Верификация программ методом model checking

А.М.Миронов

Оглавление

Глава 1

Введение

1.1 Проблема качества программных систем

Современный этап развития индустрии программных систем характеризуется значительным усложнением процесса их разработки. Однако используемые в настоящее время методы контроля качества разрабатываемых систем характеризуются неполнотой, высокой сложностью, и недостаточной надёжностью. Данная ситуация приводит к увеличению числа ошибок при разработке современных программных систем.

Для разработки сложных программных систем, отвечающих современным требованиям к их качеству, необходимы

- создание адекватного математического фундамента, предназначенного для моделирования современных программных систем, и
- разработка на основе этого фундамента эффективных алгоритмов анализа различных свойств программных систем.

Требования к качеству современных программных 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 структуры.

6. Чёткая и понятная **документированность** 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** и **совместимость**, т.е. способность системы корректно работать на разных платформах и в разных конфигурациях.

1.2 Тестирование

Как правило, анализ соответствия системы предъявляемым к ней требованиям производится либо путём её визуального анализа, либо методом **тестирования**. Тестирование системы заключается в анализе её поведения на некоторых входных данных. Тестирование является в настоящее время основной формой контроля качества разрабатываемых программных систем, и занимает примерно две трети общего времени, затрачиваемого на их разработку.

Тестирование обладает 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 проверяемыми свойствами. Как отметил Дейкстра ([2], стр. 41), «тестирование может лишь помочь выявить некоторые ошибки, но не доказать их отсутствие».

Во многих программных системах наличие даже незначительных ошибок категорически недопустимо. Например, наличие даже небольших ошибок в таких программных 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 большому ущербу для экономики и жизни людей.

1.3 Верификация

1.3.1 Понятие верификации

Гарантированное обоснование качества программных систем может быть получено только на основе альтернативного подхода к анализу качества программных систем, принципиально отличного от тестирования. Данный подход называется **верификацией**.

Верификация программной системы состоит из следующих частей.

1. Построение **математической модели** анализируемой системы.
2. Представление проверяемых свойств анализируемой системы в виде формального текста (например, в виде формулы математической логики), называемого **спецификацией**.
3. Построение **математического доказательства** наличия или отсутствия у системы проверяемых свойств, выраженных в спецификации.

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

Рассмотрим отдельно каждую из составных частей верификации.

1.3.2 Математические модели систем

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

- вершины которого называются **состояниями**, и изображают ситуации (или классы ситуаций), в которых может находиться эта система в различные моменты времени, и
- рёбра которого соответствуют переходам, по которым могут происходить изменения состояний в процессе функционирования этой системы.

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 программная система может быть представлена различными моделями, отражающими разные степени абстракции, и разные уровни детализации действий, исполняемых этой системой.

При построении модели программной системы следует руководствоваться следующими принципами:

- модель системы не должна быть чрезмерно детальной, т.к. излишняя сложность модели может вызвать существенные вычислительные проблемы при её формальном анализе,

- 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.

1.3.3 Спецификация

Спецификация программной системы – это описание анализируемых свойств этой системы в виде формального текста, выраженное например в виде формулы математической логики.

Если свойство программной системы изначально было выражено на естественном языке, то при переводе его в спецификацию важно обеспечить соответствие между естественно-языковым описанием этого свойства и его спецификацией, т.к. в случае несоблюдения этого условия результаты верификации не будут иметь смысла.

Одним из инструментов формального описания свойств программных систем является **темпоральная логика**. Свойства систем выражаются в темпоральной логике **темпоральными формулами**.

Примеры свойств, выражаемых темпоральными формулами:

- система при любом варианте своего функционирования не будет находиться ни в одном из состояний из заданного класса,

- система при некотором функционировании когда-нибудь попадёт в некоторое состояние из заданного класса.

Как правило, при проведении рассуждений, связанных с анализом свойств систем, описываемых темпоральными формулами, рассматриваются не всевозможные темпоральные формулы, а только темпоральные формулы из некоторого ограниченного класса. Классы темпоральных

формул принято называть **темпоральными логиками**, т.е. словосочетание «темпоральная логика» имеет два значения: в первом значении – это язык, на котором можно выражать спецификации в виде темпоральных формул, а во втором – некоторый класс темпоральных формул.

В этом тексте будут рассмотрены следующие темпоральные логики: CTL (Computational Tree Logic), LTL (Linear Temporal Logic), PCTL (Probabilistic Computational Tree Logic).

1.3.4 Построение формальных доказательств

Существует два основных подхода к построению формальных доказательств утверждений о том, что модель программной системы удовлетворяет или не удовлетворяет своей спецификации:

- **логический вывод**, т.е. автоматизированное построение доказательства теоремы о том, что анализируемая модель удовлетворяет проверяемой спецификации, и
- **model checking**, одна из главных задач которого заключается в том, чтобы попытаться найти вариант функционирования анализируемой модели, при котором нарушается свойство, выражаемое проверяемой спецификацией.

Поскольку заранее неизвестно, удовлетворяет ли анализируемая модель спецификации или нет, то для ее анализа следует использовать оба метода: методом построения логического вывода пытаться доказать, что модель удовлетворяет спецификации, и одновременно методом model checking пытаться построить опровержение данного утверждения.

В настоящем тексте подробно рассматривается метод model checking верификации программных систем. Главным достоинством этого метода является возможность полностью автоматического анализа модели верифицируемой системы. Наиболее существенным недостатком model checking является высокая вычислительная сложность процедуры верификации на основе этого метода.

Среди учебной литературы по model checking в первую очередь следует назвать замечательную книгу Ю.Г.Карпова [3], и также [4] и [5].

Наиболее известной промышленной системой верификации, основанной на методе model checking, является система SPIN [6]. Среди средств автоматического доказательства теорем следует отметить такие инструменты как Coq [7], Isabelle [8], PVS [9].

Глава 2

Модели программных систем

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

2.1 Основные понятия

2.1.1 Термы

Мы предполагаем, что заданы множества $\mathcal{T}$, $\mathcal{D}$, $\mathcal{X}$, Con, Fun, элементы которых называются **типами**, **значениями**, **переменными**, **константами**, и **функциональными символами (ФС)**, соответственно. Каждому элементу x множеств $\mathcal{D}$, $\mathcal{X}$, Con, Fun сопоставлен некоторый тип из $\mathcal{T}$, обозначаемым записью $\tau(x)$, причем если $x \in Fun$, то $\tau(x)$ – это запись вида $(t_1, \ldots, t_n) \to t$, где $t_1, \ldots, t_n, t \in \mathcal{T}$.

$\forall\, t \in \mathcal{T}$ запись $\mathcal{D}_t$ обозначает множество $\{d \in \mathcal{D} \mid \tau(d) = t\}$.

С каждой константой $c \in Con$ связано некоторый элемент множества $\mathcal{D}_{\tau(c)}$, называемый **значением** этой константы.

С каждым ФС $f \in Fun$ связана функция, обозначаемая тем же символом f, и имеющая вид

$$f : \mathcal{D}_{t_1} \times \ldots \times \mathcal{D}_{t_n} \to \mathcal{D}_t, \quad \text{где } \tau(f) = (t_1, \ldots, t_n) \to t.$$

$\mathcal{T}$ содержит тип $\mathbf{B}$ (называемый **булевским** типом), $\mathcal{D}_{\mathbf{B}} = \{1, 0\}$, множество Con содержит константы $\top$ и $\bot$ типа $\mathbf{B}$, с которыми связаны значения 1 и 0, соответственно.

Fun содержит **булевы ФС**: $\neg$, $\wedge$, $\vee$, где

$$\tau(\neg) = \mathbf{B} \to \mathbf{B},\ \tau(\wedge) = \tau(\vee) = (\mathbf{B}, \mathbf{B}) \to \mathbf{B}.$$

Функции, соответствующие этим ФС, являются стандартными булевыми функциями на аргументах 1 и 0 (т.е. $\neg(1) = 0$, и т.д.).

Понятие **терма** определяется индуктивно. Каждый терм e связан с некоторым типом $\tau(e) \in \mathcal{T}$. Определение терма имеет следующий вид:

- каждая переменная и каждая константа является термом, тип которого равен типу этой переменной или константы,
- если f – ФС, $e_1, \ldots, e_n$ термы, и
$$\tau(f) = (\tau(e_1), \ldots, \tau(e_n)) \to t,$$
то запись $f(e_1, \ldots, e_n)$ – терм типа t.

Будем использовать следующие понятия и обозначения.

- Термы типа $\mathbf{B}$ называются **формулами**.
- Множества всех термов и всех формул будем обозначать символами Tm и Fm соответственно.
- $\forall\, e \in Tm$ X_e обозначает множество переменных, входящих в e.
- $\forall\, X \subseteq \mathcal{X}$ записи Tm_X и Fm_X обозначают множества $\{e \in Tm \mid X_e \subseteq X\}$ и $Tm_X \cap Fm$ соответственно.
- Термы, содержащие булевы ФС, будут обозначаться так же как в математических текстах (т.е. в виде записей $e \wedge e'$, и т.д.), термы вида $e_1 \wedge \ldots \wedge e_k$ могут также обозначаться записями вида $\left\{ \begin{array}{c} e_1 \\ \ldots \\ e_k \end{array} \right\}$, или $\bigwedge_{i=1}^{k} e_i$, или $\bigwedge_{i \in \{1,\ldots,k\}} e_i$ (аналогичные обозначения для термов с $\vee$). Термы вида $\neg e$ будут обозначаться записями вида $\bar{e}$.
- $\forall\, \beta, \beta' \in Fm$ запись $\beta \to \beta'$ является сокращением формулы $\bar{\beta} \vee \beta'$.

2.1.2 Означивания

Пусть задано множество переменных $X \subseteq \mathcal{X}$. **Означиванием** переменных из X называется функция $\xi : X \to \mathcal{D}$, сопоставляющая каждой переменной $x \in X$ значение $\xi(x)$ типа $\tau(x)$. Множество всех означиваний переменных из X обозначается записью $X^\bullet$.

$\forall\, \xi \in X^\bullet$, $\forall\, e \in Tm_X$ e^ξ обозначает значение, определяемое рекурсивно:

- если $e = x \in X$, то $e^\xi = \xi(x)$,
- если $e = c \in Con$, то e^ξ – значение константы c,
- если $e = f(e_1, \ldots, e_n)$, где $f \in Fun$, то $e^\xi = f(e_1^\xi, \ldots, e_n^\xi)$.

2.2 Программы и программные системы

2.2.1 Понятие программы

Программа представляет собой конечный ориентированный граф, каждая вершина которого имеет один из следующих видов:

- **начальная** вершина, она обозначается символом $\odot$, в каждой программе имеется только одна начальная вершина, из неё выходит одно ребро, и в неё не входит ни одного ребра,
- **присваивание**, вершина такого вида обозначается записью вида

$$\boxed{x := e}$$

  где $x \in \mathcal{X}$, $e \in Tm$, $\tau(x) = \tau(e)$, из каждого присваивания выходит только одно ребро,
- **условный переход**, вершина такого вида обозначается записью вида $\textcircled{\beta}$, где $\beta \in Fm$, из каждого условного перехода выходит два ребра с метками 1 и 0 соответственно,
- **пустой оператор**, такая вершина обозначается символом $\bigcirc$, из неё выходит только одно ребро,
- **терминальная** вершина, она обозначается символом $\otimes$, из каждой терминальной вершины не выходит ни одного ребра.

С каждой программой P связана некоторая формула Pre_P, называемая **предусловием** программы P.

Если P – программа, то V_P обозначает совокупность всех вершин P, и X_P обозначает множество всех переменных, входящих в P.

2.2.2 Выполнение программы

Выполнение программы P – это обход вершин программы P, начиная с начальной вершины (т.е. последовательность переходов по рёбрам от одной вершины программы P к другой), с выполнением действий, сопоставленных проходимым вершинам. После выполнения действия, соответствующего текущей вершине, происходит переход по выходящему из неё ребру к следующей вершине. На каждом шаге выполнения программы P каждая переменная из X_P содержит некоторое значение. Значения переменных в начальный момент должны удовлетворять предусловию. Действия в вершинах выполняются в зависимости от вида вершин:

- $\odot$ или $\bigcirc$: никаких действий не происходит,
- $\boxed{x := e}$: в переменную x заносится значение терма e,
- $\textcircled{\beta}$: для перехода к следующей вершине выбирается ребро с меткой, равной значению формулы β,
- $\otimes$: выполнение программы завершается.

Формальное описание выполнения программы имеет следующий вид: выполнение P представляет собой последовательность шагов с номерами $0, 1, \ldots$, с каждым шагом $i \geq 0$ выполнения P связаны вершина $v_i \in V_P$ и означивание $\xi_i \in X_P^{\bullet}$ (называемые **текущей вершиной** и **текущим означиванием** на шаге i), причем $v_0 = \odot$, $Pre_P{}^{\xi_0} = 1$, и $\forall\, i \geq 0$

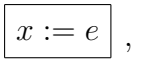
- если $v_i = \otimes$, то выполнение программы завершается, и
- если $v_i \neq \otimes$, то определен $i+1$-й шаг выполнения P, компонентами которого являются текущая вершина v_{i+1} и текущее означивание ξ_{i+1} на этом шаге, а также ребро γ из v_i в v_{i+1}, причем
  - если v_i – начальная вершина или пустой оператор, то $\xi_{i+1} = \xi_i$,
  - если $v_i = \boxed{x := e}$, то $x^{\xi_{i+1}} \stackrel{\text{def}}{=} e^{\xi_i}$, $\forall\, y \in X_P \setminus \{x\} \quad y^{\xi_{i+1}} \stackrel{\text{def}}{=} y^{\xi_i}$,
  - если $v_i = \textcircled{\beta}$, то $\xi_{i+1} = \xi_i$, и ребро γ имеет метку β^{ξ_i}.

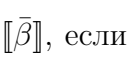
2.2.3 Процессные модели программ

Пусть задана программа P. **Процессная модель** программы P представляет собой граф, обозначаемый тем же символом P, каждое ребро которого имеет метку, называемую **действием**.

Процессная модель программы P строится следующим образом.

- На каждом ребре программы P рисуется точка. Нарисованные точки являются вершинами процессной модели. Обозначим множество всех нарисованных точек записью U_P. Точка, нарисованная на ребре, выходящем из $\odot$, обозначается записью P^0 и называется **начальным состоянием** процессной модели P.
- Пусть $v \in V_P$, и $u, u' \in U_P$ – точки на ребре γ, входящем в v и на ребре $\gamma\,'$, выходящем из v, соответственно. Тогда процессная модель программы P содержит ребро из u в u', метка которого имеет вид

– $x := e$, если $v =$  ,

– $[\![\beta]\!]$ или $[\![\bar{\beta}]\!]$, если $v =$  и γ' помечено символом 1 или 0, соответственно,

– $[\![\top]\!]$, если $v = \bigcirc$.

- Пусть $u \in U_P$ – точка на ребре, входящем в $\bigotimes$. Тогда процессная модель программы P содержит ребро из u в u с меткой $[\![\top]\!]$.

Каждому выполнению программы P соответствует некоторый обход вершин её процессной модели, начиная с P^0, с выполнением действий, являющихся метками проходимых рёбер.

2.2.4 Программные системы

Программной системой называется список программ: $(P_1, \ldots, P_k)$.

Выполнение программной системы $(P_1, \ldots, P_k)$ заключается в выполнении входящих в неё программ следующим образом: в каждый такт времени недетерминированно выбирается номер $i \in \{1, \ldots, k\}$, и выполняется очередное действие в программе P_i, все остальные программы в этом такте времени приостанавливают свою работу.

Данный вид выполнения программ, входящих в $(P_1, \ldots, P_k)$, называется **параллельным выполнением в режиме чередования**.

2.3 Системы переходов

Ниже предполагается, что задано множество $\mathcal{P}$, элементы которого называются **атомарными утверждениями** (**atomic propositions**).

2.3.1 Понятие системы переходов

Системой переходов (СП) называется пятёрка $\Sigma = (S, R, L, S^0, \mathcal{F})$, компоненты которой имеют следующий смысл:

- S – конечное множество **состояний**,
- $R \subseteq S \times S$ – **отношение перехода**,
- L – функция (называемая **оценкой**) вида $L : S \times \mathcal{P}_\Sigma \to \{1, 0\}$, где $\mathcal{P}_\Sigma \subseteq \mathcal{P}$,
- $S^0 \subseteq S$ – множество **начальных состояний**,

- $\mathcal{F}$ – некоторая совокупность подмножеств множества S, называемых **условиями fairness**.

Оценка L имеет следующий смысл: $\forall\, s \in S,\ \forall\, p \in \mathcal{P}_\Sigma$ утверждение p считается **истинным** в s, если $L(s,p) = 1$, и **ложным** в s, иначе.

Выражение $L(s,p)$ будем записывать более компактно в виде $s(p)$.

СП $\Sigma = (S, R, L, S^0, \mathcal{F})$ удобно рассматривать как граф (обозначаемый тем же символом Σ) с множеством вершин S. $\forall\,(s,s') \in R$ данный граф содержит ребро из s в s'.

Будем использовать следующие обозначения и соглашения:

- $\forall\, s \in S\ \ R(s) = \{s' \in S \mid (s,s') \in R\},\ \ R^{-1}(s) = \{s' \in S \mid (s',s) \in R\}$,
- если Σ – СП, компоненты которой не указаны, то будем обозначать её компоненты записями S_Σ, R_Σ, L_Σ, S^0_Σ, $\mathcal{F}_\Sigma$, соответственно,
- если СП Σ задана в виде тройки (S, R, L), то $S^0_\Sigma = S$, $\mathcal{F}_\Sigma = \{S\}$, если СП Σ задана в виде четверки (S, R, L, S^0), то $\mathcal{F}_\Sigma = \{S\}$.

2.3.2 Пути в системах переходов

Путь в СП Σ – это последовательность $\pi = (s_0, s_1, \ldots)$ состояний из S_Σ, такая, что $\forall\, i \geq 0 \quad s_{i+1} \in R_\Sigma(s_i)$. Если данная последовательность бесконечна, то путь π называется **бесконечным**, иначе путь π называется **конечным**. Если путь π конечный и имеет вид $(s_0, \ldots, s_n)$, то будем говорить, что π – путь из s_0 в s_n. Этот путь называется **циклом**, если $s_n = s_0$, и **пустым**, если $n = 0$.

Если $\pi = (s_0, \ldots)$ и $S \subseteq S_\Sigma$, то $\pi \subseteq S$ означает, что $\forall\, i \geq 0\ s_i \in S$.

Если пути π и π' имеют вид $(s_0, \ldots, s_n)$ и $(s_n, s_{n+1}, \ldots)$ соответственно, то запись $\pi\pi'$ обозначает путь $(s_0, \ldots, s_n, s_{n+1}, \ldots)$, называемый **конкатенацией** π и π'.

Если π – цикл, то π^∞ обозначает бесконечную конкатенацию $\pi\pi\pi\ldots$

Если π имеет вид $(s_0, s_1, \ldots)$, то будем говорить, что π – **путь из** s_0.

По умолчанию, под путями будем подразумевать бесконечные пути, а если какой-либо из рассматриваемых путей является конечным, то это будет специально оговариваться.

Для каждого пути π в СП Σ запись $\inf(\pi)$ обозначает множество

$$\{s \in S_\Sigma \mid s \text{ имеет бесконечно много вхождений в } \pi\}.$$

Путь π в СП Σ называется **fair**, если $\forall\, F \in \mathcal{F}_\Sigma\ \ \inf(\pi) \cap F \neq \emptyset$.

$\forall\, s \in S_\Sigma\ \ \Pi_s$ обозначает совокупность всех fair путей из s в Σ, и Π_Σ обозначает совокупность всех fair путей в Σ.

Запись Π^0_Σ обозначает множество $\bigcup_{s \in S^0_\Sigma} \Pi_s$.

2.3.3 Построение системы переходов, соответствующей программной системе

Пусть задана программная система $\mathbf{P} = (P_1, \ldots, P_k)$. Считаем, что $\forall\, i = 1, \ldots, k$ программа P_i задана в виде процессной модели с начальным состоянием P_i^0, множеством рёбер $Edges_i$, и предусловием Pre_i.

Будем использовать следующие обозначения:

- $\forall\, i = 1, \ldots, k$ X_i – множество, состоящее из всех переменных, входящих в P_i, а также дополнительной переменной at_i, принимающей значения в множестве вершин процессной модели P_i,
- $X = \bigcup_{i=1}^{k} X_i$, $X' = \{x' \mid x \in X\} \subseteq \mathcal{X}$, где $X \cap X' = \emptyset$,
- $\forall\, (\xi, \xi') \in X^\bullet \times X^\bullet \quad \xi \sqcup \xi'$ – означивание из $(X \sqcup X')^\bullet$, определяемое следующим образом:

$$\forall\, x \in X \;\; (\xi \sqcup \xi')(x) = \xi(x), \;\; (\xi \sqcup \xi')(x') = \xi'(x'),$$

- $\forall\, Y \subseteq X$ запись $same(Y)$ обозначает формулу $\bigwedge_{x \in Y}(x' = x)$,
- $Edges = \bigcup_{i=1}^{k} Edges_i$,
- $\forall\, \gamma \in Edges$ запись R_γ обозначает формулу, определяемую следующим образом: пусть n и n' – начало и конец γ соответственно, тогда
  - если метка ребра γ имеет вид $x := e$, то

$$R_\gamma = (at_i = n) \wedge (at_i' = n') \wedge (x' = e) \wedge same(X \setminus \{x, at_i\}),$$

  - если метка ребра γ имеет вид $[\![\beta]\!]$, то

$$R_\gamma = (at_i = n) \wedge (at_i' = n') \wedge \beta \wedge same(X \setminus \{at_i\}),$$

- символ R обозначает формулу $\bigvee_{\gamma \in Edges} R_\gamma$,
- Pre обозначает формулу $\bigwedge_{i=1}^{k} \Big(Pre_i \wedge (at_i = P_i^0)\Big)$.

