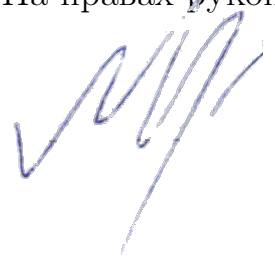
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Санкт-Петербургский государственный университет»

На правах рукописи



Мордвинов Дмитрий Александрович

АВТОМАТИЧЕСКИЙ ВЫВОД РЕЛЯЦИОННЫХ ИНВАРИАНТОВ ДЛЯ НЕЛИНЕЙНЫХ СИСТЕМ ДИЗЪЮНКТОВ ХОРНА С ОГРАНИЧЕНИЯМИ

Специальность 05.13.11 —
«Математическое и программное обеспечение вычислительных машин,
комплексов и компьютерных сетей»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор технических наук, доцент, Санкт-Петербургский государственный
университет, профессор кафедры системного программирования
КОЗНОВ Дмитрий Владимирович

Санкт-Петербург — 2020

Содержание

Стр.

Стр.

Стр.

Введение

Актуальность темы. В условиях современного мира, когда компьютеры управляют многочисленными критически важными аспектами жизни человека, доказательство корректности программного кода все больше превращается в необходимость. Объём кода в ядрах операционных систем, драйверах различных устройств, инструментах и библиотеках растет, вопросы его качества становятся всё более насущными. При этом доказательство корректности кода вручную не представляется реалистичным выходом из ситуации, так как требует слишком больших трудозатрат.

В последние 15 лет активное развитие получили подходы, которые привели к появлению SMT-решателей — инструментов эффективного поиска моделей для формул теорий логики первого порядка, используемых для автоматического доказательства теорем [1]. SMT-решатели оказались эффективными для статического анализа кода и автоматического поиска ошибок и используются в таких подходах, как символьное исполнение (Symbolic Execution) [2] и ограниченная проверка моделей (Bounded Model Checking) [3]. Они также могут быть использованы для автоматического поиска ошибок и уязвимостей в коде, для генерации тестовых данных для обеспечения высокого уровня покрытия [4]. Однако применение SMT-решателей для формальной верификации программ сталкивается со значительными трудностями.

Большинство современных подходов к верификации, включая вывод индуктивных инвариантов программ, может быть сведено к поиску символьных моделей для систем дизъюнктов Хорна с ограничениями, сформулированными в теориях первого порядка. Это позволяет использовать решатель дизъюнктов Хорна с ограничениями в качестве переиспользуемого ядра верификатора.

Дизъюнкты Хорна с ограничениями [5] позволяют моделировать программы в виде набора логических импликаций. При этом проверка корректности программы сводится к проверке выполнимости дизъюнктов в некоторой теории. Примечательно, что интерпретации, в которых выполняется система дизъюнктов, соответствуют инвариантам моделируемой программы.

В настоящее время существуют высокопроизводительные решатели систем дизъюнктов Хорна (далее — Хорн-решатели), такие как Spacer [6], Eldarica [7], Qarmc [8], HoIce [9] и FreqHorn [10], использующие SMT-решатели для проверки выполнимости ограничений (в частности, Z3 [11]).

Были также разработаны эффективные верификаторы, использующие решатели дизъюнктов Хорна для автоматического доказательства утверждений корректности программ на различных языках программирования (SeaHorn [12], JayHorn [13], AdaHorn и нек. др.). Эти верификаторы показывают хорошие результаты на различных соревнованиях (например, SV-COMP [14]), а также на практике.

Система дизъюнктов Хорна с ограничениями называется линейной, если в посылке каждого правила находится не более одного вхождения неинтерпретированного символа; в противном случае система называется нелинейной. Как показывает практика, большинство существующих подходов к решению систем дизъюнктов Хорна с ограничениями эффективны для линейных систем, но плохо справляются с нелинейными системами. В то же время нелинейные системы естественным образом возникают на практике как условия верификации свойств безопасности (т.е. свойств, опровергаемых конечными исполнениями программы), нетривиальных программ с множественными вызовами рекурсивных функций и функций с циклами, а также хорошо подходят для спецификации свойств гипербезопасности программ.

Степень разработанности темы. Исследования по построению эффективных подходов к выводу символьных представлений моделей систем дизъюнктов Хорна активно проводились, начиная с 2000-х годов. Дизъюнкты Хорна с ограничениями стали активно изучаться с появлением логического программирования в ограничениях (Constraint Logic Programming) [5]; пионером в этой области является J. Jaffar. Наиболее известными Хорн-решателями являются Spacer (N. Bjorner, A. Gurfinkel, Microsoft Research, США), Eldarica (P. Ruemmer, Швеция), Qarmc (А. Рыбальченко и K. McMillan, США), FreqHorn (Г. Федюкович, США), HoIce (A. Champion, N. Kobayashi, Франция и Япония). Подход PDR к решению систем дизъюнктов Хорна, предложенный A. Bradley, является на сегодняшний день одним из самых эффективных. Он основан на идее *достижимости, направляемой свойством* (Property-Directed Reachability) [6,15,16] и реализован в известном Хорн-решателе Spacer. В рамках данного подхода итеративно строится серия *индуктивных усилений* свойства безопасности, что обеспечивает композициональный вывод символьных моделей без раскрутки отношения перехода системы. Как показали результаты соревнований CHC-Comp 2018, этот подход хорошо справляется с построением решений линейных систем, однако на нелинейных системах он работает хуже.

Одна из сложностей решения нелинейных систем дизъюнктов Хорна заключается состоит в том, что часто их символьные модели оказываются непредставимы в теориях первого порядка, используемых Хорн-решателями. При этом у некоторых систем существует теоретико-множественная модель, но не существует символьной модели, представимой в языке ограничений. Эта проблема фундаментальна и связана с компромиссом между разрешимостью и выразительностью теории, используемой Хорн-решателями. Например, арифметика Пресбургера разрешима, но в ней представимы лишь полулинейные отношения [17], в то время как арифметика Пеано полна для представления моделей систем дизъюнктов Хорна, но неразрешима [18, 19]. Поиск эффективного подхода, который мог бы справиться с этими трудностями, при этом наследуя эффективность подхода PDR, остается открытой проблемой.

