-state Systems by Combining Abstraction and Reachability Analysis, *Lecture Notes in Computer Science 1633, pages 146-159, Springer-Verlag, 1999.*

[7] **K.L.McMillan**: Verification of Infinite state Systems by Compositional Model Checking, *Conference on Correct Hardware Design and Verification Methods, pages 219-234, 1999.*

[8] **O.Burkart, D.Caucal, F.Moller, and B.Steffen:** Verification on infinite structures, *In J. Bergstra, A. Ponse and S. Smolka, editors, Handbook of Process Algebra, chapter 9, pages 545-623, Elsevier Science, 2001.*