Заметим, что $\forall\, \xi, \xi' \in X^\bullet$

- $Pre^\xi = 1$ тогда и только тогда, когда ξ является одним из возможных означиваний переменных из X в начальный момент выполнения системы $\mathbf{P}$, и
- $R^{\xi \sqcup \xi'} = 1$ тогда и только тогда, когда верно утверждение:

– если перед выполнением какого-либо действия системы $\mathbf{P}$ каждая переменная $x \in X$ имела значение $\xi(x)$,

– то после выполнения этого действия каждая переменная $x \in X$ будет иметь значение $\xi'(x)$.

СП Σ, соответствущая системе $\mathbf{P}$, имеет следующие компоненты.

- $S_\Sigma \stackrel{\text{def}}{=} X^\bullet$, $S^0_\Sigma \stackrel{\text{def}}{=} \{\xi \in X^\bullet \mid Pre^\xi = 1\}$.
- $R_\Sigma \stackrel{\text{def}}{=} \{(\xi, \xi') \in X^\bullet \times X^\bullet \mid R^{\xi \sqcup \xi'} = 1\}$.

  Из определения R_Σ и приведенного выше замечания следует что каждому выполнению $\mathbf{P}$ соответствует некоторый путь в Σ.
- $\mathcal{P}_\Sigma \stackrel{\text{def}}{=} Fm_X$, $\forall\, \xi \in S_\Sigma$, $\forall\, \beta \in Fm_X \;\; L_\Sigma(\xi, \beta) \stackrel{\text{def}}{=} \beta^\xi$.
- Выбор множества $\mathcal{F}_\Sigma$ условий fairness индивидуален для каждой программной системы. На реальные выполнения системы $\mathbf{P}$ могут быть наложены дополнительные условия, и $\mathcal{F}_\Sigma$ должно быть выбрано с таким расчетом, чтобы множество Π^0_Σ не содержало путей, не соответствующих никакому реальному выполнению (т.е. не удовлетворяющему этим дополнительным условиям) системы $\mathbf{P}$.

  Такие дополнительные условия могут иметь следующий вид.

  – Не должно быть таких выполнений $\mathbf{P}$, в которых одна из входящих в неё программ не совершает никаких действий после некоторого момента времени.

  – Одна из программ в $\mathbf{P}$ (P_i) может обращаться с запросами к другой программе (P_j), и программа P_j может посылать программе P_i ответы на эти запросы.

    Не должно быть таких бесконечных выполнений $\mathbf{P}$, в которых

    * одно из действий представляет собой запрос от P_i к P_j, и
    * все последующие действия P_j не являются посылкой ответа от P_j к P_i на этот запрос.

  – Одна из программ в $\mathbf{P}$ может посылать сообщения другой программе, причём сообщения при пересылке могут пропадать.

    Не должно быть таких выполнений $\mathbf{P}$, в которых одна из программ в $\mathbf{P}$ бесконечно много раз посылает сообщения другой программе в $\mathbf{P}$, и все эти сообщения пропадают.

Глава 3

Model checking на основе темпоральной логики CTL

В этой главе изучается темпоральная логика CTL (Computational Tree Logic), используемая для формальной спецификации различных свойств программных систем.

Основными объектами всех темпоральных логик являются **темпоральные формулы** (называемые также просто **формулами**).

Будем считать, что для каждой из рассматриваемых в этом тексте темпоральных логик TL верно следующее:

- каждое утверждение из множества $\mathcal{P}$ атомарных утверждений, введённого в пункте 2.3, является формулой логики TL, и
- TL замкнута относительно **булевых комбинаций**, т.е. среди формул логики TL есть константы $\top$ и $\bot$, и если TL содержит формулы B и C, то TL также содержит формулы $\bar{B},\ B \wedge C,\ B \vee C$.

Запись вида $B \to C$, где $B, C \in$ TL, обозначает формулу $\bar{B} \vee C$.

Ниже будем использовать следующее обозначение: если A – произвольное утверждение (выраженное возможно на естественном языке), то запись $[\![A]\!]$ обозначает значение $\begin{cases} 1, & \text{если } A \text{ истинно, и} \\ 0, & \text{если } A \text{ ложно.} \end{cases}$

3.1 Темпоральная логика CTL

3.1.1 Формулы темпоральной логики CTL

Совокупность **формул** темпоральной логики CTL, называемых также **CTL–формулами**, обозначается записью Fm_{CTL}. Как было отмечено

выше, Fm_{CTL} содержит $\mathcal{P}$ и замкнуто относительно булевых комбинаций. Кроме того, если Fm_{CTL} содержит формулы B и C, то Fm_{CTL} содержит также формулы

$$\mathbf{AX}B,\ \ \mathbf{EX}B,\ \ \mathbf{AF}B,\ \ \mathbf{EF}B,\ \ \mathbf{AG}B,\ \ \mathbf{EG}B,\ \ \mathbf{AU}(B,C),\ \ \mathbf{EU}(B,C).$$

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

3.1.2 Значения CTL–формул

Пусть задана СП Σ.

$\forall\, s \in S_\Sigma$, $\forall\, B \in Fm_{CTL}$ **значение** $s(B) \in \{1, 0\}$ формулы B в состоянии s определяется индуктивно:

1. если $B = p \in \mathcal{P}$, то $s(B) = \begin{cases} L_\Sigma(s,p), \text{ если } p \in \mathcal{P}_\Sigma, \\ 0, \text{ если } p \notin \mathcal{P}_\Sigma, \end{cases}$

2. значения булевых комбинаций определяются стандартно:

$$\begin{array}{l} s(\top) = 1, \ \ s(\bot) = 0, \\ s(\bar{B}) = \overline{s(B)}, \ \ s(B \wedge C) = s(B) \wedge s(C), \text{ и т.д.} \end{array}$$

3. значения CTL–формул, начинающихся с темпорального оператора, определяются следующим образом:

   - $s(\mathbf{AX}B) = [\![\, \forall\, s' \in R(s) \ \ s'(B) = 1 \,]\!]$,
   - $s(\mathbf{EX}B) = [\![\, \exists\, s' \in R(s) : s'(B) = 1 \,]\!]$,
   - $s(\mathbf{AF}B) = [\![\, \forall\, \pi = (s_0, \ldots) \in \Pi_s\ \exists\, i \geq 0 : s_i(B) = 1 \,]\!]$,
   - $s(\mathbf{EF}B) = [\![\, \exists\, \pi = (s_0, \ldots) \in \Pi_s,\ \exists\, i \geq 0 : s_i(B) = 1 \,]\!]$,
   - $s(\mathbf{AG}B) = [\![\, \forall\, \pi = (s_0, \ldots) \in \Pi_s\ \forall\, i \geq 0 : s_i(B) = 1 \,]\!]$,
   - $s(\mathbf{EG}B) = [\![\, \exists\, \pi = (s_0, \ldots) \in \Pi_s :\ \forall\, i \geq 0 : s_i(B) = 1 \,]\!]$,
   - $s(\mathbf{AU}(B,C)) = [\![\, \forall\, \pi = (s_0, \ldots) \in \Pi_s\ \exists\, i \geq 0 : s_i(C) = 1, \forall\, j < i \quad s_j(B) = 1 \,]\!]$,
   - $s(\mathbf{EU}(B,C)) = [\![\, \exists\, \pi = (s_0, \ldots) \in \Pi_s, \exists\, i \geq 0 : s_i(C) = 1, \forall\, j < i \quad s_j(B) = 1 \,]\!]$.

Значением CTL-формулы B в СП Σ называется множество

$$S_\Sigma(B) \stackrel{\text{def}}{=} \{s \in S_\Sigma \mid s(B) = 1\}.$$

3.1.3 Эквивалентность CTL–формул

Будем называть CTL–формулы B и C **эквивалентными**, если для каждой СП Σ верно равенство $S_\Sigma(B) = S_\Sigma(C)$. Если CTL–формулы B и C эквивалентны, то будем обозначать этот факт записью $B = C$.

Нетрудно доказать, что верны равенства:

- законы де Моргана: $\overline{B \wedge C} = \bar{B} \vee \bar{C}$, $\overline{B \vee C} = \bar{B} \wedge \bar{C}$, $\overline{\overline{B}} = B$,
- $\overline{\mathbf{AX}B} = \mathbf{EX}\bar{B}$,
- $\mathbf{EF}B = \mathbf{EU}(\top, B)$, $\overline{\mathbf{AF}B} = \mathbf{EG}\bar{B}$, $\overline{\mathbf{AG}B} = \mathbf{EF}\bar{B}$,
- $\overline{\mathbf{AU}(B,C)} = \mathbf{EU}(\bar{C}, \bar{B} \wedge \bar{C}) \vee \mathbf{EG}\bar{C}$.

Данные равенства доказываются легко, за исключением последнего.

Докажем последнее равенство. Данное равенство эквивалентно следующему утверждению: для любой СП Σ и любого $s \in S_\Sigma$

$$s\Big(\ \overline{\mathbf{AU}(B,C)}\ \Big) = 0 \quad \Leftrightarrow \quad s\Big(\mathbf{EU}(\bar{C}, \bar{B} \wedge \bar{C}) \vee \mathbf{EG}\bar{C}\Big) = 0. \tag{3.1}$$

Левая часть (3.1) равносильна равенству $s(\mathbf{AU}(B,C)) = 1$, которое равносильно утверждению: $\forall\, \pi = (s_0, \ldots) \in \Pi_s$

$$\exists\, i \geq 0 :\ s_i(C) = 1,\ \forall\, j < i\ \ (s_j(B) = 1) \wedge (s_j(C) = 0). \tag{3.2}$$

Правая часть (3.1) равносильна конъюнкции равенств

$$s(\mathbf{EU}(\bar{C}, \bar{B} \wedge \bar{C})) = 0 \quad \text{и} \quad s(\mathbf{EG}\bar{C}) = 0$$

которая равносильна конъюнкции утверждений

$$\neg \left(\begin{array}{l} \exists\, \pi = (s_0, \ldots) \in \Pi_s : \exists\, i \geq 0 : s_i(\bar{B} \wedge \bar{C}) = 1, \\ \forall\, j < i \quad s_j(\bar{C}) = 1,\ s_j(\bar{B} \wedge \bar{C}) = 0 \end{array} \right)$$
$$\neg\Big(\exists\, \pi = (s_0, \ldots) \in \Pi_s : \forall\, i \geq 0 : s_i(C) = 0\Big)$$

что равносильно утверждению: $\forall\, \pi = (s_0, \ldots) \in \Pi_s$ верна конъюнкция

$$\left\{ \begin{array}{l} \neg \left(\begin{array}{l} \exists\, i \geq 0 : s_i(B) = 0, s_i(C) = 0, \\ \forall\, j < i \quad s_j(C) = 0,\ s_j(B) = 1 \end{array} \right) \\ \neg\Big(\forall\, i \geq 0 : s_i(C) = 0\Big) \end{array} \right\}$$

которую можно переписать следующим образом:

$$\left\{ \begin{array}{l} \forall\, i \geq 0 \quad \Big((s_i(B) = 0) \wedge (s_i(C) = 0)\Big) \rightarrow \\ \qquad\qquad \rightarrow\ \exists\, j < i : \Big(s_j(B) = 0 \text{ или } s_j(C) = 1\Big) \\ \exists\, i \geq 0 : s_i(C) = 1 \end{array} \right\} \tag{3.3}$$

Докажем, что утверждения (3.2) и (3.3) равносильны.

- Пусть верно (3.2), тогда верен второй конъюнктивный член в (3.3).

  Докажем что первый член в (3.3) тоже будет верен. Обозначим символом A импликацию в этом члене, и записью i_0 – тот номер, существование которого утверждается в (3.2). Если $i \leq i_0$, то A верна потому что ложна её посылка. Если $i > i_0$, то A верна, т.к. верно её заключение: в качестве j можно взять i_0.

- Пусть верно (3.3). Докажем, что тогда будет верно (3.2). Пусть i_0 – наименьший из номеров i, таких, что $s_i(C) = 1$ (такой номер существует согласно второму конъюнктивному члену в (3.3)).

  Докажем от противного, что $\forall\, j < i_0\ s_j(B) = 1$. Предположим, что $\exists\, j < i_0 : s_j(B) = 0$, и пусть j_0 – наименьший из таких j. Согласно выбору i_0 и j_0, имеем свойство $s_{j_0}(B) = 0$ и $s_{j_0}(C) = 0$. Согласно импликации в первом конъюнктивном члене в (3.3) для $i = j_0$, получаем: $\exists\, j < j_0 : \Big(s_j(B) = 0$ или $s_j(C) = 1\Big)$. Однако равенство $s_j(C) = 1$ невозможно в силу выбора i_0, поэтому $s_j(B) = 0$, что тоже невозможно в силу выбора j_0. ■

Из приведенных выше соотношений следует, что для любой CTL–формулы существует эквивалентная ей CTL–формула, в которую входят только следующие CTL–операторы: **EX**, **EG**, **EU**.

3.1.4 Примеры свойств программных систем, выражаемых CTL–формулами

Пусть $\mathcal{P}$ содержит утверждения Start, Ready, Request, Acknowledgement, DeviceEnabled, Restart. Используя данные утверждения, можно написать следующие CTL–формулы, выражающие определенные свойства программных систем (после каждой из приводимых ниже CTL–формул мы неформально поясняем ее смысл):

- **EF** (Start $\wedge$ $\overline{\text{Ready}}$)
  (возможно что будет достижимо состояние, в котором свойство Start выполняется, а условие Ready – не выполняется)

- **AG** (Request $\rightarrow$ **AF** Acknowledgement)
  (если получен запрос, то когда-нибудь на него будет дан ответ)

- **AG** (**AF** DeviceEnabled)
  (при любом функционировании системы условие DeviceEnabled выполнено бесконечно много раз)

- **AG** (**EF** Restart)
  (из каждого состояния возможно будет достижимо состояние, в котором выполняется свойство Restart).

3.2 Model checking для CTL

3.2.1 Задача model checking для CTL

Одна из возможных форм задачи **model checking** для CTL (которую мы будем ниже обозначать знакосочетанием **MC-CTL**) заключается в том, чтобы по заданным СП $\Sigma = (S, R, L, S^0, \mathcal{F})$ и CTL–формуле A вычислить множество $S(A)$. Можно считать, что формула A содержит только CTL–операторы вида **EX**, **EG**, **EU**.

Для вычисления $S(A)$ можно использовать излагаемый ниже рекурсивный алгоритм, сводящий вычисление множества $S(A)$ к вычислению множеств вида $S(B)$, где B – подформула формулы A.

- Если $A = p \in \mathcal{P}$, то $S(p) = \begin{cases} \{s \in S \mid L(s,p) = 1\}, & \text{если } p \in \mathcal{P}_\Sigma, \\ \emptyset, & \text{если } p \notin \mathcal{P}_\Sigma. \end{cases}$

- Если A – булева комбинация, то $S(A)$ вычисляется так:

$$S(\top) = S, \quad S(\bot) = \emptyset,$$
$$S(\bar{B}) = S \setminus S(B), \quad S(B \wedge C) = S(B) \cap S(C), \text{ и т.д.}$$

- Если $A = \mathbf{EX}B$, то $S(A) = R^{-1}(S(B)) = \{s \in S \mid R(s) \cap S(B) \neq \emptyset\}$.

- Если $A = \mathbf{EU}(B, C)$, то $S(A)$ вычисляется следующим алгоритмом:
  $S(A) := S' := S(C) \cap S(\mathbf{EG}\top)$,
  while $(S' \neq \emptyset)$
  { выбираем $s \in S'$, и удаляем s из S',
  $\forall\, s' \in R^{-1}(s)$ **если** $\left\{ \begin{array}{l} s' \in S(B) \\ s' \notin S(A) \end{array} \right\}$ **то** добавляем s' к $S(A)$ и к S'
  }

- Пусть $A = \mathbf{EG}B$.

  Согласно определению, равенство $s(\mathbf{EG}B) = 1$ означает, что

$$\exists\, \pi = (s_0, \ldots) \in \Pi_s : \ \pi \subseteq S(B). \tag{3.4}$$

  Поскольку $|S| < \infty$, то $\exists\, i \geq 0 : \inf(\pi) = \{s_j \mid j \geq i\}$.

Будем обозначать записью $S(B)$ не только множество состояний, но также и подграф графа Σ, вершинами которого являются состояния из множества $S(B)$, и рёбрами – рёбра графа Σ, начало и конец которых лежат в $S(B)$. Нетрудно видеть, что

- $\inf(\pi)$ – сильно связный подграф графа $S(B)$ (т.е. такой подграф, что для каждой пары s, s' его вершин существует непустой путь из s в s'), причём $\forall\, F \in \mathcal{F} \quad \inf(\pi) \cap F \neq \emptyset$, и
- этот подграф содержится в некотором максимальном по включению сильно связном подграфе $\mathbf{C}$ графа $S(B)$ (такие подграфы называются **сильно связными компонентами (strongly connected components, SCC**), причем

$$\forall\, F \in \mathcal{F} \quad \mathbf{C} \cap F \neq \emptyset. \tag{3.5}$$

Таким образом, из (3.4) следует, что в графе $S(B)$ существуют

- SCC $\mathbf{C}$, такая, что выполнено (3.5), и
- конечный путь $\hat{\pi}$ из s в некоторое состояние $s' \in \mathbf{C}$.

Верно и обратное – из последнего утверждения следует (3.4): в качестве искомого пути π можно взять путь $\hat{\pi}\tilde{\pi}^{\infty}$, где $\tilde{\pi}$ – цикл из s' в s', проходящий через каждую вершину $\mathbf{C}$.

Таким образом, возможен следующий алгоритм вычисления $S(A)$:

- находим в $S(B)$ все SCC (можно использовать **алгоритм Тарьяна** нахождения всех SCC в произвольном графе G, его сложность равна $O(|G|)$, где $|G|$ – число вершин и рёбер в G),
- оставляем только те из найденных SCC, которые удовлетворяют условию (3.5),
- полагаем $S(A) := S' :=$ множеству всех состояний, входящих в эти оставшиеся SCC, и затем работает цикл, аналогичный циклу в предыдущем пункте.

Нетрудно доказать, что сложность данного алгоритма – $O(|A|\,|\Sigma|\,|\mathcal{F}|)$, где $|A|$ – размер формулы A, $|\Sigma|$ – число вершин и рёбер в Σ, $|\mathcal{F}|$ – число условий fairness в Σ.

3.3 Решение задачи MC-CTL на основе понятия неподвижной точки

3.3.1 Неподвижные точки монотонных операторов

Пусть S – конечное множество, и $\mathbf{2}^S$ – множество всех подмножеств S.

Будем называть **оператором** на $\mathbf{2}^S$ отображение вида $f : \mathbf{2}^S \to \mathbf{2}^S$.

Оператор f на $\mathbf{2}^S$ называется **монотонным**, если

$$\forall\, A, B \subseteq S \quad A \subseteq B \Rightarrow f(A) \subseteq f(B).$$

Множество $A \subseteq S$ называется **неподвижной точкой (fixpoint, FP)** оператора f, если верно равенство $A = f(A)$.

FP оператора f на $\mathbf{2}^S$ называется **наименьшей** (**наибольшей**) FP, и обозначается $\mu x.f(x)$ ($\nu x.f(x)$), если она – наименьшая (наибольшая) по включению FP f, соответственно.

Теорема 1.

Если S – конечное множество, и f – монотонный оператор на $\mathbf{2}^S$, то существуют подмножества $A, B \subseteq S$, являющиеся наименьшей и наибольшей FP f, соответственно.

Доказательство.

Определим последовательности $A_0, A_1, \ldots$ и $B_0, B_1, \ldots$ подмножеств множества S следующим образом:

$$A_0 = \emptyset,\ B_0 = S,\ \forall\, i \geq 0\ \ A_{i+1} = f(A_i),\ B_{i+1} = f(B_i).$$

Т.к. $A_0 = \emptyset \subseteq A_1$, то из монотонности f следует, что $f(A_0) \subseteq f(A_1)$, т.е. $A_1 \subseteq A_2$. Аналогично получаем: $A_2 = f(A_1) \subseteq f(A_2) = A_3$, и т.д., т.е. последовательность $A_0, A_1, \ldots$ – неубывающая цепь: $A_0 \subseteq A_1 \subseteq \ldots$

Аналогично, из $B_0 = S \supseteq B_1$ и монотонности f следует $f(B_0) \supseteq f(B_1)$, т.е. $B_1 \supseteq B_2$, и т.д., т.е. $B_0, B_1, \ldots$ – невозрастающая цепь: $B_0 \supseteq B_1 \supseteq \ldots$

Поскольку $\forall\, i \geq 0\ \ A_i, B_i \subseteq S$, и множество S конечно, то

$$\exists\, i_0, j_0 \geq 0 :\ \ A_{i_0} = A_{i_0+1},\ \ B_{j_0} = B_{j_0+1},$$

или $A_{i_0} = f(A_{i_0})$ и $B_{j_0} = f(B_{j_0})$, т.е. A_{i_0} и B_{j_0} – FP f.