С 2015 года было сделано несколько попыток построить эффективный поход к решению нелинейных систем дизъюнктов Хорна с ограничениями. Некоторые подходы строят серию линейных приближений нелинейной системы (B. Kafle, J.P. Gallagher) [20, 21]; как правило, они не полны и плохо масштабируются. Более удачные попытки были предприняты в 2017-2019 годах на основе обучения с учителем для предугадывания структуры символьных моделей (P. Garg, C. Löding, P. Madhusudan, D. Neider, A. Champion, T. Chiba, N. Kobayashi, R. Sato) [9,22]. Однако они также не справляются с вышеупомянутой проблемой непредставимости моделей системы. Также существуют походы к синтаксической трансформации системы дизъюнктов, упрощающие структуру её моделей (E. De Angelis, F. Fioravanti, A. Pettorossi) [23, 24]. Такие подходы частично решают проблему непредставимости моделей, но все их реализации на сегодняшний день раскручивают отношение переходов, и вследствие этого порождают систему экспоненциального размера.

Нелинейный дизъюнкт можно рассматривать как *реляционную спецификацию* [25] корректности, т.е. спецификацию, описывающую *отношение* между входами-выходами нескольких подпрограмм, а не поведение каждой из них по отдельности. Подходы к доказательству таких свойств изучаются в области, называемой *реляционной верификацией* программ.

Один из основных подходов реляционной верификации состоит в сведении к задаче верификации функциональной спецификации одной программы путём построении *программы-произведения* (G. Barthe, J. M. Crespo) [26] с применением различных подходов к выбору отношения переходов этой программы (A.

Gurfinkel, R. Sharma, S. Shoham, Y. Vizel) [27]. Существуют подходы к реляционной верификации, основанные на синтаксических преобразованиях дизъюнктов Хорна, развиваемые группой исследователей E. De Angelis, F. Fioravanti, A. Pettorossi и M. Proietti [23,24]. В России и странах бывшего СССР реляционной верификацией занимались, в основном, в контексте проблемы эквивалентности программ. Можно выделить работы следующих исследователей: В.М. Глушкова, В.А. Захарова, А.А. Летичевского, А.А. Ляпунова, Р.И. Подловченко, В.К. Сабельфельда, Ю.И. Янова и других [28].

Решение задачи по адаптации подходов реляционной верификации с использованием PDR-подхода могло бы существенно повысить работоспособность Хорн-решателей для нелинейных систем.

Целью данной работы является разработка эффективного подхода для решения нелинейных систем дизъюнктов Хорна с ограничениями. Для её реализации были сформулированы следующие задачи.

1. Исследовать проблему представимости моделей систем дизъюнктов Хорна с ограничениями и разработать преобразование нелинейных систем, ослабляющее форму символьных моделей и упрощающее их поиск.
2. Предложить новый вид решений систем дизъюнктов Хорна, который будет представим в языке ограничений для большего множества систем, чем классические символьные модели.
3. Разработать и реализовать алгоритм автоматического построения таких решений систем дизъюнктов Хорна.
4. Провести экспериментальное исследование полученных результатов.

Соответствие диссертации паспорту специальности. Постановка цели и задач исследования соответствует следующим пунктам паспорта специальности 05.13.11: модели, методы и алгоритмы проектирования и анализа программ и программных систем, их эквивалентных преобразований, верификации и тестирования (пункт 1); языки программирования и системы программирования, семантика программ (пункт 2); программные системы символьных вычислений (пункт 5).

Методология и методы исследования. Методология исследования базируется на подходах информатики к формальной верификации программ. В работе используется формализм логики первого порядка с семантикой в стиле Тарского. Программная реализация теоретических результатов выполнена

на языке C++ на основе кодовой базы SMT-решателя Z3 и Хорн-решателя SPACER.

Основные положения, выносимые на защиту.

1. Новый подход к синтаксической синхронизации систем дизъюнктов Хорна с ограничениями первого порядка, доказательство его корректности.
2. Алгоритм CHCproduct, реализующий синхронизирующее преобразование дизъюнктов Хорна с ограничениями, доказательство его корректности, завершаемости, анализ сложности.
3. Понятие реляционного сертификата выполнимости и теорема о том, что если у системы существует реляционный сертификат выполнимости, то она выполнима.
4. Алгоритм RelRecMc для автоматического, направляемого свойством, построения реляционных сертификатов выполнимости для систем дизъюнктов Хорна с ограничениями. Доказательство его корректности.
5. Реализация алгоритмов CHCproduct и RelRecMc в SMT-решателе Z3. Экспериментальное исследование данных алгоритмов на различных тестовых примерах, включающих условия верификации свойств безопасности и реляционных проблем верификации свойств гипербезопасности.

Научная новизна результатов, полученных в рамках исследования, заключается в следующем.

- Впервые было введено и формально описано синхронизирующее преобразование дизъюнктов Хорна, а также доказана его корректность.
- Впервые введено понятие реляционного сертификата выполнимости в терминах решения нелинейных систем дизъюнктов Хорна с ограничениями.
- Предложен новый алгоритм автоматического вывода реляционных сертификатов выполнимости RelRecMc, обобщающий известный алгоритм RecMC и улучшающий его поведение на нелинейных системах.

Теоретическая и практическая значимость работы. Диссертационное исследование предлагает новый метод синхронизации нелинейных систем дизъюнктов Хорна с ограничениями, частично решающий проблему непредставимости символьных моделей систем дизъюнктов в языке ограничений.

Практическая значимость работы заключается в создании и реализации алгоритма автоматического вывода реляционных сертификатов выполнимости, который может быть использован для доказательства корректности программного кода относительно произвольных свойств безопасности или гипербезопасности первого порядка, автоматического аннотирования императивных программ в декартовой логике Хоара, для вывода уточняющих типов функциональных программ, доказательства эквивалентности программ и т.д.

Степень достоверности и апробация результатов. Достоверность и обоснованность результатов исследования обеспечивается формальными доказательствами, а также компьютерными экспериментами. Полученные результаты согласуются с результатами, установленными другими авторами.

Основные результаты работы докладывались на следующих научных конференциях и семинарах: внутренний семинар университета Вашингтона (14 декабря 2016 года, Сиэтл, США), конференция LPAR-2017 (7-12 мая 2017, Маун, Ботсвана), внутренний семинар ИСП РАН им. В.П. Иванникова (14 июня 2019, Москва, Россия), конференция ECOOP-2019 (15-19 июля 2019 г., Лондон, Великобритания), открытый семинар PSSV-2019 (1–2 июля 2019, Новосибирск, Россия), конференция FMCAD-2019 (22–25 октября 2019, Сан-Хосе, США), внутренний семинар ИСИ СО РАН (21 ноября 2019, Новосибирск, Россия), внутренний семинар ВШЭ (19 декабря 2019, Москва, Россия).

Публикации по теме диссертации. Основные результаты по теме диссертации изложены в семи печатных работах, зарегистрированных в РИНЦ. Из них две статьи изданы в журналах из “Перечня рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук”, сформированного согласно требованиям, установленным Министерством образования и науки Российской Федерации. Три статьи опубликованы в издании, входящем в базы цитирования Scopus и Web of Science.

Личный вклад автора в публикациях, выполненных с соавторами, распределён следующим образом. В работе [2] вклад автора заключается в предложении исчисления символьных куч и сведению поиска пространственных инвариантов к решению систем дизъюнктов Хорна с ограничениями; соавторы участвовали в обсуждении идей, постановке экспериментов, разработке алгоритма сведения для произвольных потоков управления. В статье [3] автор предложил формулировку понятия реляционного инварианта как решения нелиней-

ных систем дизъюнктов, разработал и реализовал алгоритм, участвовал в постановке экспериментов; соавторы участвовали в обсуждении основных идей статьи, выполняли обзор предметной области. В работах [6,7] автор представил синхронизирующее преобразования системы дизъюнктов, выполнил доказательство его корректности; соавторы предложили идею синхронизации, ставили эксперименты, участвовали в формализации и улучшении изложения идей статьи. В работах [4,5] вклад автора заключается в доказательстве неразрешимости задачи невыполнимости неэкспансивного фрагмента систем типов; соавторы формализовали задачу, участвовали в улучшении доказательства, а также предложили разрешающую процедуру для полулинейного фрагмента.

Благодарности.

В первую очередь я хотел бы поблагодарить Андрея Владимировича Иванова, Дмитрия Юрьевича Булычева и компанию JetBrains за то, что дали мне возможность сделать моё главное увлечение работой.

Я благодарен Дмитрию Владимировичу Кознову, моему руководителю, за его бесконечную энергию, постоянную опеку, его дальновидность и мудрость, которой он щедро делился на протяжении всей нашей работы. Я бесконечно признателен Григорию Геннадьевичу Федюковичу за то, что он дал мне возможность познать красоту современных формальных методов и за руководство моей работой на первых этапах. Я благодарен Владимиру Анатольевичу Захарову за ценные обсуждения, которые повлияли на эту работу.

Я благодарен кафедре системного программирования СПбГУ, в особенности Андрею Николаевичу Терехову и Якову Александровичу Кириленко за привитие интереса к формальным методам и помощь в формировании нашего коллектива. Я счастлив работать со студентами К.А. Батоевым, М.П. Костицыным, Ю.О. Костюковым и А.В. Мисонижником, энергия и любознательность которых меня не перестаёт удивлять. Они оказали большое влияние на мою работу. Я также признателен всем студентам, с которыми мне посчастливилось работать в прошлом, особенно Г.А. Зимину.

Я благодарен Д.С. Косареву за его поддержку в тяжёлых ситуациях. Я испытываю чувство глубочайшей благодарности своей семье, в особенности моим родителям, Мордвиновым Людмиле Петровне и Александру Владимировичу, благодаря которым я стал тем, кто я есть. Их любовь и поддержка на всех этапах моей жизни сделали меня счастливым человеком и дали энергию для всего, что я делаю.

Глава 1. Обзор предметной области

В данной главе вводятся основные понятия, использующиеся в данном диссертационном исследовании: языки ограничений, системы дизъюнков Хорна с ограничениями, символьные модели систем дизъюнктов и др., а также приводятся примеры. Рассматриваются также существующие способы автоматического построения символьных моделей систем дизъюнктов. В заключении главы приводятся выводы о состоянии предметной области.

1.1 Языки ограничений

Дизъюнкты Хорна с ограничениями являются центральным объектом изучения *логического программирования в ограничениях* (constraint logic programming, CLP) [5]. Данная область появилась в результате слияния двух областей: логического программирования и удовлетворения ограничений (constraint solving) [29].

В этом разделе обсуждаются основные положения области удовлетворения ограничений: вводятся понятие языка ограничений, выполнимости формул в нём, приводятся примеры языков ограничений и инструментов решения систем ограничений. Логическое программирование в ограничениях обсуждается в следующем разделе.

1.1.1 Синтаксис языка ограничений

Сигнатурой назовём тройку $\Sigma \stackrel{\text{def}}{=} \langle \Sigma^f, \Sigma^p, ar \rangle$, где Σ^f — множество *функциональных символов*, Σ^p — множество *предикатных символов*, $ar : \Sigma^f \cup \Sigma^p \to \mathbb{N}$ — функция *местности* символов, а $\Sigma^f \cap \Sigma^p = \varnothing$. Будем говорить, что Σ является *сигнатурой с равенством*, если в множество предикатных символов входит символ «=» такой, что $ar\,(=) = 2$.

Пусть $\mathcal{V}$ — некоторое множество такое, что $\mathcal{V} \cap \Sigma^f = \varnothing$ и $\mathcal{V} \cap \Sigma^p = \varnothing$. Тогда будем называть элементы $\mathcal{V}$ *предметными переменными*.

Определим *термы* сигнатуры Σ следующим образом.