Докажем, что A_{i_0} и B_{j_0} – наименьшая и наибольшая FP f, соответственно. Пусть C – произвольная FP f, т.е. $f(C) = C$. Из включений

$$A_0 = \emptyset \subseteq C \subseteq S = B_0$$

и монотонности f следует, что $f(A_0) \subseteq f(C) \subseteq f(B_0)$, т.е. $A_1 \subseteq C \subseteq B_1$.

Аналогично получаем $A_2 \subseteq C \subseteq B_2$, и т.д., т.е. $\forall\, i \geq 0\ \ A_i \subseteq C \subseteq B_i$. В частности, $A_{i_0} \subseteq C \subseteq B_{j_0}$. ■

3.3.2 Вычисление множеств $S(\mathbf{EU}(B,C))$ и $S(\mathbf{EG}B)$ на основе понятия неподвижной точки

Пусть задана СП $\Sigma = (S, R, \ldots)$.

Нетрудно доказать, что следующие операторы монотонны:

- операторы $A\cap$ и $A\cup : \mathbf{2}^S \to \mathbf{2}^S$ (где $A \subseteq S$ – фиксированное подмножество), сопоставляющие каждому $x \subseteq S$ множества $A \cap x$ и $A \cup x$ соответственно, и
- оператор $\mathbf{EX} : \mathbf{2}^S \to \mathbf{2}^S$, сопоставляющий каждому $x \subseteq S$ множество $R^{-1}(x) = \{s \in S \mid R(s) \cap x \neq \emptyset\}$.

Т.к. композиция монотонных операторов является монотонным оператором, то, в частности, операторы $f_1, f_2 : \mathbf{2}^S \to \mathbf{2}^S$, такие, что

$$\forall\, x \subseteq S \quad f_1(x) = A \cup (B \cap \mathbf{EX}(x)), \quad f_2(x) = B \cap \mathbf{EX}(x),$$

(где $A, B \subseteq S$ – фиксированные множества) являются монотонными.

Излагаемая ниже теорема описывает свойства этих операторов.

Теорема 2.

Пусть задана СП $\Sigma = (S, R, L)$, причем $\forall\, s \in S \;\; R(s) \neq \emptyset$.

Тогда $\forall\, B, C \in Fm_{CTL}$

$$\begin{aligned} &S(\mathbf{EU}(B,C)) = \mu x.\Big(S(C) \cup (S(B) \cap \mathbf{EX}(x))\Big),\\ &S(\mathbf{EG}B) = \nu x.\Big(S(B) \cap \mathbf{EX}(x)\Big). \end{aligned} \tag{3.6}$$

Доказательство.

Пусть f_1 и f_2 – операторы, соответствующие правым частям равенств в (3.6), и $A_0, A_1, \ldots$ и $B_0, B_1, \ldots$ – последовательности подмножеств S, такие, что $A_0 = \emptyset$, $B_0 = S$, и $\forall\, i \geq 0$ $A_{i+1} = f_1(A_i)$ и $B_{i+1} = f_2(B_i)$.

Из данных определений следует, что $\forall\, i \geq 0$

- $A_{i+1} = \{s \in S \mid \exists\, j \leq i, \exists \text{ путь } (s = s_0, \ldots, s_j) :$
  $\{s_0, \ldots, s_{j-1}\} \subseteq S(B), s_j \in S(C)\}$,
- $B_{i+1} = \{s \in S \mid \exists \text{ путь } (s = s_0, \ldots, s_i) \subseteq S(B)\}$.

Поэтому правые части равенств в (3.6) имеют вид

$$\begin{aligned} \textstyle\bigcup_{i\geq 0} A_i = \{s \in S \mid \exists\, i \geq 0 : \exists\, j \leq i, \exists \text{ путь } (s = s_0, \ldots, s_j) :\\ \{s_0, \ldots, s_{j-1}\} \subseteq S(B), s_j \in S(C)\}, \end{aligned} \tag{3.7}$$

$$\bigcap_{i\geq 0} B_i = \{s \in S \mid \forall\, i \geq 0\ \exists \text{ путь } (s = s_0, \ldots, s_i) \subseteq S(B)\}. \qquad (3.8)$$

Принимая во внимание условие $\forall\, s \in S\ \ R(s) \neq \emptyset$, заключаем, что $(3.7) = S(\mathbf{EU}(B, C))$.

Докажем, что $(3.8) = S(\mathbf{EG}B)$.

- Пусть $s \in S(\mathbf{EG}B)$, тогда $\exists$ бесконечный путь $(s = s_0, \ldots) \subseteq S(B)$, поэтому $s \in (3.8)$.
- Пусть $s \in (3.8)$, тогда $\exists\, \pi = (s = s_0, \ldots, s_i) \subseteq S(B)$, где $i > |S|$.

  Пусть s_j – состояние, входящее в π более одного раза. Обозначим

  – записью $\hat{\pi}$ префикс $(s_0, \ldots, s_j)$ пути π, и
  – записью $\tilde{\pi}$ – цикл из s_j в s_j, содержащийся в π.

  Имеем: $\hat{\pi}\tilde{\pi}^{\infty}$ – бесконечный путь, все состояния которого $\in S(B)$, поэтому $s \in S(\mathbf{EG}B)$. ■

Если множество $\mathcal{F}_\Sigma$ нетривиально, то процедура вычисления множеств $S(\mathbf{EU}(B, C))$ и $S(\mathbf{EG}B)$ имеет более сложный вид.

Ниже $\forall\, B, C \subseteq S$ запись $\mathbf{EU}(B, C)$ обозначает множество

$$\mu x.(C \cup (B \cap \mathbf{EX}(x))).$$

Нетрудно доказать, что оператор

$$f : \mathbf{2}^S \to \mathbf{2}^S, \quad \forall\, x \subseteq S\ \ f(x) = \mathbf{EU}(B, x),$$

где $B \subseteq S$ – фиксированное подмножество, является монотонным.

Теорема 3.
Пусть задана СП $\Sigma = (S, \ldots)$, причём $\mathcal{F}_\Sigma = \{F_1, \ldots, F_k\}$.
Тогда $\forall\, B \in Fm_{CTL} \quad S(\mathbf{EG}B) = \nu x.f(x)$, где

$$f(x) = S(B) \cap \textstyle\bigcap_{i=1}^{k} \mathbf{EX}\ \mathbf{EU}(S(B), F_i \cap x).$$

Доказательство.
По определению, $S(\mathbf{EG}B) = \{s \in S \mid \exists\, \pi \in \Pi_s : \pi \subseteq S(B)\}$.
Включение $S(\mathbf{EG}B) \subseteq \nu x.f(x)$ следует из того, что $S(\mathbf{EG}B)$ – FP f:

$$S(\mathbf{EG}B) = f(S(\mathbf{EG}B)).$$

Включение $\nu x.f(x) \subseteq S(\mathbf{EG}B)$ следует из импликации

$$x = f(x)\ \Rightarrow\ x \subseteq S(\mathbf{EG}B). \blacksquare$$

Если для какого-либо $s \in S$ есть предположение, что $s(\mathbf{EG}B) = 1$, и для обоснования этого требуется построить $\pi \in \Pi_s : \pi \subseteq S(B)$, то это можно делать в процессе вычисления соответствующих неподвижных точек (не дожидаясь окончания их построения) следующим образом:

- вычисляя $\mathbf{EU}\Big(S(B), F_1 \cap x\Big)$, строим $\pi_1 \subseteq S(B)$ из s в $s_1 \in F_1 \cap x$,
- так же строим $\pi_2 \subseteq S(B)$ из s_1 в $s_2 \in F_2 \cap x$, и т.д.

Используя $\pi_1, \pi_2, \ldots$, строим искомый путь π в виде $\hat{\pi}\tilde{\pi}^{\infty}$.

3.3.3 Алгоритм решения задачи MC-CTL на основе понятия неподвижной точки

Пусть заданы СП $\Sigma = (S, R, L)$ и формула $A \in Fm_{CTL}$.

Используя изложенные выше методы, можно построить рекурсивный алгоритм вычисления множества $S(A)$. Данный алгоритм отличается от алгоритма из пункта 3.2.1, лишь в тех пунктах, которые связаны с вычислением множеств вида $S(\mathbf{EU}(B, C))$ и $S(\mathbf{EG}B)$. Для вычисления этих множеств можно использовать соотношения (3.6).

3.4 μ–исчисление

CTL можно вложить в более выразительную логику, называемую **μ–исчислением**. Это позволяет свести задачу MC-CTL к некоторой задаче μ–исчисления.

μ–исчисление позволяет описывать свойства СП с несколькими отношениями перехода, в которых вместо одного отношения перехода R_Σ имеется совокупность $\{R^a \subseteq S_\Sigma \times S_\Sigma \mid a \in T\}$, где T – заданное множество, элементы которого называются **переходами**.

3.4.1 μ–формулы

Множество Fm_μ формул μ–исчисления (называемых **μ–формулами**) содержит $\mathcal{P}$, замкнуто относительно булевых комбинаций, и также

- содержит множество RV, элементы которого называются **реляционными переменными (РП)**,
- $\forall\, a \in T,\ \forall\, A \in Fm_\mu \quad [a]A \in Fm_\mu$ и $\langle a\rangle A \in Fm_\mu$,
- $\forall\, x \in RV,\ \forall\, A \in Fm_\mu \quad \mu x.A \in Fm_\mu$ и $\nu x.A \in Fm_\mu$.

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

- $\forall\, x \in RV$ вхождение x в μ–формулу x – свободное,
- если A имеет вид $\bar{B}$, $B \wedge C$, $B \vee C$, $[a]B$, $\langle a \rangle B$, то каждому вхождению каждой РП x в A соответствует некоторое вхождение x в B или C, и каждое вхождение x в A имеет тот же статус (свободное или связанное), который имеет соответствующее вхождение x в B или C
- если A имеет вид $\mu x.B$ или $\nu x.B$, то
  - все свободные вхождения x в B, а также вхождение x рядом с μ или ν, становятся связанными в A, и образуют (вместе с первым вхождением x в A) одну **группу связанности**, и
  - все остальные вхождения РП в A имеют тот же статус, который имеют соответствующие им вхождения этих РП в B.

Для каждой μ–формулы A

- $fv(A)$ обозначает совокупность всех РП, имеющих свободные вхождения в A, и
- $Sub(A)$ обозначает совокупность всех подформул формулы A.

μ–формула называется **правильной**, если для каждой её подформулы вида $\mu x.B$ или $\nu x.B$ число отрицаний в B, располагающихся над каждым свободным вхождением x в B, является чётным. Ниже каждая рассматриваемая μ–формула предполагается правильной.

3.4.2 Значения μ–формул

Пусть задана СП $\Sigma = (S, \{R^a \mid a \in T\}, L, S^0)$.

Означиванием РП в СП Σ называется отображение $\zeta : RV \to \mathbf{2}^S$. Множество всех означиваний РП в СП Σ обозначается записью $RV_\Sigma^\bullet$.

$RV_\Sigma^\bullet$ можно рассматривать как частично упорядоченное множество:

$$\forall\, \zeta, \zeta' \in RV_\Sigma^\bullet \quad \zeta \leq \zeta' \;\Leftrightarrow\; \forall\, x \in RV \quad \zeta(x) \subseteq \zeta'(x). \tag{3.9}$$

Значением μ–формулы A в СП Σ на означивании $\zeta \in RV_\Sigma^\bullet$ называется подмножество $A^\zeta \subseteq S$, определяемое рекурсивно следующим образом:

- $\forall\, p \in \mathcal{P} \;\; p^\zeta = S(p) = \begin{cases} \{s \in S \mid L(s,p) = 1\}, & \text{если } p \in \mathcal{P}_\Sigma, \\ \emptyset, & \text{если } p \notin \mathcal{P}_\Sigma, \end{cases}$

- $\forall\, x \in RV \;\; x^\zeta = \zeta(x)$,
- $\top^\zeta = S$, $\bot^\zeta = \emptyset$, $\bar{B}^\zeta = S \setminus B^\zeta$, $(B \wedge C)^\zeta = B^\zeta \cap C^\zeta$, и т.д.,
- $([a]B)^\zeta = \{s \in S \mid R^a(s) \subseteq B^\zeta\}$,
- $(\langle a\rangle B)^\zeta = (R^a)^{-1}(B^\zeta) = \{s \in S \mid R^a(s) \cap B^\zeta \neq \emptyset\}$,
- $(\mu x.B)^\zeta = \mu x.(y \mapsto B^{\zeta[x:=y]})$, $(\nu x.B)^\zeta = \nu x.(y \mapsto B^{\zeta[x:=y]})$, где

$$y \mapsto B^{\zeta[x:=y]} \tag{3.10}$$

  обозначает оператор, сопоставляющий каждому $S' \in \mathbf{2}^S$ множество $B^{\zeta[x:=S']} \in \mathbf{2}^S$ (монотонность (3.10) будет доказана в теореме 4), и

$$\zeta[x := S'](x) = S', \;\; \forall\, y \in RV \setminus \{x\} \;\; \zeta[x := S'](y) = \zeta(y).$$

Заметим, что A^ζ зависит только от значений ζ на РП из $fv(A)$.

Будем обозначать записью Σ_A функцию вида $RV_\Sigma^\bullet \to \mathbf{2}^S$, сопоставляющую каждому означиванию $\zeta \in RV_\Sigma^\bullet$ значение A^ζ.

Будем называть μ–формулы A и B **эквивалентными**, если для каждой СП Σ фунции Σ_A и Σ_B совпадают. Если μ–формулы A и B эквивалентны, то будем обозначать этот факт записью $A = B$.

Нетрудно доказать, что

$$\forall\, A \in Fm_\mu,\; \forall\, a \in T \quad \overline{[a]A} = \langle a\rangle \bar{A}, \;\; \overline{\langle a\rangle A} = [a]\bar{A}. \tag{3.11}$$

Теорема 4.

Пусть формула $A \in Fm_\mu$ такова, что $\forall\, x \in RV$ число отрицаний в A, располагающихся над каждым свободным вхождением x в A, чётное.

Тогда для каждой СП Σ функция Σ_A – монотонная.

Доказательство.

Обозначим записью $\iota(A)$ число вхождений в A связок $\wedge$, $\vee$, $[a]$, $\langle a\rangle$, μx, νx. Будем доказывать теорему индукцией по $\iota(A)$

- Если A имеет вид $\top$, $\bot$, p, или x (где $p \in \mathcal{P}$, $x \in RV$), то монотонность Σ_A тривиальна.
- Пусть $A = B \wedge C$ или $B \vee C$.

  B и C удовлетворяют условию теоремы, и $\iota(B) < \iota(A)$, $\iota(C) < \iota(A)$, поэтому по и.п. функции Σ_B и Σ_C монотонны.

  Σ_A – суперпозиция этих функций и монотонных функций $\cap$ или $\cup : \mathbf{2}^S \to \mathbf{2}^S$ соответственно, поэтому Σ_A тоже монотонна.

- Пусть $A = [a]B$ или $\langle a\rangle B$.

  Формула B удовлетворяет условию теоремы, и $\iota(B) < \iota(A)$, поэтому по и.п. функция Σ_B монотонна.

  Σ_A – суперпозиция функции Σ_B и монотонных функций $[a]$ или $\langle a\rangle : \mathbf{2}^S \to \mathbf{2}^S$ соответственно, где $\forall\, x \in \mathbf{2}^S$

  $$[a](x) = \{s \in S \mid R^a(s) \subseteq x\}, \quad \langle a\rangle(x) = (R^a)^{-1}(x)$$

  поэтому Σ_A тоже монотонна.

- Пусть $A = \mu x.B$ или $\nu x.B$.

  Из условия теоремы и определения правильной формулы следует, что формула B удовлетворяет условию теоремы, и $\iota(B) < \iota(A)$, поэтому по и.п. функция Σ_B монотонна. В частности, $\forall\, \zeta : RV \to \mathbf{2}^S$ оператор $y \mapsto B^{\zeta[x:=y]}$ монотонный.

  Докажем монотонность функции Σ_A, т.е. утверждение

  $$\forall\, \zeta, \zeta' \in RV_\Sigma^\bullet \quad \text{если } \ \zeta \le \zeta', \ \text{ то } \ A^\zeta \subseteq A^{\zeta'}.$$

  Рассмотрим лишь случай $A = \mu x.B$ (случай $A = \nu x.B$ рассматривается аналогично).

  По определению значения формул вида $\mu x.B$, $\exists\, n \ge 0$: $A^\zeta = S_n$, $A^{\zeta'} = S'_n$ где S_n, S'_n – последние члены последовательностей

  $$S_0 = \emptyset,\ S_1 = B^{\zeta[x:=S_0]},\ S_2 = B^{\zeta[x:=S_1]},\ \ldots, S_{n+1} = B^{\zeta[x:=S_n]} = S_n,$$
  $$S'_0 = \emptyset,\ S'_1 = B^{\zeta'[x:=S'_0]},\ S'_2 = B^{\zeta'[x:=S'_1]},\ \ldots, S'_{n+1} = B^{\zeta'[x:=S'_{n]}} = S'_n.$$

  $\forall\, i = 0, \ldots, n$ из включения $S_i \subseteq S'_i$ и неравенства $\zeta \le \zeta'$ следует неравенство $\zeta[x := S_i] \le \zeta'[x := S'_i]$, откуда, учитывая монотонность функции Σ_B, получаем: $S_{i+1} = B^{\zeta[x:=S_i]} \subseteq B^{\zeta'[x:=S'_i]} = S'_{i+1}$.

  Таким образом, поскольку $S_0 = \emptyset \subseteq S'_0$, то $\forall\, i = 0, \ldots, n \ \ S_i \subseteq S'_i$.

  В частности, $A^\zeta = S_n \subseteq S'_n = A^{\zeta'}$.

- Пусть $A = \bar{B}$.

  Случаи $B = p \in \mathcal{P}$, $B = \top$ или $B = \bot$ тривиальны.

  Случай $B = x \in RV$ невозможен по условию теоремы.

  Если $B = \bar{C}$, то вместо формулы A рассматриваем эквивалентную ей формулу C (которая имеет меньший размер, удовлетворяет условию теоремы, и $\iota(C) = \iota(A)$).

Если B имеет вид $C \wedge D$, $C \vee D$, $[a]C$, или $\langle a\rangle C$, то вместо формулы A рассматриваем эквивалентную ей формулу $\bar{C} \vee \bar{D}$, $\bar{C} \wedge \bar{D}$, $\langle a\rangle \bar{C}$, или $[a]\bar{C}$, соответственно. Формулы $\bar{C}$ и $\bar{D}$, удовлетворяют условию теоремы, $\iota(\bar{C}) < \iota(A)$ и $\iota(\bar{D}) < \iota(A)$, далее рассуждаем как в соответствующих пунктах выше.

- Случаи $B = \mu x.C$ и $B = \nu x.C$. Рассмотрим лишь первый случай (второй случай рассматривается аналогично), т.е. $A = \overline{\mu x.C}$.

  Обозначим записью $C(\bar{x})$ формулу, получаемую из C заменой всех свободных вхождений x на $\bar{x}$. Из условия теоремы и определения правильной формулы следует, что $\overline{C(\bar{x})}$ удовлетворяет условию теоремы, и $\iota(\,\overline{C(\bar{x})}\,) < \iota(A)$, поэтому по и.п. функция $\Sigma_{\overline{C(\bar{x})}}$ монотонна.

  $\forall\, \zeta : RV \to \mathbf{2}^S$ оператор $y \mapsto C^{\zeta[x:=y]}$ монотонный, т.к. если $y_1 \subseteq y_2$, то $\bar{y}_1 \supseteq \bar{y}_2$, и из монотонности функции $\Sigma_{\overline{C(\bar{x})}}$ следует, что

  $$(\,\overline{C}\,)^{\zeta[x:=y_1]} = (\,\overline{C(\bar{x})}\,)^{\zeta[x:=\bar{y}_1]} \supseteq (\,\overline{C(\bar{x})}\,)^{\zeta[x:=\bar{y}_2]} = (\,\overline{C}\,)^{\zeta[x:=y_2]},$$

  т.е. $C^{\zeta[x:=y_1]} \subseteq C^{\zeta[x:=y_2]}$.