1. Предметная переменная является термом.
2. Если $f \in \Sigma^f$, $n = ar\,(f)$, $t_1, \ldots, t_n$ — термы, то $f(t_1, \ldots, t_n)$ тоже является термом.

Если $p \in \Sigma^p$, $n = ar\,(p)$ и $t_1, \ldots, t_n$ являются термами, то будем называть $p(t_1, \ldots, t_n)$ *атомарной формулой* (или просто *атомом*). Иногда вместо $p(t_1, \ldots, t_n)$ будем писать $p(\overline{t})$, где $\overline{t} = \langle t_1, \ldots, t_n \rangle$.

Множество *формул сигнатуры* Σ (далее — просто Σ-*формул*) определим следующим образом.

1. Любая атомарная формула является формулой.
2. Если φ — формула, то $\neg\varphi$ — формула.
3. Если φ, ψ — формулы, то $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$ — формулы.
4. Если φ — формула, $x \in \mathcal{V}$, то $\forall x.\varphi$, $\exists x.\varphi$ — формулы.

Назовём формулу *бесквантворной*, если в неё не входят кванторы (символы $\forall$ и $\exists$). Будем обозначать формулы буквами φ, ψ, η (возможно, с индексами). Будем называть вхождение предметной переменной x в формулу φ *связанным*, если оно входит в область действия кванторной приставки $\forall x$ или $\exists x$. Будем называть предметную переменную x *свободной переменной* формулы φ, если хотя бы одно её вхождение не является связанным. Если $x_1, \ldots, x_n \in \mathcal{V}$, то $\varphi(x_1, \ldots, x_n)$ обозначает формулу φ, у которой каждая свободная переменная является элементом множества $\{x_1, \ldots, x_n\}$.

Если $x_1, \ldots, x_n$ — все свободные переменные формулы φ, то через $\forall\varphi$ будем обозначать *универсальное замыкание* φ, т.е. формулу $\forall x_1 \ldots \forall x_n.\varphi(x_1, \ldots, x_n)$. Подстановку термов $\overline{t}$ вместо свободных переменных $\overline{x}$ формулы обозначим $\varphi[\overline{t}/\overline{x}]$.

Языком первого порядка сигнатуры Σ назовём множество Σ-формул. *Предложением языка* $\mathcal{A}$ будем называть формулу языка $\mathcal{A}$ без свободных переменных.

Зафиксируем сигнатуру Σ с равенством. Будем называть язык $\mathcal{A}$ первого порядка сигнатуры Σ *языком ограничений*. Будем называть бесквантворные формулы языка ограничений *ограничениями*. Обозначим через $\mathcal{A}(x_1, \ldots, x_n)$ множество ограничений со свободными переменными из множества $\{x_1, \ldots, x_n\}$.

1.1.2 Семантика языка ограничений

В данной диссертации используется стандартный способ задания семантики языков первого порядка в стиле Тарского [30–32].

Интерпретацией языка $\mathcal{A}$ называется упорядоченная пара $\mathcal{M} \stackrel{\text{def}}{=} \langle |\mathcal{M}|, \mu \rangle$, где $|\mathcal{M}|$ — непустое множество, называемое *носителем* интерпретации, а μ — отображение, которое обладает следующими свойствами.

1. Каждому n-местному функциональному символу (т.е. такому $f \in \Sigma^f$, что $ar(f) = n$) при $n > 0$ сопоставляет функцию из $|\mathcal{M}|^n$ в $|\mathcal{M}|$, при $n = 0$ — элемент множества $|\mathcal{M}|$.
2. Символу равенства сопоставляет бинарное отношение $\{\langle x,x \rangle \mid x \in |\mathcal{M}|\}$.
3. Каждому другому n-местному предикатному символу при $n > 0$ сопоставляет n-местное отношение на $|\mathcal{M}|$, при $n = 0$ — истинностное значение.

Выполнимость формул языка $\mathcal{A}$ в интерпретации $\mathcal{M}$ определяется стандартно [30–32]. Будем обозначать выполнимость формулы $\varphi(x_1, \ldots, x_n)$ в $\mathcal{M}$ при оценке свободных переменных x_i элементами носителя a_i следующим образом:

$$\mathcal{M} \models \varphi(a_1, \ldots, a_n).$$

Если φ — предложение языка $\mathcal{A}$ и $\mathcal{M} \models \varphi$, то будем говорить, что $\mathcal{M}$ является *моделью* φ. $\mathcal{M}$ является моделью множества предложений Γ, если $\mathcal{M} \models \varphi$ для всех $\varphi \in \Gamma$. Будем называть предложение φ *общезначимым* и обозначать этот факт как $\models \varphi$, если любая его интерпретация является моделью. Будем говорить, что φ *равносильно* ψ, обозначая $\varphi \sim \psi$, если $\mathcal{M} \models \varphi$ тогда и только тогда, когда $\mathcal{M} \models \psi$. Предложение ψ является *логическим следствием* предложения φ, если любая модель φ является моделью ψ; этот факт будем обозначать $\varphi \models \psi$.

Для Σ-формулы $\varphi(x_1, \ldots, x_n)$, где $x_1, \ldots, x_n$ — свободные переменные, обозначим через $\mathcal{M}(\varphi)$ множество оценок свободных переменных, выполняющих φ в $\mathcal{M}$, т.е.

$$\mathcal{M}(\varphi) \stackrel{\text{def}}{=} \{\langle a_1, \ldots, a_n \rangle \mid \mathcal{M} \models \varphi(a_1, \ldots, a_n)\}.$$

Пусть 2^X — булеан множества X. Договоримся, что запись 2^{X^n} обозначает $2^{(X^n)}$. В частности, для формулы $\varphi(x_1, \ldots, x_n)$ справедливо следующее:

$$\mathcal{M}(\varphi) \in 2^{|\mathcal{M}|^n}.$$

Будем говорить, что n-арное отношение $R \in 2^{|\mathcal{M}|^n}$ *представимо* в интерпретации $\mathcal{M}$, если существует формула $\varphi \in \mathcal{A}(x_1, \ldots, x_n)$ такая, что для некоторых попарно различных $x_1, \ldots, x_n \in \mathcal{V}$ выполняется $\mathcal{M}(\varphi) = R$. В противном

случае R *непредставимо* в $\mathcal{M}$. Если интерпретация $\mathcal{M}$ ясна из контекста, то будем также говорить, что R представимо или непредставимо в языке $\mathcal{A}$.