  Докажем монотонность $\Sigma_A = \Sigma_{\overline{\mu x.C}}$, т.е. утверждение

  $$\forall\, \zeta, \zeta' \in RV_\Sigma^\bullet \quad \text{если } \zeta \le \zeta', \text{ то } (\mu x.C)^\zeta \supseteq (\mu x.C)^{\zeta'}.$$

  По определению значения формул вида $\mu x.C$, $\exists\, n \ge 0$: $(\mu x.C)^\zeta$ равно последнему члену последовательности

  $$\begin{aligned} &S_0 = \emptyset,\ S_1 = C^{\zeta[x:=S_0]},\ S_2 = C^{\zeta[x:=S_1]},\ \ldots, \\ &S_{n+1} = C^{\zeta[x:=S_n]} = S_n. \end{aligned} \tag{3.12}$$

  Нетрудно видеть, что $\forall\, i \ge 0$ равенство $S_{i+1} = C^{\zeta[x:=S_i]}$ равносильно равенству $S \setminus S_{i+1} = (\,\overline{C(\bar{x})}\,)^{\zeta[x:=S\setminus S_i]}$, поэтому можно переписать (3.12) следующим образом:

  $$\begin{aligned} &S \setminus S_0 = S,\ S \setminus S_1 = (\,\overline{C(\bar{x})}\,)^{\zeta[x:=S\setminus S_0]}, \\ &S \setminus S_2 = (\,\overline{C(\bar{x})}\,)^{\zeta[x:=S\setminus S_1]},\ \ldots, \\ &S \setminus S_{n+1} = (\,\overline{C(\bar{x})}\,)^{\zeta[x:=S\setminus S_n]} = S \setminus S_n. \end{aligned} \tag{3.13}$$

  Из (3.13) и из определения значений формул вида $\nu x.A$ следует, что $S \setminus S_n = (\nu x.\overline{C(\bar{x})}\,)^\zeta$, т.е. $(\mu x.C)^\zeta = S \setminus (\nu x.\overline{C(\bar{x})}\,)^\zeta$.

  Выше было установлено, что из монотонности функции $\Sigma_{\overline{C(\bar{x})}}$ следует монотонность функции $\Sigma_{\nu x.\overline{C(\bar{x})}}$, поэтому из $\zeta \le \zeta'$ следует

включение $(\nu x.\overline{C(\bar{x})}\,)^{\varsigma} \subseteq (\nu x.\overline{C(\bar{x})}\,)^{\varsigma'}$, из которого следует желаемое соотношение

$$(\mu x.C)^{\varsigma} = S \setminus (\nu x.\overline{C(\bar{x})}\,)^{\varsigma} \supseteq S \setminus (\nu x.\overline{C(\bar{x})}\,)^{\varsigma'} = (\mu x.C)^{\varsigma'}. \ \blacksquare$$

Отметим, что мы попутно доказали утверждение:

$$\forall\, A \in Fm_{\mu} \quad \overline{\mu x.A} = \nu x.\overline{A(\bar{x})}, \ \ \overline{\nu x.A} = \mu x.\overline{A(\bar{x})}. \tag{3.14}$$

где $A(\bar{x})$ получается из A заменой всех свободных вхождений x на $\bar{x}$.

3.4.3 Ускоренное вычисление значений μ–формул

Определение значения A^{ς} в пункте 3.4.2 является также и алгоритмом вычисления этого значения. Этот алгоритм имеет сложность $O(|\Sigma|^{|A|})$.

Вычисление значений формул вида $\mu x.B$ и $\nu x.B$ можно ускорить. Ниже излагается рекурсивный алгоритм вычисления значения A^{ς} для формулы A вида $\mu x.B$, сложность которого – $O(|\Sigma|^{d})$, где d – **глубина чередования** формулы A, определяемая как максимальная длина последовательности $B_1, \ldots, B_k$ подформул формулы A, каждая из которых начинается с μ или ν, $\forall\, i = 1, \ldots, k-1$ B_{i+1} – подформула формулы B_i, и если B_i начинается с μ, то B_{i+1} начинается с ν, а если B_i начинается с ν, то B_{i+1} начинается с μ.

Алгоритм имеет следующий вид.

1. Пронесём в A все отрицания вниз, используя законы де Моргана и утверждения (3.11) и (3.14). Получившуюся формулу обозначим тем же символом A.

   Из определения правильности μ–формулы следует, что после этого все отрицания будут располагаться только над атомарными утверждениями и свободными вхождениями РП.

   Переименуем связанные РП (если это необходимо) так, чтобы в каждой группе связанности в A та РП, которая входит в эту группу, отличалась бы от любой РП, имеющей вхождение в A вне этой группы.

2. Пусть список всех подформул формулы A (включая саму A), которые начинаются с μ и не содержатся в подформулах, начинающихся с ν, имеет вид $\{\mu x.B_x \mid x \in X\}$.

   Будем использовать следующие обозначения:

   - $RV^{\bullet}_{\Sigma,X} = \{\sigma \in RV^{\bullet}_{\Sigma} \mid \forall\, x \in RV \setminus X \ \ \sigma(x) = \zeta(x)\}$,

- $\forall\, B \in Sub(A)$ запись $\hat{B}$ обозначает формулу, получаемую из B удалением входящих в нее записей $\mu x.$, где $x \in X$, имеем:

$$\forall\, B \in Sub(A) \quad fv(\hat{B}) \subseteq fv(A) \sqcup X,$$

- $f_{\hat{A}}$ – оператор вида $RV^{\bullet}_{\Sigma,X} \to RV^{\bullet}_{\Sigma,X}$, определяемый следующим образом:

$$\forall\, \sigma \in RV^{\bullet}_{\Sigma,X},\ \forall\, x \in X \quad f_{\hat{A}}(\sigma)(x) = \hat{B}^{\sigma}_{x}. \tag{3.15}$$

3. Вычисляем последовательность $\sigma_0, \sigma_1, \ldots$ означиваний из $RV^{\bullet}_{\Sigma,X}$:

$$\forall\, x \in X\ \ \sigma_0(x) = \emptyset, \quad \forall i \geq 0\ \ \sigma_{i+1} = f_{\hat{A}}(\sigma_i).$$

Если для некоторого i верно равенство $\sigma_{i+1} = \sigma_i$, то алгоритм завершает работу, искомое значение A^{ζ} полагается равным $\hat{A}^{\sigma_i}$.

Вычисление значений $(\nu x.B)^{\zeta}$ производится двойственным образом.

Теорема 5.

Приведенный выше алгоритм всегда завершается и вычисляет правильное значение A^{ζ}.

Доказательство.

Используя теорему 4, нетрудно доказать, что оператор $f_{\hat{A}}$ является монотонным относительно порядка (3.9).

Поскольку σ_0 – наименьший элемент относительно этого порядка, то последовательность $\sigma_0, \sigma_1, \ldots$ является цепью.

Поскольку множество $RV^{\bullet}_{\Sigma,X}$ конечно, то $\exists\, i \geq 0 : \sigma_i = \sigma_{i+1}$, т.е. описанный выше алгоритм всегда завершается.

Докажем равенство $\hat{A}^{\sigma_i} = A^{\zeta}$.

Из алгоритма вычисления значения A^{ζ} следует, что

$$\exists\, \sigma_* \in RV^{\bullet}_{\Sigma,X} : \quad A^{\zeta} = \hat{A}^{\sigma_*} \quad \text{и} \quad \forall\, x \in X\ \ \sigma_*(x) = \hat{B}^{\sigma_*}_{x}.$$

Из (3.15) следует, что $\sigma_* = f_{\hat{A}}(\sigma_*)$, т.е. σ_* – неподвижная точка $f_{\hat{A}}$.

По построению, σ_i – наименьшая неподвижная точка $f_{\hat{A}}$, т.е.

$$\forall\, x \in X\ \ \hat{B}^{\sigma_i}_{x} = \sigma_i(x) \subseteq \sigma_*(x) = \hat{B}^{\sigma_*}_{x}.$$

С другой стороны, $\forall\, x \in X\ \ \hat{B}^{\sigma_*}_{x} \subseteq \hat{B}^{\sigma_i}_{x}$, т.к. σ_* получается в результате итераций, начиная с σ_0 ($\forall\, x \in X\ \ \sigma_0(x) = \emptyset$), которое $\leq \sigma_i$, и каждый шаг итерации заключается в переходе от означивания $\sigma \leq \sigma_i$ к означиванию σ', которое тоже будет $\leq \sigma_i$, т.к. $\forall\, x \in X$ либо $\sigma'(x) = \sigma(x)$, либо $\sigma'(x) = \emptyset$, либо $\sigma'(x) = \hat{B}^{\sigma}_{x} \leq \hat{B}^{\sigma_i}_{x} = \sigma_i(x)$, поэтому $\sigma_* \leq \sigma_i$.

Таким образом, $\forall\, x \in X\ \ \hat{B}^{\sigma_*}_{x} = \hat{B}^{\sigma_i}_{x}$.

Поскольку $A = B_x$ для некоторого $x \in X$, то $A^{\zeta} = \hat{A}^{\sigma_*} = \hat{A}^{\sigma_i}$. ■

3.4.4 Вложение CTL в μ–исчисление

Пусть $T = \{a\}$.

Каждой CTL–формуле A, в которую входят только CTL–операторы **EX**, **EG**, **EU**, можно сопоставить замкнутую μ–формулу A_μ, получаемую из A заменой в ней подформул, начинающихся с CTL–операторов, на μ–формулы, по правилу:

- $\mathbf{EX}B$ заменяется на $\langle a \rangle B$,
- $\mathbf{EU}(B, C)$ заменяется на $\mu x.(C \vee (B \wedge \langle a \rangle x))$,
- $\mathbf{EG}B$ заменяется на $\nu x.(B \wedge \langle a \rangle x)$.

Пусть задана СП $\Sigma = (S, R, L)$, где $\forall\, s \in S\ R(s) \neq \emptyset$. Обозначим записью Σ_μ СП $(S, \{R^a\}, L)$, где $R^a = R$. Из теоремы 2 следует, что $\forall\, A \in Fm_{CTL}$ значение $S(A)$ в Σ совпадает со значением A_μ в Σ_μ.

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

Глава 4

Бинарные диаграммы решений

4.1 Бинарные диаграммы решений и связанные с ними понятия

Ниже запись $\mathcal{X}_{\mathbf{B}}$ обозначает множество $\{x \in \mathcal{X} \mid \tau(x) = \mathbf{B}\}$.

4.1.1 Определение бинарной диаграммы решений

Пусть задано множество переменных $X \subseteq \mathcal{X}_{\mathbf{B}}$.

Бинарная диаграмма решений (Binary Decision Diagram, BDD) над множеством переменных X – это ациклический граф Δ с выделенной начальной вершиной Δ^0. Вершины Δ делятся на два класса:

- **терминальные**, из них не выходит ни одного рёбра, и
- **нетерминальные**, из каждой из них выходят два ребра, одно из которых имеет метку 0, а другое – метку 1.

Множество вершин BDD Δ обозначается тем же символом Δ. Каждая вершина $v \in \Delta$ имеет метку $l(v)$. Если v терминальная, то $l(v) \in \{0, 1\}$, а если v нетерминальная, то $l(v) \in X$.

$\forall\, \xi \in X^{\bullet}$ определено значение $\Delta^{\xi} \in \{0, 1\}$, которое вычисляется путем прохода по Δ, начиная с Δ^0. В каждый момент этого прохода

- если мы находимся в нетерминальной вершине v, то переходим к следующей вершине по ребру с меткой $\xi(l(v))$, и
- если мы находимся в терминальной вершине, то вычисление заканчивается, и в качестве Δ^{ξ} выдаётся значение $l(v)$.

$\forall\, X \subseteq \mathcal{X}_{\mathbf{B}}$ будем обозначать записью Δ_X совокупность всех BDD над множеством переменных X.

4.1.2 Эквивалентность и изоморфность бинарных диаграмм решений

Пусть задано множество переменных $X \subseteq \mathcal{X}_{\mathbf{B}}$.

BDD Δ_1 и Δ_2 из Δ_X называются

- **эквивалентными**, если $\forall\, \xi \in X^\bullet \;\; \Delta_1^\xi = \Delta_2^\xi$, и
- **изоморфными**, если существует биекция $\varphi : \Delta_1 \to \Delta_2$, такая, что
  - $\varphi(\Delta_1^0) = \Delta_2^0, \quad \forall\, v \in \Delta_1 \;\; l(v) = l(\varphi(v))$, и
  - Δ_1 содержит ребро из v в v' с меткой $l \in \{0, 1\}$ тогда и только тогда, когда Δ_2 содержит ребро $\varphi(v)$ в $\varphi(v')$ с меткой l.

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

Нетрудно построить алгоритм проверки изоморфности BDD Δ_1 и Δ_2 со сложностью $O(|\Delta_1| + |\Delta_2|)$, заключающийся в попытке определить отображение $\varphi : \Delta_1 \to \Delta_2$, обладающее указанными выше свойствами:

- сначала значение φ определяется на Δ_1^0, и
- затем выполняется цикл: если для какой-либо вершины $v \in \Delta_1$ значение $\varphi(v)$ уже определено, то однозначно определяется значение φ на концах рёбер, выходящих из v, так, чтобы выполнялись условия на φ из предыдущего абзаца: $\forall\, b \in \{0, 1\}$ если v_b – конец выходящего из v ребра с меткой b, то $\varphi(v_b)$ – конец выходящего из $\varphi(v)$ ребра с меткой b.

4.1.3 Представление множеств означиваний бинарными диаграммами решений

Пусть заданы множество переменных $X \subseteq \mathcal{X}_{\mathbf{B}}$, BDD $\Delta \in \Delta_X$, и подмножество $S \subseteq X^\bullet$.

Будем говорить что Δ является **представлением** множества S, если

$$S = \{\xi \in X^\bullet \mid \Delta^\xi = 1\}.$$

Понятие представления множеств означиваний при помощи BDD можно распространить на множества означиваний $S \subseteq X^\bullet$, где не все переменные в X имеют тип $\mathbf{B}$, излагаемым ниже образом.

Пусть значениями каждой переменной $x \in X$ являются битовые строки длины d_x. Обозначим записью X_x совокупность из d_x новых переменных типа $\mathbf{B}$, и записью $\tilde{X}$ – множество $\bigsqcup_{x \in X} X_x$. Нетрудно видеть, что

имеется естественная биекция

$$f : X^\bullet \to \tilde{X}^\bullet$$

сопоставляющая каждому $\xi \in X^\bullet$ означивание

$$f(\xi) : \bigsqcup_{x \in X} X_x \to \{0, 1\},$$

где $\forall\, x \in X \;\; f(\xi)$ отображает каждую переменную из X_x в соответствующий бит значения $\xi(x)$.

Будем говорить, что BDD $\Delta \in \Delta_{\tilde{X}}$ представляет множество $S \subseteq X^\bullet$, если эта BDD является представлением множества

$$f(S) = \{f(\xi) \mid \xi \in S\} \subseteq \tilde{X}^\bullet.$$

4.1.4 Редукция бинарных диаграмм решений

Редукцией BDD называется преобразование её в эквивалентную ей BDD меньшего размера. Существуют три операции редукции:

1. если BDD содержит пару v, v' вершин, таких, что $l(v) = l(v')$, и если v и v' нетерминальны, то концы выходящих из v и v' рёбер с одинаковыми метками совпадают, то можно

   $$\begin{array}{l}\text{удалить } v \text{ и выходящие из неё рёбра, и}\\ \text{рёбра, входящие в } v, \text{ перенаправить в } v',\\ \text{если } v \text{ – начальная вершина исходной BDD,}\\ \text{то начальной вершиной новой BDD будет } v',\end{array} \tag{4.1}$$

2. если BDD содержит вершину v, такую, что выходящие из v рёбра имеют один и тот же конец v', то можно сделать (4.1),

3. если BDD содержит недостижимую вершину (т.е. такую вершину, в которую не существует пути из начальной вершины), то можно удалить эту вершину и все связанные с нею рёбра,

Нетрудно доказать, что при применении каждой из описанных выше операций редукции к какой-либо BDD получившаяся BDD будет эквивалентна исходной BDD.

Редуцированием BDD Δ называется итеративный процесс применения описанных выше операций редукции к Δ (на каждом шаге итерации текущая операция применяется к результату предыдущего шага), до тех пор, пока их будет возможно применять.

BDD называется **нередуцируемой**, если к ней невозможно применить никакую из описанных выше операций редукции.

4.1.5 Представление булевых функций

Пусть $X = \{x_1, \ldots, x_k\} \subseteq \mathcal{X}_{\mathbf{B}}$.

С каждой BDD $\Delta \in \Delta_X$ связана булева функция

$$f_\Delta : X^\bullet \to \{0,1\},$$

сопоставляющая каждому означиванию $\xi \in X^\bullet$ значение $f_\Delta(\xi) \stackrel{\text{def}}{=} \Delta^\xi$.

С другой стороны, для каждой булевой функции $f : X^\bullet \to \{0,1\}$ существует BDD $\Delta_{(f)} \in \Delta_X$, такая, что $f_{\Delta_{(f)}} = f$ – это полное бинарное дерево глубины k, каждое ребро которого имеет метку 0 или 1, и

- корень которого помечен переменной x_1,
- вершины следующего уровня – переменной x_2, и т.д.,
- вершины уровня $k-1$ – переменной x_k, и
- каждая вершина v уровня k – значением $f(\xi)$, где $(\xi(x_1), \ldots, \xi(x_k))$ – последовательность меток ребер, образующих путь из корня в v.

4.1.6 Подстановка значений вместо переменных

Пусть Δ – BDD, и $b \in \{0,1\}$, тогда

- если $v \in \Delta$ – нетерминальная вершина, то запись v_b обозначает конец выходящего из v ребра с меткой b, и
- $\forall\, x \in \mathcal{X}_{\mathbf{B}}$ запись $\Delta(b/x)$ обозначает BDD, получаемую из Δ удалением всех вершин с меткой x и выходящих из них рёбер, причём перед удалением каждой такой вершины все входящие в неё рёбра перенаправляются в v_b. Если удаляемая вершина – начальная в Δ, то начальной вершиной в $\Delta(b/x)$ будет v_b.

Пусть заданы множество $X \subseteq \mathcal{X}_{\mathbf{B}}$ и функция $f : X^\bullet \to \{0,1\}$.

$\forall\, x \in \mathcal{X}_{\mathbf{B}}, \forall\, b \in \{0,1\}$ запись $f(b/x)$ обозначает функцию $X^\bullet \to \{0,1\}$, сопоставляющую каждому $\xi \in X^\bullet$ значение $f^{\xi[x:=b]}$, где

$$\xi[x := b](x) = b, \;\; \forall\, y \in X \setminus \{x\} \;\; \xi[x := b](y) = \xi(y).$$

Нетрудно доказать, что

$$\begin{aligned} &\forall\, f : X^\bullet \to \{0,1\}, \forall\, x \in \mathcal{X}_{\mathbf{B}} \quad f = \bigvee_{b \in \{0,1\}} ((x = b) \wedge f(b/x)), \\ &\forall\, \Delta \in \Delta_X, \forall\, x \in \mathcal{X}_{\mathbf{B}}, \forall\, b \in \{0,1\} \quad f_{\Delta(b/x)} = f_\Delta(b/x) \end{aligned} \tag{4.2}$$

4.2 Согласованность с порядком переменных в бинарных диаграммах решений

Пусть задано множество $X \subseteq \mathcal{X}_{\mathbf{B}}$, и на нём задан линейный порядок ρ.

BDD $\Delta \in \Delta_X$ называется **согласованной** с порядком ρ, если для каждого ребра из одной нетерминальной вершины $v \in \Delta$ в другую нетерминальную вершину v' имеет место неравенство $l(v) < l(v')$. Запись $\Delta_{X,\rho}$ обозначает совокупность всех $\Delta \in \Delta_X$, согласованных с ρ.

Пусть заданы множество означиваний $S \subseteq X^\bullet$ и линейный порядок ρ на X. Запись $size(S, \rho)$ обозначает наименьший возможный размер BDD $\Delta \in \Delta_{X,\rho}$, которая представляет S.

Значения $size(S, \rho)$ могут существенно различаться при разных ρ. Например, если $X = \{x_1, y_1, \ldots x_k, y_k\}$, и

$$S = \{\xi \in X^\bullet \mid \Big(\bigwedge_{i=1}^{k} (x_i = y_i) \Big)^\xi = 1\},$$

то $size(S, \rho) = \begin{cases} 3k + 2, \text{ если } \rho = (x_1 < y_1 < \ldots < x_k < y_k), \\ 3 \cdot 2^k - 1, \text{ если } \rho = (x_1 < \ldots < x_k < y_1 < \ldots < y_k). \end{cases}$

Порядок ρ на X называется **оптимальным** для представления S, если для любого порядка ρ' на X $size(S, \rho) \leq size(S, \rho')$. Задача проверки оптимальности выбранного порядка является NP–полной.

При выборе порядка на X для представления $S \subseteq X^\bullet$ рекомендуется располагать ближе друг к другу те переменные, которые связаны между собой по смыслу.

Теорема 6.

Пусть заданы множество $X \subseteq \mathcal{X}_{\mathbf{B}}$, линейный порядок ρ на X, и две нередуцируемые BDD $\Delta_1, \Delta_2 \in \Delta_{X,\rho}$.

Если Δ_1 и Δ_2 эквивалентны, то они изоморфны.

Доказательство. (Автор излагаемого ниже доказательства – студент Высшей Школы Экономики Антон Гнатенко.)

Для доказательства теоремы сначала докажем лемму.

Лемма.

Пусть $\Delta \in \Delta_{X,\rho}$ – нередуцируемая BDD. Свяжем с каждой вершиной $v \in \Delta$

- BDD Δ_v, отличающуюся от Δ только начальной вершиной: $\Delta_v^0 = v$, и

- функцию $f_v : X^\bullet \to \{0,1\} : \ \forall \xi \in X^\bullet \ \ f_v(\xi) = \Delta_v^\xi$.

Тогда $\forall\, v, v' \in \Delta$, если $f_v = f_{v'}$, то $v = v'$.

Доказательство леммы.

Будем доказывать утверждение $v = v'$ индукцией по числу $d(v, v')$, которое равно сумме длин максимальных путей из v и v' в терминальную вершину.

Рассмотрим отдельно следующие случаи.

1. v и v' – терминальные вершины.
   Из $f_v = f_{v'}$ следует, что $l(v) = l(v')$. Т.к. Δ нередуцируема, то это возможно только если $v = v'$.
2. Одна из вершин v, v' нетерминальная, другая - терминальная.
   Пусть, например, v – нетерминальная, v' – терминальная.
   В этом случае функция f_v – константа, равная $l(v')$, и все пути из v ведут в v'. Пусть π – путь из v в v' максимальной длины, и v'' – последняя нетерминальная вершина на этом пути. Оба ребра, выходящие из v'', ведут в v' (т.к. иначе имеем противоречие с максимальностью π), но это невозможно по причине нередуцируемости Δ.
3. v и v' – нетерминальные вершины.
   Обозначим $x = l(v)$, $y = l(v')$, v_b и v'_b – концы ребер с меткой b ($b = 0, 1$), выходящих из v и v', соответственно.
   Предположим, что $x \neq y$. Пусть, например, $x < y$. В этом случае функция $f_v = f_{v'}$ не зависит существенно от x, поэтому $f_{v_0} = f_{v_1} = f_v = f_{v'}$. Имеем пары вершин (v_b, v'), такие, что $f_{v_b} = f_{v'}$, и $d(v_b, v') < d(v, v')$ ($b = 0, 1$). По и.п. (индуктивному предположению) верны равенства $v_0 = v'$ и $v_1 = v'$, откуда следует, что $v_0 = v_1$, что противоречит нередуцируемости Δ.
   Таким образом, $x = y$.
   Предположим, что $v \neq v'$.
   Если бы были верны равенства $v_0 = v'_0$ и $v_1 = v'_1$, то к Δ можно было бы применить первое правило редукции, что противоречит нередуцируемости Δ, поэтому $\exists\, b \in \{0,1\} : v_b \neq v'_b$.
   Нетрудно видеть, что $f_{v_b} = f_{v'_b}$ и $d(v_b, v'_b) < d(v, v')$, откуда по и.п. следует, равенство $v_b = v'_b$, которое по предположению неверно.
   Таким образом, доказано, что предположение $v \neq v'$ неверно.
   ■

Переходим к доказательству теоремы.

Будем вести доказательство индукцией по $|X|$.

Если $|X| = 0$, то $\Delta_1 = \Delta_1^0$, $\Delta_2 = \Delta_2^0$, и $l(\Delta_1^0) = l(\Delta_2^0)$. Такие BDD очевидно изоморфны.

Пусть $|X| = n > 0$, $X = \{x_1, \ldots, x_n\}$, и $x_1 < \ldots < x_n$.

Отдельно рассматриваем случаи, когда:

- одна из вершин Δ_1^0, Δ_2^0, терминальна,
- Δ_1^0 и Δ_2^0 нетерминальны и имеют разные метки,
- Δ_1^0 и Δ_2^0 нетерминальны и имеют одинаковые метки.

В первых двух случаях, рассуждая аналогично тому, как в лемме, получаем, что либо Δ_1 и Δ_2 изоморфны, либо одна из них редуцируема, что по предположению невозможно.

Рассмотрим третий случай: $l(\Delta_1^0) = l(\Delta_2^0)$.

Если $l(\Delta_1^0) = l(\Delta_2^0) \neq x_1$, то $\Delta_1, \Delta_2 \in \Delta_{X\setminus\{x_1\}}$, откуда по и.п. получаем доказываемое утверждение.

Пусть $l(\Delta_1^0) = l(\Delta_2^0) = x_1$, и $\forall\, i = 1, 2$

- v_{i0} и v_{i1} – концы рёбер с началом в Δ_i^0, и
- Δ_{i0} и Δ_{i1} – BDD, состоящие из тех вершин BDD Δ_i, которые достижимы из v_{i0} и v_{i1} соответственно, $\Delta_{i0}^0 = v_{i0}$ и $\Delta_{i1}^0 = v_{i1}$.

Нетрудно видеть, что $\forall\, b = 0, 1\ \ \Delta_{1b}, \Delta_{2b} \in \Delta_{X\setminus\{x_1\},\rho}$ – нередуцируемые эквивалентные BDD. Следовательно, по и.п. эти BDD изоморфны, т.е. $\exists\, \varphi_b : \Delta_{1b} \to \Delta_{2b} : \varphi(v_{1b}) = v_{2b}$ $(b = 0, 1)$, и выполнены другие свойства φ_b из определения изоморфности BDD (пункт 4.1.2).

Докажем, что $\forall\, v \in \Delta_{10} \cap \Delta_{11}\ \ \varphi_0(v) = \varphi_1(v)$. Согласно лемме, для этого достаточно доказать равенство функций $f_{\varphi_0(v)}$ и $f_{\varphi_1(v)}$. Обозначим записями Δ_{1v} и $\Delta_{2\varphi_b(v)}$ $(b = 0, 1)$ BDD, состоящие из тех вершин BDD Δ_1 и Δ_2, которые достижимы из v и $\varphi_b(v)$ соответственно, $\Delta_{1v}^0 = v$ и $\Delta_{2\varphi_b(v)}^0 = v_{\varphi_b(v)}$. $\forall\, b = 0, 1$ ограничение φ_b на Δ_{1v} задает изоморфизм между Δ_{1v} и $\Delta_{2\varphi_b(v)}$, откуда нетрудно получить равенства $f_v = f_{\varphi_0(v)}$ и $f_v = f_{\varphi_1(v)}$, откуда следует желаемое равенство $f_{\varphi_0(v)} = f_{\varphi_1(v)}$.

Искомое отображение $\varphi : \Delta_1 \to \Delta_2$ определяется так:

$$\varphi(\Delta_1^0) \stackrel{\text{def}}{=} \Delta_2^0, \quad \forall\, b = 0, 1, \forall\, v \in \Delta_{1b}\ \ \varphi(v) \stackrel{\text{def}}{=} \varphi_b(v).$$

Корректность данного определения следует из приведенных выше рассуждений. Нетрудно проверить, что φ обладается свойствами, изложенными в определении изоморфности BDD в пункте 4.1.2. ■

4.3 Алгебраические операции на бинарных диаграммах решений

4.3.1 Булевы операции

На BDD можно определить булевы операции: нульарные операции (т.е. константы) 0 и 1, унарная операция отрицания $^-$, бинарные операции $\wedge$ и $\vee$, причем данные операции будут согласованы с операциями на булевых функциях, соответствующих BDD, т.е. будут удовлетворять условиям

$$f_0 = 0, \quad f_1 = 1, \quad f_{\overline{\Delta}} = \overline{f_\Delta}, \quad f_{\Delta_1 \wedge \Delta_2} = f_{\Delta_1} \wedge f_{\Delta_2}, \quad \text{и т.д.}$$

Данные операции определяются следующим образом.

- BDD, соответствующая булевой константе 0 или 1, обозначается так же, как эта константа, и состоит из одной вершины с меткой 0 или 1 соответственно.
- Для каждой BDD Δ BDD $\overline{\Delta}$ получается из Δ заменой меток терминальных вершин: 0 заменяется на 1, а 1 – на 0.
- Пусть заданы множество $X \subseteq \mathcal{X}_{\mathbf{B}}$, линейный порядок ρ на X, и пара BDD $\Delta_1, \Delta_2 \in \Delta_{X,\rho}$.

  Конъюнкция $\Delta_1 \wedge \Delta_2 \in \Delta_{X,\rho}$ определяется рекурсивно следующим образом.

  - Если Δ_1 или Δ_2 является константой, то $\Delta_1 \wedge \Delta_2$ – или константа, или совпадает либо с одной из Δ_i (в соответствии с определением конъюнкции).
  - Пусть Δ_1 и Δ_2 не константы. Обозначим $x = \Delta_1^0$, $y = \Delta_2^0$.
    - Если $x = y$, то $(\Delta_1 \wedge \Delta_2)^0$ имеет метку x, из неё выходит ребро с меткой b в $(\Delta_1(b/x) \wedge \Delta_2(b/x))^0$ (где $b = 0, 1$).
    - Если $x < y$, то $(\Delta_1 \wedge \Delta_2)^0$ имеет метку x, из неё выходит ребро с меткой b в $(\Delta_1(b/x) \wedge \Delta_2)^0$ (где $b = 0, 1$).
    - Если $y < x$, то $\Delta_1 \wedge \Delta_2$ определяется аналогично.
  - дизъюнкция $\Delta_1 \vee \Delta_2$ определяется аналогично (в вышеприведенном определении все символы $\wedge$ заменяются на $\vee$).

Заметим, что вспомогательные BDD $\Delta_i(b/x)$, участвующие в определении $\Delta_1 \wedge \Delta_2$ и $\Delta_1 \vee \Delta_2$, являются подграфами Δ_1 и Δ_2, и полностью

определяются теми вершинами Δ_1 и Δ_2, которые являются начальными в этих 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 $O(|\Delta_1| \cdot |\Delta_2|)$.

4.3.2 Произведение

Пусть заданы BDD $\Delta_1 \in \Delta_{X_1}$ и $\Delta_2 \in \Delta_{X_2}$, которые согласованы с заданным линейным порядком ρ на $X_1 \cup X_2$.

Произведение $\Delta_1 \underset{X}{\wedge} \Delta_2$, где $X = \{x_1, \ldots, x_k\} = X_1 \cap X_2$ – это BDD из $\Delta_{(X_1 \cup X_2) \setminus X, \rho}$, эквивалентная BDD

$$\exists\, x_1 \ldots \exists\, x_k (\Delta_1 \wedge \Delta_2),$$

где для каждой BDD Δ запись $\exists\, x\, \Delta$ обозначает BDD $\Delta(0/x) \vee \Delta(1/x)$.

Ниже излагается рекурсивная программа **Prod**, которая по паре аргументов (Δ_1, Δ_2) указанного выше вида возвращает результат $\Delta_1 \underset{X}{\wedge} \Delta_2$.

В данной программе используется вспомогательная переменная $Cache$, в которой хранятся тройки вида $(\Delta_1, \Delta_2, \mathbf{Prod}(\Delta_1, \Delta_2))$.

Программа **Prod** имеет следующий вид:

Prod$(\Delta_1, \Delta_2) =$

1. если $\Delta_1 = 0$ или $\Delta_2 = 0$, то **return** 0
2. если $\Delta_1 = 1$ и $\Delta_2 = 1$, то **return** 1
3. если $(\Delta_1, \Delta_2, \Delta) \in Cache$ то **return** Δ
4. иначе выполняем следующую последовательность действий:
   (a) $x :=$ максимальная (в смысле указанного выше линейного порядка ρ) из переменных, входящих в Δ_1 или Δ_2,
   (b) $\Delta_{xb} := \mathbf{Prod}(\Delta_1(b/x), \Delta_2(b/x)) \quad (b = 0, 1)$
   (c) $\Delta := \begin{cases} \Delta_{x0} \vee \Delta_{x1}, \text{ если } x \in X, \\ (\Delta_{(x)} \wedge \Delta_{x1}) \vee (\Delta_{(\bar{x})} \wedge \Delta_{x0}), \text{ если } x \notin X, \end{cases}$
   (d) добавляем $(\Delta_1, \Delta_2, \Delta)$ в $Cache$
   (e) **return** Δ

В наихудшем случае сложность этого алгоритма экспоненциальна.

Теорема 7.

BDD $\mathbf{Prod}(\Delta_1, \Delta_2)$, вычисляемая описанным выше алгоритмом, эквивалентна BDD $\Delta_1 \underset{X}{\wedge} \Delta_2$.

Доказательство.

Очевидно, что достаточно доказать следующее утверждение: если $\Delta_{xb} = \Delta_1(b/x) \underset{X}{\wedge} \Delta_2(b/x))$, то BDD Δ, вычисляемая в пункте (4c) описанного выше алгоритма, эквивалентна $\Delta_1 \underset{X}{\wedge} \Delta_2$.

Для этого достаточно доказать совпадение булевых функций, соответствующих рассматриваемым BDD.

Обозначим символами f и g функции $f_{\Delta_1 \underset{X}{\wedge} \Delta_2}$ и $f_{\Delta_1} \wedge f_{\Delta_2}$, соответственно. В этих обозначениях

$$\begin{aligned} f &= \bigvee_{(b_1 \ldots b_k) \in \{0,1\}^k} g(b_1/x_1) \ldots (b_k/x_k) \\ f_{\Delta_{xb}} &= \bigvee_{(b_1 \ldots b_k) \in \{0,1\}^k} g(b/x)(b_1/x_1) \ldots (b_k/x_k) \end{aligned} \tag{4.3}$$

Требуется доказать, что $\forall\, x \in \mathcal{X}_{\mathbf{B}}$

$$\left.\begin{aligned} x \in X \;&\Rightarrow\; f = f_{\Delta_{x0}} \vee f_{\Delta_{x1}} \\ x \notin X \;&\Rightarrow\; f = ((x = 1) \wedge f_{\Delta_{x1}}) \vee ((x = 0) \wedge f_{\Delta_{x0}}) \end{aligned}\right\} \tag{4.4}$$

(4.4) можно переписать в виде

$$\left.\begin{aligned} x \in X \;&\Rightarrow\; f = \bigvee_{b \in \{0,1\}} f_{\Delta_{xb}} \\ x \notin X \;&\Rightarrow\; f = \bigvee_{b \in \{0,1\}} ((x = b) \wedge f_{\Delta_{xb}}) \end{aligned}\right\} \tag{4.5}$$

Согласно первому утверждению в (4.2), $\forall\, x \in \mathcal{X}_{\mathbf{B}}$

$$g = \bigvee_{b \in \{0,1\}} ((x = b) \wedge g(b/x)). \tag{4.6}$$

Обозначим последовательность подстановок $(b_1/x_1) \ldots (b_k/x_k)$ символом θ. Применяя θ к обоим частям (4.6), получаем

$$x \notin X \;\Rightarrow\; g\theta = \bigvee_{b \in \{0,1\}} ((x = b) \wedge g(b/x))\theta. \tag{4.7}$$

Кроме того, нетрудно видеть, что

$$x \in X \;\Rightarrow \bigvee_{(b_1 \ldots b_k) \in \{0,1\}^k} g\theta = \bigvee_{(b_1 \ldots b_k) \in \{0,1\}^k} \bigvee_{b \in \{0,1\}} g(b/x)\theta. \tag{4.8}$$

Из (4.7) и (4.8), учитывая (4.3), получаем (4.5). ■

4.4 Решение задачи MC-CTL при помощи бинарных диаграмм решений

Напомним, что одна из форм задачи MC-CTL заключается в вычислении по заданным СП $\Sigma = (S, R, L)$ и формуле $A \in Fm_{CTL}$ множества $S(A)$.

В том случае, когда

- S имеет вид $X^\bullet$, где $X \subseteq \mathcal{X}_{\mathbf{B}}$ – конечное множество переменных, и
- компоненты СП Σ определяются формулами из Fm_X и $Fm_{X \sqcup X'}$,

данная задача м.б. решена методом, связанным с понятием неподвижной точки (изложенным в параграфе 3.3), с использованием BDD.

Идея излагаемого ниже подхода заключается в том, чтобы представлять каждое подмножество $S' \subseteq S = X^\bullet$, возникающее в процессе вычисления $S(A)$, не явным образом, а в виде BDD, представляющей S' (в соответствии с тем, как это было определено в пункте 4.1.3).

Как и раньше, без ограничения общности мы можем предполагать, что в A могут входить лишь CTL–операторы вида **EX**, **EG**, **EU**. Вычисление множества $S(A)$ заключается в вычислении множеств вида $S(B)$ для всех подформул формулы A.

Предположим, что

- для каждого атомарного утверждения $p \in \mathcal{P}$, входящего в A, задана BDD $\Delta(p) \in \Delta_X$, представляющая множество $S(p)$, и
- задана BDD $\Delta(R) \in \Delta_{X \sqcup X'}$, представляющая отношение перехода $R \subseteq X^\bullet \times X^\bullet$,
- $X = \{x_1, \ldots, x_k\}$, и все упомянутые в этом абзаце BDD согласованы с подрядком $x_1 < \ldots < x_k < x'_1 < \ldots < x'_k$.

Тогда для вычисления BDD $\Delta(A)$, представляющей множество $S(A)$, можно использовать излагаемый ниже рекурсивный алгоритм, Данный алгоритм сводит вычисление BDD $\Delta(A)$ к вычислению BDD $\Delta(B)$, представляющих множества вида $S(B)$, где B – подформула формулы A.

1. Если $A = p \in \mathcal{P}$, то BDD $\Delta(p)$ предполагается заданной.
2. Если A – булева комбинация, то $\Delta(A)$ вычисляется путем применения соответствующих булевых операций к BDD, соответствующим компонентам этой булевой комбинации:

$$\Delta(\top) = 1, \;\; \Delta(\bot) = 0,$$
$$\Delta(\bar{B}) = \overline{\Delta(B)}, \;\; \Delta(B \wedge C) = \Delta(B) \wedge \Delta(C), \text{ и т.д.}$$

3. Пусть $A = \mathbf{EX}B$, и имеется BDD $\Delta(B)$, представляющая $S(B)$.

   Из определения множества

   $$S(\mathbf{EX}B) = R^{-1}(S(B)) = \{\xi \in X^\bullet \mid R(\xi) \cap S(B) \neq \emptyset\}$$

   следует, что $\forall\, \xi \in X^\bullet$

   $$\xi \in S(\mathbf{EX}B) \;\; \Leftrightarrow \;\; \exists\, \xi' \in X^\bullet : (\xi, \xi') \in R, \;\; \xi' \in S(B). \tag{4.9}$$

   Соотношение (4.9) можно переписать в терминах BDD, представляющих упомянутые в нём множества:

   $$\Delta(\mathbf{EX}B)^\xi = 1 \;\; \Leftrightarrow \;\; \exists\, \xi' \in X^\bullet : \Delta(R)^{\xi \sqcup \xi'} = 1 \;\text{ и }\; \Delta(B)^{\xi'} = 1. \tag{4.10}$$

   Истинность (4.10) $\forall\, \xi \in X^\bullet$ равносильна эквивалентности

   $$\Delta(\mathbf{EX}B) \;=\; \exists\, x'_1 \ldots \exists\, x'_k \, \Big(\Delta(R) \wedge \Delta(B)(x'_1, \ldots, x'_k)\Big)$$

   где $X = \{x_1, \ldots, x_k\}$, и $\Delta(B)(x'_1, \ldots, x'_k)$ получается из $\Delta(B)$ заменой каждой переменной $x_i \in X$ на соответствующую переменную x'_i $(i = 1, \ldots, x_k)$. Будем обозначать данную BDD записью $\Delta(B)'$.

   Таким образом, вычисление $\Delta(\mathbf{EX}B)$ сводится к вычислению произведения $\Delta(R) \underset{X'}{\wedge} \Delta(B)'$. Будем обозначать его записью $\mathbf{EX}(\Delta(B))$.

4. Если $A = \mathbf{EU}(B, C)$ или $\mathbf{EG}B$, то $\Delta(A)$ вычисляется согласно (3.6):

   $$\begin{aligned} &\Delta(\mathbf{EU}(B, C)) = \mu x.\Big(\Delta(C) \vee (\Delta(B) \wedge \mathbf{EX}(x))\Big), \\ &\Delta(\mathbf{EG}B) = \nu x.\Big(\Delta(B) \wedge \mathbf{EX}(x)\Big), \end{aligned} \tag{4.11}$$

   где $\forall\, \Delta \in \Delta_X$ запись $\mathbf{EX}(\Delta)$ обозначает произведение $\Delta(R) \underset{X'}{\wedge} \Delta'$, и Δ' определяется аналогично $\Delta(B)'$ в предыдущем пункте.

   Неподвижные точки в (4.11) вычисляются путем построения соответствующих последовательностей BDD, начиная с 0 или 1. После вычисления каждого нового члена последовательности производится его редукция и проверка его эквивалентности предыдущему члену этой последовательности. Согласно теореме 6, данная проверка сводится к проверке изоморфности этих членов. Построение последовательности заканчивается, когда новый построенный её член изоморфен предыдущему её члену.