1.1.3 Теории в языке ограничений

В отдельных случаях бывает необходимым рассматривать выполнимость ограничений в некоторой *теории*, а не в отдельно взятой интерпретации. В этом случае в данной работе используется подход из области SMT [1]: такая теория определяется как класс моделей вместо классического теоретико-модельного определения теории как множества предложений.

Будем называть *теорией* $\mathcal{T}$ языка $\mathcal{A}$ некоторый класс Σ-интерпретаций. *Противоречивой* теорией назовём пустой класс интерпретаций. Если существует множество предложений языка $\mathcal{A}$, класс моделей которого совпадает с $\mathcal{T}$, то назовём такое множество *аксиоматизацией* $\mathcal{T}$. Элементы $\mathcal{T}$ будем называть *моделями теории* $\mathcal{T}$.

Σ-предложение φ *выполнимо в теории* $\mathcal{T}$, если существует интерпретация, которая является моделью $\mathcal{T}$ и моделью φ одновременно; в противном случае говорят, что предложение *невыполнимо в теории* $\mathcal{T}$. Предложение φ *общезначимо в теории* $\mathcal{T}$, если любая модель этой теории является также и моделью φ (этот факт обозначается как $\mathcal{T} \models \varphi$).

1.1.4 SMT-решатели

На практике для решения ограничений широко используются *SMT-решатели* (Satisfiability Modulo Theory, [1]) — высокопроизводительные инструменты проверки выполнимости формул первого порядка в различных теориях. На вход SMT-решателю поступает формула, использующая символы каких-либо теорий, поддерживаемых решателем. В случае остановки решатель выдаёт один из трёх ответов: формула выполнима (`sat`), невыполнима (`unsat`) или неизвестно (`unknown`). В случае ответа `sat` решатель выдаёт модель, в случае ответа `unsat` — доказательство невыполнимости формулы (как правило, резолютивное), в случае ответа `unknown` — объяснение причины этого ответа.

Существует большое количество высокопроизводительных SMT-решателей, самыми известными являются Z3 [11], CVC4 [33], Yices [34] и

Boolector [35]. Среди SMT-решателей проводятся ежегодные соревнования [36].

Ядром SMT-решателя является какой-либо алгоритм решения задачи SAT выполнимости булевых формул. Как правило, современные SMT-решатели используют алгоритм CDCL (Conflict Driven Clause Learning) [37, 38]. Несмотря на NP-полноту задачи SAT, современные реализации алгоритма CDCL показывают хорошие результаты на практике [39, 40].

В контексте данной работы интерес представляют алгоритмы проверки выполнимости бесквантворных формул, т.к. по определению ограничение — формула без кванторов. Пусть на вход SMT-решателю поступает ограничение φ. С помощью CDCL-алгоритма решатели эффективно перебирают различные модели *пропозициональной абстракции* данной формулы φ_{prop}. Пропозициональная абстракция формулы φ получается заменой всех атомарных формул в φ на нульместные предикатные символы. Если пропозициональная абстракция невыполнима, то невыполнима и исходная формула. В противном случае каждая модель-кандидат φ_{prop} декодируется обратно в бесквантворную конъюнкцию исходных литералов теорий. Если такая конъюнкция выполнима в теории, то выполнима и исходная формула, в противном случае решатель переходит к следующей модели.

Задача выполнимости бесквантворных конъюнкций литералов решается в SMT-решателях с помощью подключаемых модулей, называемых *решателями теорий*. Ниже описаны некоторые языки и теории ограничений, используемые в данной работе в качестве примеров, для которых реализованы решатели теорий в подавляющем большинстве SMT-решателей.

Линейная целочисленная арифметика. Язык первого порядка над сигнатурой $\{0,1,+,-,\leqslant\}$ называют языком *линейной целочисленной арифметики*. Выполнимость арифметических ограничений определяются в *стандартной модели арифметики*, т.е. интерпретации $\mathcal{N}$, носителем которой является множество целых чисел $\mathbb{Z}$ со стандартными арифметическими интерпретациями символов. Таким образом *теорией линейной целочисленной арифметики* назовём теорию $\mathcal{T}_{LIA}$, состоящую из единственной интерпретация $\mathcal{N}$[1].

[1]Часто теорией целочисленной арифметики называют множество предложений-аксиом Пеано. Однако по теореме Лёвенгейма-Скулема [41] помимо стандартной модели есть бесконечное множество нестандартных моделей, в то время, как SMT-решатели фактически проверяют выполнимость

Важным свойством линейной целочисленной арифметики является её разрешимость. Хотя выполнимость в бесквантором фрагменте линейной целочисленной арифметике полиномиальна [42], на практике наиболее эффективны алгоритмы, основанные на экспоненциальном симплекс-методе [43, 44].

Нелинейная целочисленная арифметика получается путем добавления в сигнатуру линейной арифметики умножения. Нелинейная арифметика, в отличие от линейной, неразрешима [19], поэтому на практике в качестве языка ограничений имеет смысл использовать именно линейную арифметику.

Экстенсиональная теория массивов. Рассмотрим язык первого порядка $\mathcal{A}_{Arr}$ над сигнатурой $\{read, write, =\}$, где $read$ и $write$ — это функциональные символы арности 2 и 3 соответственно. *Экстенсиональная теория массивов* — это теория $\mathcal{T}_{Arr}$ с системой аксиом $\{Ax_1, Ax_2, Ext\}$:

$$
\begin{aligned}
Ax_1 &\equiv \forall a, i, v.\ read(write(a, i, v), i) = v \\
Ax_2 &\equiv \forall a, i, j, v.\ i \neq j \rightarrow read(write(a, i, v), j) = read(a, j) \\
Ext &\equiv \forall a, b.\ \big(\forall i.\, read(a, i) = read(b, i)\big) \rightarrow a = b
\end{aligned}
$$

Неформально говоря, $read(a, i)$ обозначает чтение массива a по индексу i, $write(a, i, v)$ обозначает массив, полученный из массива a заменой значения в ячейке с индексом i на значение v. Ax_1 и Ax_2 называются *функциональными аксиомами Маккарти* [45] и определяют правила чтения массивов после записи. Аксиома экстенсинальности Ext задаёт равенство на массивах.