4.5 Оптимизирующие преобразования при построении бинарных диаграмм решений

Если СП Σ строится по программной системе согласно определениям из пункта 2.3.3, то её отношение перехода определяется формулой $\bigvee_{i=1}^{n} R_i$. Поскольку квантор $\exists$ коммутирует с дизъюнкцией, то $\forall\, \Delta \in \Delta_X$ для вычисления $\mathbf{EX}(\Delta)$ можно вычислять BDD, определяемую выражением

$$\bigvee_{i=1}^{n} \exists\, x'_1 \ldots \exists\, x'_k \left(\Delta_{(R_i)} \wedge \Delta(x'_1, \ldots, x'_k)\right). \tag{4.12}$$

$\forall\, i = 1, \ldots, n$ обозначим записью C_i i-й дизъюнктивный член в (4.12). Можно оптимизировать выражение C_i (т.е. преобразовать его в такое выражение $\tilde{C}_i$, по которому соответствующая BDD вычисляется быстрее) следующим образом. Пусть $R_i = \bigwedge_{j=1}^{n_i} R_{ij}$, где некоторые из R_{ij} имеют вид $x' = x$ или $x' = b$, где $b \in \{0, 1\}$, или не содержат переменных из X'. Тогда можно выполнить следующие преобразования выражения C_i.

- Каждый конъюнктивный член в R_i вида $x' = x$ удаляется, и компонента $\Delta(x'_1, \ldots, x'_k)$ выражения C_i преобразуется путем замен всех вхождений переменной x' на x.

  Обозначим записью $\tilde{\Delta}$ результат этих преобразований.

- Каждый конъюнктивный член в R_i вида $x' = b$, где $b \in \{0, 1\}$, удаляется, компонента $\tilde{\Delta}$ преобразуется в $\tilde{\Delta}(b/x')$.

  Будем обозначать получившуюся BDD той же записью $\tilde{\Delta}$.

- Каждый конъюнктивный член R_{ij} в R_i, не содержащий переменных из X', удаляется из R_i, и в выражении C_i перед кванторами существования приписывается конъюнктивный член $\Delta_{R_{ij}}$.

  Обозначим записью $\tilde{R}_i$ конъюнкцию оставшихся членов в R_i, и записью $\tilde{C}_i$ – преобразованное выражение C_i.

- Если какая-либо из переменных $x' \in X'$ не входит в $\tilde{R}_i$ и $\tilde{\Delta}$, то фрагмент $\exists\, x'$ удаляется из $\tilde{C}_i$.

Нетрудно видеть, что C_i и $\tilde{C}_i$ определяют эквивалентные BDD.

Глава 5

Model checking на основе темпоральной логики LTL

5.1 Формулы темпоральной логики LTL

Совокупность **формул** темпоральной логики LTL (Linear Temporal Logic), называемых также **LTL-формулами**, обозначается записью Fm_{LTL} и определяется следующим образом:

- $\mathcal{P} \subseteq Fm_{LTL}$,
- Fm_{LTL} замкнуто относительно булевых комбинаций,
- $\forall\, B \in Fm_{LTL}$ записи $\mathbf{X}B$, $\mathbf{F}B$, $\mathbf{G}B$ являются LTL-формулами,
- $\forall\, B, C \in Fm_{LTL}$ запись $\mathbf{U}(B, C)$ является LTL-формулой.

Жирные символы ($\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{U}$) в LTL-формулах называются **темпоральными операторами**.

Пусть задана СП Σ, и $\pi = (s_0, s_1, \ldots)$ – путь в Σ. Для каждой LTL-формулы B **значение** $\pi(B) \in \{0, 1\}$ формулы B на пути π определяется ниже индуктивно. $\forall\, i \geq 0$ будем обоначать записью π_i хвост пути π, начинающийся с позиции i: $\pi_i = (s_i, s_{i+1}, \ldots)$.

- Если $B = p \in \mathcal{P}$, то $\pi(B) \stackrel{\text{def}}{=} \begin{cases} s_0(p) = L_\Sigma(s_0, p), \text{ если } p \in \mathcal{P}_\Sigma, \\ 0, \text{ если } p \notin \mathcal{P}_\Sigma. \end{cases}$
- Значения булевых комбинаций определяются стандартно:

$$\pi(\top) = 1, \;\; \pi(\bot) = 0,$$
$$\pi(\bar{B}) = \overline{\pi(B)}, \;\; \pi(B \wedge C) = \pi(B) \wedge \pi(C), \text{ и т.д.}$$

- Значения LTL-формул, начинающихся с темпорального оператора, определяются следующим образом:
  - $\pi(\mathbf{X}B) \stackrel{\text{def}}{=} \pi_1(B)$,
  - $\pi(\mathbf{F}B) = 1$, если $\exists\, i \geq 0 :\ \pi_i(B) = 1$,
  - $\pi(\mathbf{G}B) = 1$, если $\forall\, i \geq 0 \quad \pi_i(B) = 1$,
  - $\pi(\mathbf{U}(B, C)) = 1$, если либо $\pi(C) = 1$, либо $\exists\, i > 0$:

$$\pi_i(C) = 1, \forall\, j < i \quad \pi_j(B) = 1.$$

Будем называть LTL-формулы A и B **эквивалентными**, если для каждой СП Σ, $\forall\, \pi \in \Pi_\Sigma$ верно равенство $\pi(A) = \pi(B)$. Запись $A = B$, где $A, B \in Fm_{LTL}$, означает что A и B эквивалентны.

Нетрудно доказать, что верны соотношения

$$\mathbf{F}B = \mathbf{U}(\top, B), \quad \overline{\mathbf{X}B} = \mathbf{X}\bar{B}, \quad \overline{\mathbf{G}B} = \mathbf{F}\bar{B}.$$

Следовательно, для любой LTL–формулы существует эквивалентная ей LTL–формула, в которую входят только связки $^{-}, \vee, \mathbf{X}, \mathbf{U}$.

Кроме того, нетрудно доказать, что $\forall\, B, C \in Fm_{LTL}$ верно равенство

$$\mathbf{U}(B, C) = C \vee \Big(B \wedge \mathbf{X}\mathbf{U}(B, C)\Big). \tag{5.1}$$

Ниже будет использоваться соглашение: знакосочетание $\mathbf{U}(B, C)$ будет обозначаться символом $*$. В этих обозначениях (5.1) записывается в виде равенства $* = C \vee \Big(B \wedge \mathbf{X} * \Big)$.

5.2 Квантифицированные LTL-формулы

Совокупность **квантифицированных LTL–формул** обозначается записью QFm_{LTL} и определяется индуктивно:

- записи вида $\mathbf{A}B$ или $\mathbf{E}B$, где B – LTL-формула, являются квантифицированными LTL–формулами,
- QFm_{LTL} замкнуто относительно булевых комбинаций.

Пусть задана СП $\Sigma = (S, \ldots)$. $\forall\, s \in S, \forall\, A \in QFm_{LTL}$ **значение** A в s обозначается знакосочетанием $s(A)$ и определяется следующим образом:

$$s(\mathbf{A}B) = 1, \ \text{ если } \ \forall\, \pi \in \Pi_s \ \ \pi(B) = 1,$$
$$s(\mathbf{E}B) = 1, \ \text{ если } \ \exists\, \pi \in \Pi_s : \pi(B) = 1,$$

и значения булевых комбинаций определются стандартным образом.

Будем называть формулы $A, B \in QFm_{LTL}$ **эквивалентными**, если для каждого состояния s произвольной СП верно равенство $s(A) = s(B)$. Если формулы $A, B \in QFm_{LTL}$ эквивалентны, то будем обозначать этот факт знакосочетанием $A = B$.

Из этого определения следует, что

$$\forall\, B \in Fm_{LTL} \quad \overline{\mathbf{A}B} = \mathbf{E}\overline{B}. \tag{5.2}$$

Аналогично определяется эквивалентность между CTL–формулами и квантифицированными LTL–формулами. Можно доказать, что

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

5.3 Задачи model checking для LTL

5.3.1 Система переходов Σ_A

Ниже предполагаем, что каждая рассматриваемая LTL–формула содержит связки только из множества $\{^{-}, \vee, \mathbf{X}, \mathbf{U}\}$.

Будем использовать следующие понятия и обозначения:

- $\forall\, A \in Fm_{LTL}$ запись $\langle A\rangle$ обозначает наименьшее (по отношению включения) множество формул, удовлетворяющее условиям:
  - $\langle A\rangle$ содержит все подформулы формулы A,
  - если $* \in \langle A\rangle$, то $\mathbf{X}* \in \langle A\rangle$,

  множество $\langle A\rangle$ называется **замыканием** формулы A,
- множество $\mathcal{P} \cap \langle A\rangle$ будем обозначать записью $\mathcal{P}_A$.

С каждой формулой $A \in Fm_{LTL}$ связана СП

$$\Sigma_A = (S_A, R_A, L_A, S_A^0, \mathcal{F}_A)$$

определяемая следующим образом:

- S_A состоит из функций $f : \langle A\rangle \to \{0,1\}$, таких, что

$$\begin{array}{l}\forall\, \bar{B} \in \langle A\rangle \quad f(\bar{B}) = \overline{f(B)},\\ \forall\, (B \vee C) \in \langle A\rangle \quad f(B \vee C) = f(B) \vee f(C)\\ \forall\, * \in \langle A\rangle \quad f(*) = f(C) \vee \Big(f(B) \wedge f(\mathbf{X}*)\Big)\end{array} \tag{5.3}$$

- R_A состоит из всех пар $(f, f') \in S_A \times S_A$, таких, что

$$\forall\, \mathbf{X}B \in \langle A\rangle \quad f(\mathbf{X}B) = f'(B), \tag{5.4}$$

- $\mathcal{P}_{\Sigma_A} = \mathcal{P}_A, \forall\, f \in S_A, \forall\, p \in \mathcal{P}_A \quad L_A(f,p) = f(p)$,
- $S_A^0 \stackrel{\text{def}}{=} \{f \in S_A \mid f(A) = 1\}$,
- $\mathcal{F}_A = \{F_* \mid * \in \langle A\rangle\}$, где для каждой формулы из $\langle A\rangle$ вида $*$

$$F_* \stackrel{\text{def}}{=} \{f \in S_A \mid f(*) \le f(C)\}. \tag{5.5}$$

Лемма 1.
Для каждого пути $\varphi = (f_0, \ldots)$ в СП Σ_A

$$\varphi - \text{fair} \quad \Leftrightarrow \quad \forall\, i \ge 0,\ \forall\, * \in \langle A\rangle \quad f_i(*) \le \bigvee_{j \ge i} f_j(C). \tag{5.6}$$

Доказательство.
Пусть левая часть (5.6) верна, а правая - нет, т.е. $\exists\, i \ge 0$, $\exists\, * \in \langle A\rangle$:

$$f_i(*) = 1, \forall\, j \ge i \quad f_j(C) = 0. \tag{5.7}$$

Из (5.7) следует, что

$$\begin{array}{l}\forall\, j \ge i \quad 1 = f_i(*) = f_i(C) \vee (f_i(B) \wedge f_i(\mathbf{X}*)) =\\ = f_i(B) \wedge f_i(\mathbf{X}*) = f_i(B) \wedge f_{i+1}(*),\end{array}$$

поэтому $f_{i+1}(*) = 1$, т.е. верно утверждение (5.7) с заменой i на $i+1$. Отсюда следует, что $f_{i+2}(*) = 1$, и т.д., т.е.

$$\forall\, j \ge i \quad f_j(*) = 1. \tag{5.8}$$

Т.к. путь φ – fair, то, в частности, $\exists\, j \ge i : f_j \in F_*$, т.е.

$$f_j(*) \le f_j(C). \tag{5.9}$$

(5.9) противоречит утверждению (5.8) и свойству $\forall\, j \geq i \quad f_j(C) = 0$.

Обратно, пусть правая часть (5.6) верна, а левая – нет, т.е. путь φ – не fair, что означает

$$\exists * \in \langle A \rangle,\ \exists\, i \geq 0 : \forall\, j \geq i \ \ f_j \notin F_*, \tag{5.10}$$

т.е. $\forall\, j \geq i \ \ f_j(*) = 1,\ f_j(C) = 0$, откуда следует, что

$$f_i(*) = 1, \quad \bigvee_{j \geq i} f_j(C) = 0, \tag{5.11}$$

что противоречит правой части (5.6). ■

5.3.2 Система переходов $\Sigma \times \Sigma_A$

Пусть заданы СП $\Sigma = (S_\Sigma, R_\Sigma, L_\Sigma)$ и LTL–формула A.

Запись $\Sigma \times \Sigma_A$ обозначает СП $(S, R, L, S^0, \mathcal{F})$, где

- $S = \{(s, f) \in S_\Sigma \times S_A \mid \forall\, p \in \mathcal{P}_A \ \ s(p) = f(p)\}$,
- $R = \{((s, f), (s', f')) \mid (s, s') \in R_\Sigma, (f, f') \in R_A\}$,
- $\forall\, (s, f) \in S, \forall\, p \in \mathcal{P}_A \quad L((s, f), p) = s(p) = f(p)$,
- $S^0 = S^0_\Sigma \times S^0_A$,
- $\mathcal{F} = \{S_\Sigma \times F_* \mid * \in \langle A \rangle\}$, где множества F_* определяются в (5.5).

Нетрудно видеть, что путь ψ в $\Sigma \times \Sigma_A$ – fair тогда и только тогда, когда последовательность его вторых компонентов – fair путь в Σ_A.

Каждый путь π в Σ определяет путь $\varphi_\pi = (f_0, \ldots)$ в Σ_A, где

$$\forall\, A' \in \langle A \rangle, \quad \forall\, i \geq 0 \qquad f_i(A') = \pi_i(A'). \tag{5.12}$$

Нетрудно видеть, что $\forall\, i \geq 0 \ \ f_i \in S_A$ и $(f_i, f_{i+1}) \in R_A$.

Лемма 2. φ_π – fair путь в Σ_A.

Доказательство.

Если φ_π не fair, то верно (5.10) и (5.11), откуда, учитывая (5.12) получаем:

$$\exists * \in \langle A \rangle,\ \exists\, i \geq 0 : \forall\, j \geq i \quad \pi_j(*) = 1,\ \pi_j(C) = 0. \tag{5.13}$$

Но т.к. $\pi_i(*) = 1$, то $\exists\, j \geq i : \pi_j(C) = 1$, что противоречит (5.13). ■

$\forall\, \pi \in \Pi_\Sigma$ обозначим записью ψ_π путь в $\Sigma \times \Sigma_A$, имеющий вид

$$((s_0, f_0), (s_1, f_1), \ldots), \quad \text{где } \pi = (s_0, s_1, \ldots),\ \varphi_\pi = (f_0, f_1, \ldots). \tag{5.14}$$

Из леммы 2 следует, что путь ψ_π – fair.

Лемма 3.
Соответствие $\pi \mapsto \psi_\pi$ из Π_Σ в $\Pi_{\Sigma\times\Sigma_A}$ биективно.

Доказательство.
Инъективность данного соответствия очевидна.
Докажем его сюръективность, т.е. докажем, что для каждого пути

$$\psi = ((s_0, f_0), (s_1, f_1), \ldots) \in \Pi_{\Sigma\times\Sigma_A}$$

верно утверждение (5.12), где $\pi = (s_0, s_1, \ldots)$. Доказательство проведем индукцией по структуре формулы A'. Предполагаем, что для собственных подформул формулы A' это утверждение верно.

1. $A' = p \in \mathcal{P}$: в этом случае, по определению состояний СП $\Sigma \times \Sigma_A$,

$$f_i(A') = f_i(p) = s_i(p) = \pi_i(p) = \pi_i(A'),$$

2. A' имеет вид булевой комбинации: в этом случае доказательство (5.12) не представляет особой сложности,

3. $A' = \mathbf{X}B$: $f_i(\mathbf{X}B) = f_{i+1}(B) = \pi_{i+1}(B) = \pi_i(\mathbf{X}B)$,

4. $A' = *$: докажем равенство $f_i(*) = \pi_i(*)$.

   - Предположим, что $f_i(*) = 1$. Т.к. путь $(f_0, f_1, \ldots)$ – fair, то по лемме 1 верна правая часть (5.6), поэтому из $f_i(*) = 1$ следует, что $\exists\, j \geq i : f_j(C) = 1$. Будем считать, что j – наименьший индекс $\geq i$, такой, что $f_j(C) = 1$. Если $j > i$, то из последнего равенства в (5.3), где $f = f_i$, следует, что $f_i(B) = 1$, $f_{i+1}(*) = 1$. Если $j > i{+}1$, то аналогично получаем $f_{i+1}(B) = 1$, $f_{i+2}(*) = 1$, и т.д. Таким образом, имеем равенство

$$f_i(B) \wedge \ldots \wedge f_{j-1}(B) \wedge f_j(C) = 1. \tag{5.15}$$

     По и.п. все вхождения символа f в этом равенстве можно заменить на символ π, откуда по определению значения $\pi_i(*)$ следует, что $\pi_i(*) = 1$.

- Предположим, что $\pi_i(*) = 1$. По определению значения $\pi_i(*)$, отсюда следует что $\exists\, j \geq i$: верно равенство

$$\pi_i(B) \wedge \ldots \wedge \pi_{j-1}(B) \wedge \pi_j(C) = 1, \tag{5.16}$$

Используя и.п., из (5.16) можно получить (5.15), из которого нетрудно получить равенство $f_i(*) = 1$. ■

5.3.3 Первая задача model checking для LTL

Одна из задач **MC-LTL** имеет следующий вид: заданы СП $\Sigma = (S, R, L)$, состояние $s \in S$, и квантифицированная LTL–формула Q. Требуется вычислить значение $s(Q)$. Из определения понятия квантифицированной LTL–формулы и свойства (5.2) следует, что данная задача сводится к аналогичной задаче для случая, когда Q имеет вид $\mathbf{E}A$.

Согласно определению, равенство $s(\mathbf{E}A) = 1$ означает, что

$$\exists\, \pi \in \Pi_s :\ \pi(A) = 1. \tag{5.17}$$

Как было сказано выше, каждому пути $\pi \in \Pi_s$ соответствует путь

$$\psi_\pi \in \Pi_{(s,f_0)}, \tag{5.18}$$

где состояние $f_0 \in S_A$ определяется соотношением (5.12). Частным случаем этого соотношения является равенство $f_0(A) = \pi(A)$.

Теорема 8.

Для проверки свойства (5.17) можно использовать следующий метод: $\forall\, f \in S_A^0$ проверить существование пути

$$\psi \in \Pi_{(s,f)} \tag{5.19}$$

если для какого-либо $f \in S_A^0$ такой путь существует, то ответ положителен (т.е. свойство (5.17) верно), иначе – ответ отрицателен.

Доказательство.

- Если существует путь $\pi \in \Pi_s$, такой, что $\pi(A) = 1$, то упомянутое выше состояние $f_0 \in S_A$, определяемое этим путем, обладает свойством $f_0(A) = \pi(A) = 1$, т.е. $f_0 \in S_A^0$. Поскольку верно свойство (5.18), то, следовательно существует путь (5.19) (это будет путь ψ_π), т.е. этот метод даст положительный ответ.

- Обратно, предположим что описанный выше метод даст положительный ответ, т.е. $\exists f \in S_A^0$, $\exists \psi$: (5.19). По лемме 3, $\exists \pi \in \Pi_s$: $\psi = \psi_\pi$. В частности, $f(A) = \pi(A)$. Т.к. $f \in S_A^0$, то $f(A) = 1$, откуда следует, что $\pi(A) = 1$. ■

Задача проверки существования пути (5.19) может решаться различными способами. Один из них – рассматривать её как задачу MC-CTL, которая заключается в вычислении значения CTL–формулы $\mathbf{EG}\top$ в состоянии (s, f) СП $\Sigma \times \Sigma_A$. Эту задачу можно решать алгоритмом MC-CTL, основанным на использовании BDD, его сложность – $O(|\Sigma| \cdot 2^{|A|})$. Отметим, что результатом работы данного алгоритма является множество $S_{\Sigma\times\Sigma_A}(\mathbf{EG}\top)$, из которого может быть получено множество $S(\mathbf{E}A)$.

Существование пути ψ, удовлетворяющего условию (5.19), может быть проверено также с использованием следующей теоремы.

Теорема 9.
Для каждого состояния (s, f) СП $\Sigma \times \Sigma_A$ следующие условия эквивалентны:

1. существует путь ψ, удовлетворяющий условию (5.19),
2. существует конечный путь из (s, f) в некоторую SCC $\mathbf{C}$, которая является fair, т.е. удовлетворяет условию

$$\forall * \in \langle A \rangle \;\; \mathbf{C} \cap F_* \neq \emptyset, \quad \text{где } F_* \stackrel{\text{def}}{=} \{f \in S_A \mid f(*) \leq f(C)\}. \qquad (5.20)$$

Доказательство.
(1) $\Rightarrow$ (2): $\exists i$: $\psi_i = \inf(\psi) \subseteq$ SCC $\mathbf{C}$. Нетрудно видеть, что $\mathbf{C}$ удовлетворяет условию (5.20).
(2) $\Rightarrow$ (1): $\psi \stackrel{\text{def}}{=} \hat{\psi} \cdot \tilde{\psi}^\infty$, где $\hat{\psi}$ – конечный путь из (s, f) в $(s', f') \in \mathbf{C}$, и $\tilde{\psi}$ – цикл из (s', f') в (s', f'), содержащий все состояния из $\mathbf{C}$. ■

Можно доказать, что описанная выше задача MC-LTL PSPACE–полна.

5.3.4 Вторая задача model checking для LTL

Пусть заданы СП $\Sigma = (S, R, L, S^0)$ и LTL–формула A.
Вторая задача **MC-LTL** имеет следующий вид: доказать, что

$$\forall \pi \in \Pi_\Sigma^0 \quad \pi(A) = 1. \qquad (5.21)$$

Данная задача сводится к задаче, связанной с излагаемым в пункте 5.4 понятием **автомата Бюхи**. Для такого сведения мы сформулируем вспомогательные понятия и приведём связанные с ними утверждения.

Будем использовать следующие обозначения: пусть $A \in Fm_{LTL}$, тогда

- запись $[A]$ обозначает множество функций вида $g : \mathcal{P}_A \to \{0, 1\}$,
- запись $[A]^\infty$ обозначает множество всех бесконечных последовательностей элементов множества $[A]$,
- для произвольных СП Σ и пути $\pi = (s_0, \ldots) \in \Pi_\Sigma$ запись $\mathcal{L}(\pi)$ обозначает последовательность

$$(g_0, g_1, \ldots) \in [A]^\infty, \text{ где } \forall\, i \geq 0,\ \forall\, p \in \mathcal{P}_A \quad g_i(p) = \pi_i(p) = s_i(p).$$

Лемма 4.

$\forall\, A \in Fm_{LTL}$, $\forall$ СП Σ, $\forall\, \pi \in \Pi_\Sigma, \forall\, \varphi \in \Pi_{\Sigma_A}$

$$\mathcal{L}(\pi) = \mathcal{L}(\varphi) \quad \Leftrightarrow \quad \varphi_\pi = \varphi.$$

Доказательство.

Пусть π и φ имеют вид $(s_0, s_1, \ldots)$ и $(f_0, f_1, \ldots)$ соответственно.

- Импликация "$\Rightarrow$" следует из леммы 3 и из того, что путь

  $$((s_0, f_0), (s_1, f_1), \ldots)$$

  является fair путём в $\Sigma \times \Sigma_A$.
- Импликация "$\Leftarrow$", т.е. равенство $\mathcal{L}(\pi) = \mathcal{L}(\varphi_\pi)$, следует из определения 5.12 пути φ_π. ■

Будем обозначать множество $\{\mathcal{L}(\pi) \mid \pi \in \Pi^0_\Sigma\}$ (где Σ – произвольная СП) записью $\mathcal{L}(\Sigma)$.

Лемма 5.

Свойство (5.21) равносильно включению $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\Sigma_A)$.

Доказательство.

- Пусть верно (5.21). Докажем, что $\forall\, \pi \in \Pi^0_\Sigma$

  $$\mathcal{L}(\pi) \in \mathcal{L}(\Sigma_A). \tag{5.22}$$

  Согласно (5.21), из $\pi \in \Pi^0_\Sigma$ следует, что $\pi(A) = 1$. По лемме 4 верно равенство $\mathcal{L}(\pi) = \mathcal{L}(\varphi_\pi)$, поэтому для доказательства (5.22) нужно доказать, что $\varphi_\pi \in \Pi^0_{\Sigma_A}$, т.е. если φ_π имеет вид $(f_0, f_1, \ldots)$, то $f_0(A) = 1$. Из (5.12) следует, что $f_0(A) = \pi_0(A) = \pi(A) = 1$.

- Обратно, пусть $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\Sigma_A)$, т.е. $\forall\, \pi \in \Pi^0_\Sigma\ \mathcal{L}(\pi) \in \mathcal{L}(\Sigma_A)$.

  Докажем, что верно (5.21), т.е. $\forall\, \pi \in \Pi^0_\Sigma \quad \pi(A) = 1$.

  Из $\mathcal{L}(\pi) \in \mathcal{L}(\Sigma_A)$ следует, что $\exists\, \varphi \in \Pi^0_{\Sigma_A} : \mathcal{L}(\pi) = \mathcal{L}(\varphi)$. По лемме 4 из этого равенства следует, что $\varphi = \varphi_\pi$. Из (5.12) следует, что если $(f_0, f_1, \ldots) = \varphi = \varphi_\pi$, то $\pi(A) = \pi_0(A) = f_0(A) = 1$. ■

Теорема 10.
Свойство (5.21) равносильно равенству

$$\mathcal{L}(\Sigma) \cap \mathcal{L}(\Sigma_{\bar{A}}) = \emptyset \tag{5.23}$$

Доказательство.
Утверждение теоремы следует из леммы 5 и из соотношений

$$\begin{array}{l} \mathcal{L}(\Sigma_A) \cap \mathcal{L}(\Sigma_{\bar{A}}) = \emptyset, \\ \forall\, g = (g_0, g_1, \ldots) \in [A]^\infty \quad g \in \mathcal{L}(\Sigma_A) \ \text{ или } \ g \in \mathcal{L}(\Sigma_{\bar{A}}). \end{array} \tag{5.24}$$

Для доказательства этих соотношений введём следующие обозначения: если $g = (g_0, \ldots) \in [A]^\infty$, то

- запись Σ_g обозначает СП (S, R, L, S^0), где
  - $S = \{s_0, s_1, \ldots\}$, $R = \{(s_i, s_{i+1}) \mid i \geq 0\}$, $S^0 \stackrel{\text{def}}{=} \{s_0\}$,
  - $\forall\, i \geq 0, \forall\, p \in \mathcal{P}_A \ \ L(s_i, p) \stackrel{\text{def}}{=} g_i(p)$,
- символ π_g обозначает путь $(s_0, s_1, \ldots) \in \Pi_{\Sigma_g}$,
- записи $\varphi_{\pi_g} = (f_0, \ldots)$ и $\varphi'_{\pi_g} = (f'_0, \ldots)$ обозначают пути в Π_{Σ_A} и $\Pi_{\Sigma_{\bar{A}}}$ соответственно, соответствующие пути π_g.

Из (5.12) следуют равенства:

- $g = \mathcal{L}(\varphi_{\pi_g}) = \mathcal{L}(\varphi'_{\pi_g})$, т.к. $\forall\, i \geq 0,\ \forall\, p \in \mathcal{P}_A = \mathcal{P}_{\bar{A}}$

  $$g_i(p) = L(s_i, p) = (\pi_g)_i(p) = f_i(p) = f'_i(p),$$

- $f_0(A) = \pi_g(A)$, $f'_0(\bar{A}) = \pi_g(\bar{A})$, откуда следует равенство

  $$\overline{f_0(A)} = f'_0(\bar{A}). \tag{5.25}$$

1. Обоснуем первое соотношение в (5.24). Пусть оно неверно, т.е.

   $$\exists\, \varphi \in \Pi^0_{\Sigma_A},\ \exists\, \varphi' \in \Pi^0_{\Sigma_{\bar{A}}} :\ \mathcal{L}(\varphi) = \mathcal{L}(\varphi').$$

   Обозначим $g \stackrel{\text{def}}{=} \mathcal{L}(\varphi) = \mathcal{L}(\varphi')$. Т.к. $f_0 \in S^0_{\Sigma_A}$ и $f'_0 \in S^0_{\Sigma_{\bar{A}}}$, то $f_0(A) = 1$ и $f'_0(\bar{A}) = 1$, что противоречит (5.25).

2. Обоснуем второе соотношение в (5.24).

   Если $f_0(A) = 1$, то $g = \mathcal{L}(\varphi_{\pi_g}) \in \mathcal{L}(\Sigma_A)$. Если же $f_0(A) = 0$, то из (5.25) следует, что $f'_0(\bar{A}) = 1$, поэтому $g = \mathcal{L}(\varphi'_{\pi_g}) \in \mathcal{L}(\Sigma_{\bar{A}})$. ■

Таким образом, исходная задача доказательства (или опровержения) утверждения (5.21) сводится к проверке равенства (5.23). Проверка данного равенства м.б. выполнена на основе излагаемого в следующем пункте понятия автомата Бюхи.

5.4 Автоматы Бюхи

5.4.1 Понятие автомата Бюхи

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

$$\mathcal{B} = (\mathcal{A}, S, R, S^0, \mathcal{F}), \tag{5.26}$$

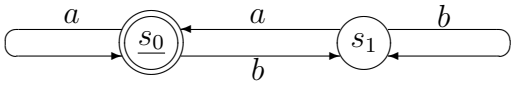
компоненты которой имеют следующий смысл:

- $\mathcal{A}$ – множество, называемое **алфавитом** автомата $\mathcal{B}$,
- S – множество, элементы которого называются **состояниями**,
- $R \subseteq S \times \mathcal{A} \times S$ – **отношение перехода**,
- $S^0 \subseteq S$ – множество **начальных состояний**,
- $\mathcal{F} = \{F_i \subseteq S \mid i = 1, \ldots, n\}$ – совокупность **условий fairness**.

Элементы R называются **переходами**. Каждый переход (s, a, s') из R может обозначаться записью $s \xrightarrow{a} s'$.

Автомат (5.26) можно представить в виде графа (обозначаемого тем же символом $\mathcal{B}$), вершинами которого являются состояния этого автомата. $\forall\, (s \xrightarrow{a} s') \in R$ граф $\mathcal{B}$ содержит ребро с меткой a из s в s'.

Пример автомата:



(5.27)

Данный автомат имеет следующие компоненты:

- $\mathcal{A} = \{a, b\}$, $S = \{s_0, s_1\}$, $S^0 = \{s_0\}$ (начальные состояния обозначаются двойными кружочками),
- $\mathcal{F} = \{\{s_0\}\}$ (состояния из условий fairness выделяются подчёркиванием).

5.4.2 Язык автомата

Пусть задан автомат $\mathcal{B} = (\mathcal{A}, S, R, S^0, \mathcal{F})$.

Будем использовать следующие понятия и обозначения: если $\pi = (s_0 \stackrel{a_1}{\to} s_1 \stackrel{a_2}{\to} \ldots)$ – путь в $\mathcal{B}$, то

- первое состояние в π (т.е. s_0) называется **началом** пути π,
- запись $\inf(\pi)$ обозначает множество всех состояний, которые входят в π бесконечное число раз,
- π называется **fair путём**, если $\forall\, F \in \mathcal{F} \quad \inf(\pi) \cap F \neq \emptyset$,
- запись $\mathcal{L}(\pi)$ обозначает последовательность меток рёбер, из которых состоит π, т.е. $\mathcal{L}(\pi) = (a_1, a_2, \ldots)$.

Для каждого состояния s автомата $\mathcal{B}$ запись Π_s обозначает множество всех fair путей, началом которых является s.

Язык автомата $\mathcal{B}$ – это множество

$$\mathcal{L}(\mathcal{B}) = \{\mathcal{L}(\pi) \mid \exists\, s \in S^0, \exists\, \pi \in \Pi_s\}.$$

Например, язык автомата (5.27) состоит из всех бесконечных конкатенаций $u_1 u_2 \ldots$, где $\forall\, i \geq 1$ – цепочка вида $b \ldots ba$, состоящей из ≥ 0 символов b, после которых следует символ a. Данный язык обозначается записью $(b^* a)^\infty$.

Теорема 11.
Пусть задан автомат $\mathcal{B}$ вида (5.26), причем $\mathcal{F} = \{F\}$.
Следующие условия эквивалентны:

- $\mathcal{L}(\mathcal{B}) \neq \emptyset$,
- существует конечный путь из некоторого состояния $s_0 \in S^0$ в некоторую SCC $\mathbf{C}$ графа $\mathcal{B}$, такую, что $\mathbf{C} \cap F \neq \emptyset$,
- существует конечный путь из некоторого состояния $s_0 \in S^0$ в некоторое состояние $s \in F$, через которое проходит цикл графа $\mathcal{B}$. ■

5.4.3 Эквивалентность автоматов

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

Теорема 12.
Для каждого автомата (5.26) существует эквивалентный ему автомат

$$\mathcal{B}_1 = (\mathcal{A}, S_1, R_1, S_1^0, \mathcal{F}_1)$$

такой, что $\mathcal{F}_1$ состоит только из одного множества.

Доказательство.
Компоненты автомата $\mathcal{B}_1$ можно определить, например, так:

- $S_1 = S \times \{0, 1, \ldots, n\}$, $S_1^0 = S^0 \times \{0\}$, $\mathcal{F}_1 = \{S \times \{n\}\}$,
- R_1 состоит из переходов $(s, j) \xrightarrow{a} (s', j')$, таких, что $s \xrightarrow{a} s'$ и

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

Ниже для каждого автомата, у которого вид совокупности $\mathcal{F}$ его условий fairness не описан явно, будем предполагать, что

- совокупность $\mathcal{F}$ состоит из одного множества, и
- эта совокупность будет обозначаться той же записью, что и то единственное множество, из которого состоит $\mathcal{F}$.

5.4.4 Пересечение автоматов

Теорема 13.
Для каждой пары автоматов $\mathcal{B}_1, \mathcal{B}_2$ с общим алфавитом, где

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

существует автомат $\mathcal{B}_1 \cap \mathcal{B}_2 = (\mathcal{A}, S, R, S^0, F)$, называемый **пересечением** $\mathcal{B}_1$ и B_2, и обладающий следующим свойством:

$$\mathcal{L}(\mathcal{B}_1 \cap \mathcal{B}_2) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2). \tag{5.28}$$

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

- $S = S_1 \times S_2 \times \{0,1,2\}$, $S^0 = S_1^0 \times S_2^0 \times \{0\}$, $F = S_1 \times S_2 \times \{2\}$,
- R состоит из переходов $(s_1, s_2, j) \xrightarrow{a} (s_1', s_2', j')$, таких, что

$$s_i \xrightarrow{a} s_i' \quad (i = 1,2), \quad j' = \begin{cases} 1 & \text{если } j = 0 \text{ и } s_1' \in F_1, \\ 2 & \text{если } j = 1 \text{ и } s_2' \in F_2, \\ 0 & \text{если } j = 2, \\ j & \text{в остальных случаях.} \end{cases}$$

Если $F_1 = S_1$, то компоненты автомата $\mathcal{B}_1 \cap \mathcal{B}_2$ можно определить проще:

- $S = S_1 \times S_2, \quad S^0 = S_1^0 \times S_2^0, \quad F = S_1 \times F_2,$
- $R \stackrel{\text{def}}{=} \{(s_1, s_2) \xrightarrow{a} (s_1', s_2') \mid s_i \xrightarrow{a} s_i' \quad (i = 1,2)\}$. ■

5.4.5 Использование автоматов Бюхи для MC-LTL

Лемма 6.

Пусть заданы СП Σ и LTL–формула A.

Имеет место равенство $\mathcal{L}(\Sigma) = \mathcal{L}(\mathcal{B}_\Sigma)$, где $\mathcal{B}_\Sigma = (\mathcal{A}, S, R, S^0, \mathcal{F})$ – автомат Бюхи, компоненты которого определяются следующим образом:

$$\mathcal{A} = [A], \;\; S = S_\Sigma \sqcup \{init\}, \;\; S^0 = \{init\}, \;\; \mathcal{F} = \mathcal{F}_\Sigma,$$
$$R = \{s \xrightarrow{g_{s'}} s' \mid (s \to s') \in R_\Sigma\} \cup \{init \xrightarrow{g_s} s \mid s \in S_\Sigma^0\}$$

где $\forall\, s \in S_\Sigma, \forall\, p \in \mathcal{P}_A \;\; g_s(p) = s(p)$. ■

Основанный на данной лемме метод проверки соотношения (5.23) заключается в построении автоматов $\mathcal{B}_\Sigma$, $\mathcal{B}_{\Sigma_{\bar{A}}}$, и проверке пустоты языка автомата $\mathcal{B}_\Sigma \cap \mathcal{B}_{\Sigma_{\bar{A}}}$. Согласно теореме 11, язык этого автомата непуст тогда и только тогда, когда существует путь из его начального состояния в какое-либо fair состояние, через которое проходит цикл.

Автомат $\mathcal{B}_\Sigma \cap \mathcal{B}_{\Sigma_{\bar{A}}}$ можно строить "на лету" ("on-the-fly"): сначала строится $\mathcal{B}_{\Sigma_{\bar{A}}}$, который используется в процессе построения $\mathcal{B}_\Sigma$. Пусть состояниями $\mathcal{B}_\Sigma \cap \mathcal{B}_{\Sigma_{\bar{A}}}$ являются пары $(s, s_{\bar{A}})$, где s – состояние $\mathcal{B}_\Sigma$, и $s_{\bar{A}}$ – состояние $\mathcal{B}_{\Sigma_{\bar{A}}}$. Если уже построено некоторое состояние s автомата $\mathcal{B}_\Sigma$, то к тому фрагменту автомата $\mathcal{B}_\Sigma$, который уже построен, добавляются только такие состояния s', что

$$\exists\, a \in \mathcal{A}, \exists\, s_{\bar{A}}' : \quad s \xrightarrow{a} s', s_{\bar{A}} \xrightarrow{a} s_{\bar{A}}'.$$

Если построена часть $\mathcal{B}_\Sigma \cap \mathcal{B}_{\Sigma_{\bar{A}}}$, содержащая путь, упомянутый в теореме 11, то в дальнейшем построении $\mathcal{B}_\Sigma$ уже нет необходимости.

5.4.6 Оптимизация построения автомата $\mathcal{B}_{\bar{A}}$

Автомат $\mathcal{B}_{\bar{A}}$ можно строить не по $\Sigma_{\bar{A}}$, а по более компактной СП Σ, такой, что $\mathcal{L}(\Sigma) = \mathcal{L}(\Sigma_{\bar{A}})$. Ниже излагается алгоритм построения такой СП.

Будем использовать соглашение: $\forall\, B, C \in Fm_{LTL}$ запись $\mathbf{R}(B, C)$ рассматривается как LTL–формула, которая по определению равна формуле $\overline{\mathbf{U}(\overline{B}, \overline{C})}$. Из этого соглашения следует, что $\forall\, B, C \in Fm_{LTL}$

$$\begin{array}{l} \bar{*} = \mathbf{R}(\bar{B}, \bar{C}), \\ \mathbf{R}(B, C) = C \wedge \Big(B \vee \mathbf{X}\mathbf{R}(B, C)\Big). \end{array} \tag{5.29}$$

Преобразуем $\bar{A}$ в эквивалентную ей формулу $\tilde{A}$, в которой отрицания располагаются только над атомарными утверждениями, используя законы де Моргана и равенства $\overline{\mathbf{X}}\overline{B} = \mathbf{X}\bar{B}$ и $\bar{*} = \mathbf{R}(\bar{B}, \bar{C})$.

Далее строим СП, каждое состояние которой является подмножеством множества $\langle\tilde{A}\rangle$. На каждом шаге построения данная СП является аппроксимацией искомой СП Σ. Мы будем обозначать одним и тем же символом Σ данную СП на каждом шаге ее построения. В конце построения данная СП будет представлять собой искомую СП. На каждом шаге построения СП Σ должны быть выполнены следующие условия:

- $\forall\, s \in S_\Sigma$ указано разбиение $s = \mathrm{New}(s) \sqcup \mathrm{Old}(s)$,
- $\forall\, s \in S_\Sigma,\ \forall\, p \in \mathcal{P}_A \quad L_\Sigma(s, p)$ равно 1, если $p \in s$, и 0 – если $\bar{p} \in s$,
- $\mathcal{F}_\Sigma = \{F_* \mid * \in \langle\tilde{A}\rangle\}$, где $\forall * \in \langle\tilde{A}\rangle \quad F_* \stackrel{\text{def}}{=} \{s \in S \mid * \in s \ \Rightarrow\ C \in s\}$,
- $\forall\, s \in S_\Sigma,\ \forall\, B \in s,\ \forall\, \pi \in \Pi_s \quad \pi(B) = 1$.

Сначала $S_\Sigma \stackrel{\text{def}}{=} \{\tilde{A}\} = \mathrm{New}(s_0)$.

На очередном шаге построения выбираем произвольное $s \in S_\Sigma$, такое, что $\mathrm{New}(s) \neq \emptyset$ (если таких s нет, то построение закончено), и выполняем одно из двух следующих действий:

1. если $\exists\, D \in \mathrm{New}(s)$, не начинающаяся с $\mathbf{X}$, то переносим D из $\mathrm{New}(s)$ в $\mathrm{Old}(s)$, после чего
   (a) если $D = B \wedge C$, то добавляем B и C к $\mathrm{New}(s)$,
   (b) если $D = B \vee C$, то
   - добавляем дубликат s' состояния s (с теми же New и Old), и для каждого ребра, ведущего в s, добавляем новое ребро с тем же началом, но с концом в s',

- добавляем B к $\mathrm{New}(s)$, C к $\mathrm{New}(s')$,

(c) если $D = *$, то обрабатываем её как $C \vee \Big(B \wedge \mathbf{X} * \Big)$, т.е.