Выполнимость формул без кванторов в теории массивов разрешима, но NP-полна [46]. Однако для её решения на практике существуют эффективные алгоритмы [46, 47], использующие алгоритмы вычисления конгруэнтного замыкания отношений [48]. Выполнимость произвольных формул языка $\mathcal{A}_{Arr}$ неразрешима [47].

Теория алгебраических типов данных. *Алгебраический тип данных* определяется множеством конструкторов $C_1, \ldots, C_n$, аргументами которых могут быть снова термы алгебраических типов данных. Примерами алгебраиче-

только в стандартной модели. По этой причине здесь принят подход к определению теории как класса моделей.

ских типов данных являются списки и деревья:

$$\begin{array}{ll} List(T) := & Tree(T) := \\ \quad \mid nil : List(T) & \quad \mid leaf : Tree(T) \\ \quad \mid cons : T \times List(T) \to List(T) & \quad \mid node : T \times Tree(T) \times Tree(T) \to Tree(T) \end{array}$$

Пусть дан алгебраический тип D с конструкторами $C_1, \ldots, C_n$, язык первого порядка $\mathcal{A}$ с сигнатурой Σ с функциональными символами $C_1, \ldots, C_n$ и единственным предикатным символом равенства[2].

Теория $\mathcal{T}_D$ алгебраического типа D состоит только из Эрбрановской интерпретации $\mathcal{H}$ данной сигнатуры. В этой интерпретации носителем является множество термов, не содержащих переменные, а функциональные символы интерпретируются сами собой, т.е. k-местный символ C интерпретируется функцией, сопоставляющей термам $t_1, \ldots, t_k$ терм $C(t_1, \ldots, t_k)$.

Выполнимость ограничений в $\mathcal{T}_D$ разрешима, NP-полна [50], но существуют эффективные на практике разрешающие процедуры [49, 51, 52].

Другие теории. Среди других теорий, поддерживаемых большинством современных SMT-решателей, но не рассматриваемых в данной диссертационной работе подробно, на практике применяются теория линейной рациональной арифметики [53], вещественной арифметики [54], теория неинтерпретированных функций с равенством (т.е. теория, аксиоматизируемая пустым множеством предложений), теория битовых векторов [55], а также теория строк [56].

Комбинирование теорий. Пусть даны два языка ограничений с непересекающимися сигнатурами Σ_1 и Σ_2 с равенством (т.е. $\Sigma_1^f \cap \Sigma_2^f \equiv \varnothing$ и $\Sigma_1^p \cap \Sigma_2^p \equiv \{=\}$) и две теории $\mathcal{T}_1$ и $\mathcal{T}_2$ сигнатур Σ_1 и Σ_2 соответственно. Пусть $\mathcal{A}$ — язык над сигнатурой $\Sigma_1 \cup \Sigma_2$. Будем говорить, что предложение $\varphi \in \mathcal{A}$ *выполнимо в комбинации теорий* $\mathcal{T}_1$ *и* $\mathcal{T}_2$, если существует $\Sigma_1 \cup \Sigma_2$-интерпретация $\mathcal{M}$ и модели $\mathcal{M}_1$ и Σ_2 теорий $\mathcal{T}_1$ и $\mathcal{T}_2$ соответственно, такие, что $|\mathcal{M}_i| \subseteq |\mathcal{M}|$, для всех символов сигнатуры Σ_i сужение интерпретации этого символа в $\mathcal{M}$ на $|\mathcal{M}_i|$ совпадает с интерпретацией этого символа в $\mathcal{M}_i$ для $i = 1{,}2$ (очевидно, для единственного общего символа равенства это всегда верно). Если предложение φ выполнимо

[2]Стандартный способ обеспечить «типизированность» конструкторов заключается в использовании *многосортных языков первого порядка* [49]. Но в данной работе эти детали не будут приниматься во внимание: будет считаться, что если терм не «типизируется», то формулы, в которые он входит, являются невыполнимыми.

в любой такой интерпретации $\mathcal{M}$, то оно *общезначимо в комбинации* $\mathcal{T}_1$ *и* $\mathcal{T}_2$ (обозн. $\mathcal{T}_1 \cup \mathcal{T}_2 \models \varphi$).

SMT-решатели могут определять выполнимость формул в комбинации теорий. Это позволяет, к примеру, комбинировать ограничения, использующие арифметику и теорию массивов одновременно. В основном, для этого используются реализации методов Нельсона-Оппена [57] и Шостака [58]. Таким образом, SMT-решатели способны определять, например, выполнимость ограничений над списками целых чисел с арифметическими ограничениями на элементы, или на массивы деревьев со строками в узлах.

1.2 Дизъюнкты Хорна с ограничениями

Далее будем считать фиксированными некоторый язык ограничений $\mathcal{A}$ сигнатуры Σ и соответствующую теорию $\mathcal{T}$. В примерах в качестве теории ограничений будут использоваться теория линейной целочисленной арифметики, экстенсиональная теория массивов, теория списков, деревьев, либо какая-то их комбинация.

1.2.1 Синтаксис и выполнимость

Пусть $\mathcal{R} = \{P_1, \ldots, P_n\}$ — конечное множество предикатных символов, которые будем называть *неинтерпретированными символами*. Положим также, что $\Sigma^p \cap \mathcal{R} = \varnothing$. Атомарные формулы вида $P_i(\overline{x})$, где $\overline{x}$ — кортеж предметных переменных, будем называть *неинтерпретированными атомами*.