- добавляем дубликат s' состояния s и рёбра в s',
- добавляем C к $\mathrm{New}(s)$, B и $\mathbf{X}*$ к $\mathrm{New}(s')$,

(d) если $D = \mathbf{R}(B, C)$, то обрабатываем её как $C \wedge \Big(B \vee \mathbf{X}D\Big)$, т.е.

- добавляем дубликат s' состояния s и рёбра в s',
- добавляем C и B к $\mathrm{New}(s)$, C и $\mathbf{X}D$ к $\mathrm{New}(s')$,

после чего

- если s содержит $\bot$, или пару формул вида $p, \bar{p}$, то удаляем s и все ведущие в него рёбра,
- если в s входит $\top$, то удаляем $\top$ из s,
- если в s входит пара одинаковых формул, то удаляем ту из них, которая входит в $\mathrm{New}(s)$,
- в случаях (b–d) выполняем те же действия для s',

2. если все формулы в $\mathrm{New}(s)$ начинаются с $\mathbf{X}$, то

(a) если $\exists\, s' \neq s$: $\mathrm{New}(s) \subseteq \mathrm{New}(s')$ и $\mathrm{Old}(s) = \mathrm{Old}(s')$, то удаляем s, и перенаправляем в s' все рёбра, ведущие в s,

(b) иначе добавляем $s' = \{B \mid \mathbf{X}B \in \mathrm{New}(s)\} = \mathrm{New}(s')$ и ребро из s в s', и удаляем все формулы из $\mathrm{New}(s)$.

После завершения построения S_Σ полагаем $S^0_\Sigma \stackrel{\mathrm{def}}{=} \{s \in S_\Sigma \mid \tilde{A} \in s\}$.

Глава 6

Вероятностный model checking

6.1 Введение

Наряду с изложенными выше моделями дискретных динамических систем используются и другие модели, среди которых наибольшую популярность получили **вероятностные системы переходов (ВСП)**.

ВСП являются эффективным средством моделирования таких программных систем, в которых присутствуют ненадежные компоненты, выходящие из строя с некоторой вероятностью, или возникают различные случайные события (например потеря сообщений в каналах связи), или в алгоритмах явно присутствует рандомизация.

В этой главе мы будем изучать логический язык описания свойств ВСП и методы их анализа, называемые **вероятностным model checking** (**probabilistic model checking, PMC**). В настоящее время PMC является одним из наиболее широко используемых методов моделирования и верификации вычислительных систем.

Понятие ВСП является обобщением понятия цепи Маркова [11], которое имеет широкие применения в естественных и гуманитарных науках. Понятие ВСП можно рассматривать также как частный случай понятия вероятностного автомата [12].

Одной из главных причин актуальности ВСП в задачах верификации программных систем является существенно меньшая (по сравнению с моделями в виде СП) сложность вероятностных моделей анализируемых систем. Однако построение и анализ моделей анализируемых систем в виде ВСП является нетривиальной задачей по следующим причинам:

- для получения численных значений вероятностей переходов в модели анализируемой системы в виде ВСП необходимо проведение достаточно трудоемких экспериментов с исходной системой,

- свойства модели в виде ВСП могут отличаться от свойств исходной системы, это приводит к проблеме оценки меры соответствия свойств исходной системы и свойств ее модели в виде ВСП.

Первые алгоритмы PMC были предложены в 1980-е годы в работах [13], [14], [15]. Данные алгоритмы были предназначены для верификации качественных вероятностных свойств (то есть таких, которые выполняются с вероятностью 1 или 0). Затем эти алгоритмы были обобщены на случай верификации количественных вероятностных свойств. Эти алгоритмы были изложены в работах [16], [17], [18]. Программые реализации этих алгоритмов были представлены в работах [19], [20]. Обзор первого этапа развития PMC можно найти в [21].

Первые промышленные системы PMC были разработаны в 2000-х годах [22], [23]. Эти системы успешно применяются во многих областях, в том числе: анализ распределенных алгоритмов, телекоммуникационные протоколы, компьютерная безопасность, криптографические протоколы, моделирование биологических процессов. С использованием этих систем PMC были обнаружены уязвимости и аномальные поведения анализируемых систем, некоторые подробности см. в [24] и [26]. При помощи систем PMC могут быть вычислены такие характеристики программных систем как например вероятность вторжения злоумышленника в компьютерную сеть, мат. ожидание времени отклика веб-сервиса, и другие количественные и качественные характеристики.

Наиболее популярной практической системой PMC является система PRISM [25], [26], разработанная на факультете компьютерных наук Оксфордского Университета (Великобритания) в группе Quantitative Analysis and Verification под руководством Марты Квятковской [10].

6.2 Вероятностные системы переходов

6.2.1 Понятие вероятностной системы переходов

Вероятностная система переходов (ВСП) (называемая также **Discrete Time Markov Chain**) – это четверка Σ вида

$$\Sigma = (S, P, L, s^0), \tag{6.1}$$

компоненты которой имеют следующий смысл:

- S – множество **состояний** ВСП Σ,

- P – функция вида $P : S \times S \to [0, 1]$, называемая **функцией переходов** ВСП Σ, и удовлетворяющая условию:

$$\forall\, s \in S \quad \sum_{s' \in S} P(s, s') = 1,$$

- L – функция вида $L : S \times \mathcal{P} \to \{0, 1\}$, называемая **оценкой**,
- $s^0 \in S$ – **начальное состояние** ВСП Σ.

$\forall\, (s_1, s_2) \in S \times S$ число $P(s_1, s_2)$ понимается как вероятность того, что если в текущий момент времени Σ находится в состоянии s_1, то через один такт времени Σ будет находиться в состоянии s_2.

Оценка L имеет следующий смысл: $\forall\, s \in S$, $\forall\, p \in \mathcal{P}$ утверждение p считается **истинным** в s, если $L(s, p) = 1$, и **ложным**, иначе.

ВСП $\Sigma = (S, P, L, s^0)$ удобно рассматривать как граф (обозначаемый тем же символом Σ), в котором

- вершинами являются состояния из S, и
- для каждой пары $(s_1, s_2) \in S \times S$ такой, что $P(s_1, s_2) > 0$, имеется ребро из s_1 в s_2 с меткой $P(s_1, s_2)$.

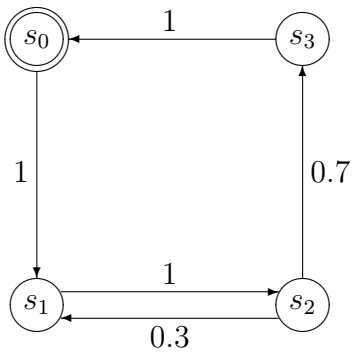
Рёбра данного графа будем называть **переходами** ВСП Σ.

6.2.2 Примеры вероятностных систем переходов

1. Упрощённая модель протокола передачи сообщений через ненадёжный канал, в котором сообщения могут пропадать. Протокол представляет собой систему, состоящую из

   - двух агентов - отправителя и получателя, а также
   - канала, в который помещаются сообщения, пересылаемые от одного агента другому.

   Граф, представляющий эту ВСП, имеет следующий вид:



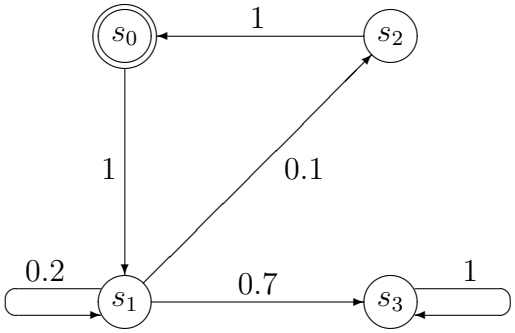
(6.2)

Переходы этой ВСП имеют следующий смысл.

- Переход $s_0 \xrightarrow{1} s_1$ заключается в получении отправителем от внешнего источника сообщения, которое должно быть передано через канал получателю.
- Переход $s_1 \xrightarrow{1} s_2$ заключается в помещении сообщения в канал отправителем.
- Переход $s_2 \xrightarrow{0.3} s_1$ заключается в потере сообщения в канале.
- Переход $s_2 \xrightarrow{0.7} s_3$ заключается в передаче сообщения из канала получателю.
- Переход $s_3 \xrightarrow{1} s_0$ заключается в получении сообщения получателем и посылка им подтверждения отправителю.

2. Упрощённая модель агента, работа которого представляет собой последовательность сеансов. В каждом из этих сеансов агент пытается выполнить некоторое действие. Если действие было выполнено, то он переходит к следующему сеансу, а если не было выполнено, то он завершает свою работу.

   Граф, представляющий эту ВСП, имеет следующий вид:



   Состояния этой ВСП имеют следующий смысл:

   - В состоянии s_0 агент начинает очередной сеанс.
   - В состоянии s_1 агент предпринимает попытку выполнения действия (переход $s_1 \xrightarrow{0.2} s_1$ означает, что попытка выполнения действия пока не осуществима).

- В состояние s_2 агент попадает тогда, когда действие было выполнено успешно.
- В состоянии s_3 агент оказывается тогда, когда его попытка выполнить действие закончилась неудачей.

6.3 Темпоральная логика PCTL

6.3.1 Свойства вероятностных систем переходов

Одним из логических языков, предназначенных для формального описания свойств поведения ВСП, является темпоральная логика **PCTL** (**Probabilistic Computation Tree Logic**). Логика PCTL была введена Х. Ханссоном (H. Hansson) и Б. Джонссоном (B. Jonsson) в работе [16].

Формулы логики PCTL могут отражать различные вероятностные аспекты поведения систем, к числу которых относятся, например

- частота выполнения тех или иных действий или переходов,
- вероятность отказа компонентов систем,
- вероятностный характер взаимодействия системы с её окружением, например: частота поступления входных запросов или сообщений, частота получения искажённых сообщений (для протоколов передачи сообщений в компьютерных сетях), и т.п.

Примеры свойств, которые можно описать формулами PCTL:

- вероятность доставки сообщения в заданном временном интервале $[t_1, t_2]$ не меньше 0.999,
- в результате работы алгоритма избрания процесса-лидера избрание такого процесса завершится с вероятностью 1,
- вероятность успешной атаки на криптографический протокол не превышает 0.0001,
- вероятность отклика веб-сервиса в течение 5ms – не менее 0.8,
- ожидаемое время в худшем случае для доставки пакета данных при помощи протокола беспроводной связи – 1c,
- максимальное ожидаемое энергопотребление за 24 часа для устройства с электропитанием от батареи - не более 190J,

- вероятность того, что система после любого отказа восстановится за $\leq n$ шагов, не менее $1 - \varepsilon$,
- вероятность того что сигнал ready будет получен в течение следующих n единиц времени, больше половины.

6.3.2 Формулы логики PCTL

Совокупность **формул** логики PCTL, называемых также **PCTL-формулами**, обозначается записью Fm_{PCTL}. Формулы логики PCTL делятся на два класса: Fm^s_{PCTL} (**state-формулы**) и Fm^p_{PCTL} (**path-формулы**), определяемые следующим образом:

- $\mathcal{P} \subseteq Fm^s_{PCTL}$,
- Fm^s_{PCTL} замкнут относительно булевых комбинаций,
- $\forall\, A \in Fm_{PCTL}$ запись $\mathbf{X}A$ является path-формулой,
- $\forall\, A, B \in Fm^s_{PCTL},\ \forall\, n \geq 0$ записи $\mathbf{U}^{\leq n}(A, B)$ и $\mathbf{U}(A, B)$ являются path-формулами,
- $\forall\, A \in Fm^p_{PCTL},\ \forall\, a \in [0, 1]$ записи $[\![A]\!]_{\geq a}$, $[\![A]\!]_{>a}$, $[\![A]\!]_{\leq a}$ $[\![A]\!]_{<a}$, являются state-формулами.

$\forall\, A \in Fm^s_{PCTL},\ \forall\, n \geq 0$ записи $\mathbf{F}^{\leq n}A$ и $\mathbf{F}A$ обозначают формулы $\mathbf{U}^{\leq n}(\top, A)$ и $\mathbf{U}(\top, A)$ соответственно, и записи $\mathbf{G}^{\leq n}A$ и $\mathbf{G}A$ обозначают формулы $\overline{\mathbf{F}^{\leq n}\overline{A}}$ и $\overline{\mathbf{F}\ \overline{A}}$ соответственно.

6.3.3 Значения формул логики PCTL в состояниях вероятностных систем переходов

Пусть задана ВСП $\Sigma = (S, P, L, s^0)$.

$\forall\, s \in S$, $\forall\, A \in Fm_{PCTL}$ ниже определяется **значение** формулы A в состоянии s, которое обозначается записью $s(A)$, и

- если $A \in Fm^s_{PCTL}$, то $s(A) \in \{0, 1\}$,
- если $A \in Fm^p_{PCTL}$, то $s(A) \in [0, 1]$, $s(A)$ интерпретируется как вероятность того, что формула A истинна в состоянии s.

В излагаемом ниже определении значений вида $s(A)$ будем использовать следующие обозначения. Пусть S имеет вид $\{s_1, \ldots, s_n\}$.

- $\forall\, A \in Fm_{PCTL}$ будем обозначать записью $S(A)$ столбец $\begin{pmatrix} s_1(A) \\ \dots \\ s_n(A) \end{pmatrix}$.

- Для любых столбцов $U = \begin{pmatrix} u_1 \\ \dots \\ u_n \end{pmatrix}$, $V = \begin{pmatrix} v_1 \\ \dots \\ v_n \end{pmatrix}$ из $[0,1]^n$ записи $\max(U,V)$ и UV обозначают столбцы $\begin{pmatrix} \max(u_1,v_1) \\ \dots \\ \max(u_n,v_n) \end{pmatrix}$ и $\begin{pmatrix} u_1v_1 \\ \dots \\ u_nv_n \end{pmatrix}$ соответственно.

- Символ P обозначает матрицу порядка $n \times n$, в которой элемент в i-й строке и j-м столбце равен $P(s_i, s_j)$.

- Для любого столбца $U \in [0,1]^n$ запись $[P^*U]$ обозначает столбец из $\{0,1\}^n$, получаемый заменой всех ненулевых компонентов матрицы P и столбца U на 1, и вычислением $(\sum\limits_{i\geq 0} P^i)U$, где сложение понимается как дизъюнкция (т.е. $1+1=1$, и сумма $\sum\limits_{i\geq 0} P^i$ конечна).

Значения формул логики PCTL в состояниях ВСП определяются индукцией по структуре формул в соответствии с излагаемыми ниже правилами. В одних из этих правил мы определяем значение $s(A)$, в других определяем столбец $S(A)$ целиком.

- $\forall\, s \in S,\ \forall\, p \in \mathcal{P} \quad s(p) = L(s,p)$.

- $\forall\, s \in S \ \ s(\top) = 1,\ s(\bot) = 0$.

- Значения булевых комбинаций определяются стандартным образом: $\forall\, s \in S \ \ s(\bar{A}) = \overline{s(A)}, s(A \wedge B) = s(A) \wedge s(B)$, и т.д.

- $S(\mathbf{X}A) = PS(A)$.

- $S(\mathbf{U}^{\leq 0}(A,B)) = S(B)$,
  $\forall\, n > 0 \quad S(\mathbf{U}^{\leq n}(A,B)) = \max\Big(S(B),\ S(A)S(\mathbf{X}\mathbf{U}^{\leq n-1}(A,B))\Big)$.

- $S(\mathbf{U}(A,B))$ определяется системой линейных уравнений

$$S(\mathbf{U}(A,B)) = \max\Big(S(B),\ [P^*S(B)]S(A)(PS(\mathbf{U}(A,B)))\Big).$$

- $\forall\, s \in S \;\; s([\![A]\!]_{\geq a}) = [\![s(A) \geq a]\!]$, т.е. $s([\![A]\!]_{\geq a}) = \begin{cases} 1, & \text{если } s(A) \geq a, \\ 0, & \text{иначе,} \end{cases}$
- аналогичное определение для формул вида $[\![A]\!]_{>a}$, $[\![A]\!]_{\leq a}$, $[\![A]\!]_{<a}$.

6.3.4 Интерпретация значений формул логики PCTL

Пусть задана ВСП Σ. $\forall\, s \in S_\Sigma$, $\forall\, A \in Fm_{PCTL}$ значение $s(A)$ интерпретируется следующим образом:

- если $A \in Fm^s_{PCTL}$, то формула A истинна в s при $s(A) = 1$, и ложна в s при $s(A) = 0$,
- если $A = \mathbf{X}B$, то $s(A)$ интерпретируется как математическое ожидание значения формулы B в том состоянии, в которое будет совершён переход из состояния s за один такт времени,
- если $A = \mathbf{U}^{\leq n}(B, C)$, то $s(A)$ интерпретируется как вероятность того, что для произвольного пути π в графе Σ, выходящего из s, существует состояние s' на этом пути, такое, что длина отрезка от s до s' пути π не превосходит n, и
  - в каждом состоянии этого отрезка, кроме м.б. s', истинна B,
  - в состоянии s' истинна C,
- если $A = \mathbf{U}(B, C)$, то $s(A)$ интерпретируется так же, как значение предыдущей формулы, без упоминания того, что длина пути из s в s' не превосходит n.

На основе этой интерпретации можно описывать свойства ВСП формулами логики PCTL. Эти свойства могут выражать динамические аспекты поведения ВСП, т.е. описывать зависимость значения какого-либо утверждения в некотором состоянии рассматриваемой ВСП от значений других утверждений в других состояниях этой ВСП. Например, свойство протокола из пункта 6.2.2, представленного ВСП (6.2)

> каждое сообщение, полученное отправителем от
> внешнего источника, будет доставлено получателю
> за не более чем 5 шагов с вероятностью ≥ 0.9

выражается формулой $\mathbf{G}\Big(p_0 \to [\![\mathbf{F}^{\leq 5} p_3]\!]_{\geq 0.9}\Big)$, где $s_i(p_j) = [\![i = j]\!]$.

Литература

[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] **Лекции лауреатов премии Тьюринга.** *Москва, Мир, 1993.*

[3] **Карпов Ю.Г.**: Model Checking. Верификация параллельных и распределенных программных систем *СПб.: БХВ-Петербург, 2010.*

[4] **E. Clarke, O. Grumberg, D. Peled:** Model checking. *The MIT Press, 2001.*

[5] **C. Baier and J.-P. Katoen**: Principles of Model Checking *The MIT Press, 2008.*

[6] `http://spinroot.com/spin/whatispin.html`

[7] `http://coq.inria.fr/`

[8] `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/`

[9] `http://pvs.csl.sri.com/`

[10] `http://qav.comlab.ox.ac.uk/`

[11] **Кемени Дж., Снелл Дж.** Конечные цепи Маркова. *Москва, Наука, 1970.*

[12] **Бухараев Р.Г.** Основы теории вероятностных автоматов. *Москва, Наука, 1985.*

[13] **S. Hart, M. Sharir, A. Pnueli.** Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems, 5(3):356-380, 1983.*

[14] **M. Vardi.** Automatic verification of probabilistic concurrent finite state programs. *In Proc. 26th Annual Symposium on Foundations of Computer Science (FOCS'85), pages 327-338. IEEE Computer Society Press, 1985.*

[15] **C. Courcoubetis and M. Yannakakis.** Verifying temporal properties of finite state probabilistic programs. *In Proc. 29th Annual Symposium on Foundations of Computer Science (FOCS'88), pages 338-345. IEEE Computer Society Press, 1988.*

[16] **H. Hansson and B. Jonsson.** A logic for reasoning about time and reliability. *Formal Aspects of Computing, 6(5):512-535, 1994.*

[17] **A. Bianco and L. de Alfaro.** Model checking of probabilistic and nondeterministic systems. *In P. Thiagarajan, editor, Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95), volume 1026 of LNCS, pages 499-513. Springer, 1995.*

[18] **C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen.** Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering, 29(6):524-541, 2003.*

[19] **H. Hansson.** Time and Probability in Formal Design of Distributed Systems. *Elsevier, 1994.*

[20] **C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan.** Symbolic model checking for probabilistic processes. *In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, Proc. 24th International Colloquium on Automata, Languages and Programming (ICALP'97), volume 1256 of LNCS, pages 430-440. Springer, 1997.*

[21] **Marta Kwiatkowska, David Parker.** Advances in Probabilistic Model Checking.
`http://qav.comlab.ox.ac.uk/papers/marktoberdorf11.pdf`

[22] **L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, and R. Segala.** Symbolic model checking of probabilistic processes using MTBDDs and the Kronecker representation. *In S. Graf and M. Schwartzbach, editors, Proc. 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00), volume 1785 of LNCS, pages 395-410. Springer, 2000.*

[23] **H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle.** A Markov chain model checker. *In S. Graf and M. Schwartzbach, editors, Proc. 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00), volume 1785 of LNCS, pages 347-362. Springer, 2000.*

[24] **M. Kwiatkowska, G. Norman, and D. Parker.** Probabilistic model checking in practice: Case studies with PRISM. *ACM SIGMETRICS Performance Evaluation Review, 32(4):16-21, 2005.*

[25] **M. Kwiatkowska, G. Norman, and D. Parker.** PRISM 4.0: Verification of probabilistic real-time systems. *In G. Gopalakrishnan and S. Qadeer, editors, Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585-591. Springer, 2011.*

[26] `http://www.prismmodelchecker.org/`