Определение 1. *Дизъюнкт Хорна с ограничением* (далее — *дизъюнкт Хорна* или просто *дизъюнкт*) является $\Sigma \cup \mathcal{R}$-формулой вида

$$\varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m) \rightarrow H,$$

где φ — бескванторная Σ-формула, $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)$ — неинтерпретированные атомы, H — либо неинтерпретированный атом $R(\overline{x})$, либо $\bot$. Посылка импликации называется *телом дизъюнкта* C и обозначается $body(C)$, заключение H называется *заголовком дизъюнкта* и обозначается $head(C)$.

Определение 2. *Система дизъюнктов Хорна с ограничениями* $\mathcal{S}$ (далее — просто *система* $\mathcal{S}$ или *система*) — это конечное множество дизъюнктов Хорна.

Множество *запросов* системы является множеством дизъюнктов, заголовок которых состоит только из символа $\bot$:

$$queries_{\mathcal{S}} \stackrel{\text{def}}{=} \{C \in \mathcal{S} \mid head(C) = \bot\}.$$

Будем считать, если дизъюнкты $C_i, C_j \in \mathcal{S}$ имеют заголовки $R(\overline{x})$ и $R(\overline{x}')$ соответственно, то $\overline{x} = \overline{x}'$. Другими словами, будем считать, все дизъюнкты, озаглавленные одним неинтерпретированным символом R, имеют одинаковые заголовки[3]. Переменные в заголовке таких дизъюнктов обозначим обозначим как $\overline{v}_R$, т.е. можно считать, что если дизъюнкт C озаглавлен символом R, то $head(C) = R(\overline{v}_R)$.

Множеством *правил* для неинтерпретированного символа R назовём множество дизъюнктов с заголовком $R(\overline{v}_R)$:

$$rules_{\mathcal{S}}(R) \stackrel{\text{def}}{=} \{C \in \mathcal{S} \mid head(C) = R(\overline{v}_R)\}.$$

Когда из контекста будет ясно, о какой конкретно системе идёт речь, множество правил для символа R и соответствующих запросов будем обозначать просто $rules(R)$ и $queries$ соответственно.

Телом неинтерпретируемого символа назовём дизъюнкцию тел его правил:

$$body(R) \stackrel{\text{def}}{=} \bigvee_{C \in rules(R)} body(C).$$

Экзистенциальные (или *локальные*) переменные R — это свободные переменные $body(R)$, не входящие в переменные заголовка $\overline{v}_R$. Экзистенциальные переменные R обозначим за $\overline{\ell}_R$. Аналогично, экзистенциальные переменные дизъюнкта C — это его свободные переменные, не входящие в заголовок.

Дизъюнкт, тело которого является ограничением (т.е. не содержит неинтерпретированных символов), будем называть *ограниченным фактом*.

Определение 3. *Система дизъюнктов линейна*, если каждый дизъюнкт в ней *линеен*, то есть в его тело входит не более одного неинтерпретированного атома.

Определение 4. *Система дизъюнктов нелинейна*, если в ней имеются *нелинейные* дизъюнкты, т.е., в их тела входит более одного неинтерпретированного атома.

[3]Это допущение не ограничивает общность: если оно не выполняется, просто переименуем переменные.

Пример 1. Пусть $\mathcal{T}$ — комбинация теорий линейной целочисленной арифметики и теории бинарных деревьев. Рассмотрим следующую систему:

$$
\begin{aligned}
T = \mathit{leaf} \wedge n = 0 &\to \mathit{size}(T, n) \\
T = \mathit{node}(v, L, R) \wedge n = 1 + n^L + n^R \wedge \mathit{size}(L, n^L) \wedge \mathit{size}(R, n^R) &\to \mathit{size}(T, n) \\
T = \mathit{leaf} \wedge s = 0 &\to \mathit{sum}(T, s) \\
T = \mathit{node}(v, L, R) \wedge s = v + s^L + s^R \wedge & \\
\mathit{sum}(L, s^L) \wedge \mathit{sum}(R, s^R) &\to \mathit{sum}(T, s) \\
T = \mathit{leaf} \wedge U = \mathit{leaf} &\to \mathit{inc}(T, U) \\
T = \mathit{node}(v, L, R) \wedge U = \mathit{node}(v+2, L', R') \wedge & \\
\mathit{inc}(L, L') \wedge \mathit{inc}(R, R') &\to \mathit{inc}(T, U) \\
s' \neq s + 2n \wedge \mathit{size}(T, n) \wedge \mathit{sum}(T, s) \wedge \mathit{inc}(T, T') \wedge \mathit{sum}(T', s') &\to \bot
\end{aligned}
$$

Здесь $\mathcal{R} = \{\mathit{size}, \mathit{sum}, \mathit{inc}\}$ и $\overline{v}_{\mathit{size}} = \{T, n\}$, $\overline{\ell}_{\mathit{size}} = \{v, L, R, n^L, n^R\}$, $\mathit{body}(\mathit{size}) = (T = \mathit{leaf} \wedge n = 0) \vee \big(T = \mathit{node}(v, L, R) \wedge n = 1 + n^L + n^R \wedge \mathit{size}(L, n^L) \wedge \mathit{size}(R, n^R)\big)$.

Пусть $\mathcal{M}$ — Σ-интерпретация. Пусть $\overline{X} = \langle X_1, \ldots, X_n \rangle$ — кортеж отношений на носителе $|\mathcal{M}|$, где $X_i \subseteq |\mathcal{M}|^{ar(P_i)}$. Обозначим обогащение $\mathcal{M}$ вида $\mathcal{M}\{P_1 \mapsto X_1, \ldots, P_n \mapsto X_n\}$ через $(\mathcal{M}, \overline{X})$.

Определение 5. Система дизъюнктов $\mathcal{S}$ называется *выполнимой в интерпретации* $\mathcal{M}$ теории ограничений $\mathcal{T}$, если существует кортеж отношений $\overline{X}$ такой, что

$$(\mathcal{M}, \overline{X}) \models \bigwedge_{C \in \mathcal{S}} \forall C.$$

В противном случае система называется *невыполнимой в интерпретации* $\mathcal{M}$.

Определение 6. Будем говорить, что *система дизъюнктов выполнима*, если она выполнима в каждой модели $\mathcal{M}$ теории ограничений $\mathcal{T}$.

В противном случае будем говорить, что *система дизъюнктов невыполнима*. Другими словами, система невыполнима, если существует модель $\mathcal{M}$ теории $\mathcal{T}$, в которой система невыполнима.

Пример 2. Рассмотрим следующую систему дизъюнктов с ограничениями в линейной целочисленной арифметике:

$$x=0 \wedge z=y \rightarrow sum(x,y,z)$$
$$x>0 \wedge x'=x-1 \wedge z=z'+1 \wedge sum(x',y,z') \rightarrow sum(x,y,z)$$
$$x=x' \wedge y=y' \wedge z\neq z' \wedge sum(x,y,z) \wedge sum(x',y',z') \rightarrow \bot$$

Эта система выполнима, т.к. формулы дизъюнктов выполняются в $(\mathcal{N}, X)$, где

$$X \stackrel{\text{def}}{=} \{\langle a,b,c\rangle \in \mathbb{Z}^3 \mid c=a+b\}$$

Пример 3. Система дизъюнктов $\{C_1,C_2,Q\}$ невыполнима, т.к. не выполняется ни в одном расширении $\mathcal{N}$ (это будет доказано ниже):

$$x=0 \wedge z=y \rightarrow sum(x,y,z) \qquad \equiv C_1$$
$$x>0 \wedge x'=x-1 \wedge z=z'+1 \wedge sum(x',y,z') \rightarrow sum(x,y,z) \qquad \equiv C_2$$
$$x=1 \wedge y=z \wedge sum(x,y,z) \rightarrow \bot \qquad \equiv Q$$

1.2.2 Сертификаты выполнимости

Отношения X_i, интерпретирующие символы P_i (см. определение 5), могут содержать бесконечное количество элементов, как в примере 2. Поэтому автоматическая проверка выполнимости систем дизъюнктов методом поэлементного построения X_i невозможна. Вместо этого используется *символьное представление* отношений: k-арное отношение, интерпретирующее символ $P \in \mathcal{R}$, представляется *формулой языка* $\mathcal{A}$ с k свободными переменными (как правило, бескванторной).

Определение 7. *Символьная интерпретация* — это произвольное отображение $\mathcal{I}$ следующего вида:

$$\mathcal{R} \rightarrow \mathcal{A}(\overline{v}_P).$$

Для бескванторной $\Sigma \cup \mathcal{R}$-формулы Φ определим $[\![\Phi]\!]_{\mathcal{I}}$ как ограничение, полученное одновременной заменой всех неинтерпретированных атомов в Φ фор-

мулами их $\mathcal{I}$-образов:

$$\begin{aligned}
[\![\varphi]\!]_{\mathcal{I}} &\stackrel{\text{def}}{=} \varphi, && \text{если } \varphi \text{ — ограничение}\\
[\![P(\overline{x})]\!]_{\mathcal{I}} &\stackrel{\text{def}}{=} \mathcal{I}(P)[\overline{x}/\overline{v}_P], && \text{если } P \in \mathcal{R}\\
[\![\Phi_1 \circ \Phi_2]\!]_{\mathcal{I}} &\stackrel{\text{def}}{=} [\![\Phi_1]\!]_{\mathcal{I}} \circ [\![\Phi_1]\!]_{\mathcal{I}} && \text{для } \circ \in \{\wedge, \vee, \to, \leftrightarrow\}\\
[\![\neg\Phi]\!]_{\mathcal{I}} &\stackrel{\text{def}}{=} \neg\, [\![\Phi]\!]_{\mathcal{I}}\\
[\![\mathcal{Q}x.\Phi]\!]_{\mathcal{I}} &\stackrel{\text{def}}{=} \mathcal{Q}x.\, [\![\Phi]\!]_{\mathcal{I}}\,, && \text{если } \mathcal{Q} \in \{\forall,\exists\} \text{ и } x \text{ не входит}\\
& && \text{в элементы образа } \mathcal{I} \text{ свободно}
\end{aligned}$$

Лемма 1. Пусть $\mathcal{M}$ — Σ-интерпретация, $\mathcal{I}$ — символьная интерпретация, Φ — $\Sigma \cup \mathcal{R}$-формула со свободными переменными $x_1, \ldots, x_k$, $\overline{a} \in |\mathcal{M}|^k$, $\overline{X} = \{X_1, \ldots, X_n\}$, где $X_i = \mathcal{M}(\mathcal{I}(P_i))$. Тогда справедливо следующее утверждение:

$$\left(\mathcal{M}, \overline{X}\right) \models \Phi(\overline{a}) \text{ тогда и только тогда, когда } \mathcal{M} \models [\![\Phi]\!]_{\mathcal{I}}(\overline{a}).$$

Доказательство. Выполним доказательство индукцией по структуре формулы Φ. Если Φ не содержит неинтерпретированных символов, то справедливо следующее утверждение:

$$\mathcal{M} \models [\![\Phi]\!]_{\mathcal{I}}(\overline{a}) \quad \Leftrightarrow \quad \mathcal{M} \models \Phi(\overline{a}) \quad \Leftrightarrow \quad \left(\mathcal{M}, \overline{X}\right) \models \Phi(\overline{a}).$$

Если $\Phi \equiv \neg\Phi'$, то по индукционному предположению $\left(\mathcal{M}, \overline{X}\right) \models \Phi'(\overline{a}) \Leftrightarrow \mathcal{M} \models [\![\Phi']\!]_{\mathcal{I}}(\overline{a})$. Тогда в силу закона исключённого третьего имеем:

$$\mathcal{M} \models [\![\Phi]\!]_{\mathcal{I}}(\overline{a}) \quad \Leftrightarrow \quad \mathcal{M} \models \neg\, [\![\Phi']\!]_{\mathcal{I}}(\overline{a}) \quad \Leftrightarrow \quad \left(\mathcal{M}, \overline{X}\right) \models \neg\Phi'(\overline{a}).$$

Случаи, когда $\Phi \equiv \Phi_1 \circ \Phi_2$ для $\circ \in \{\wedge, \vee, \to, \leftrightarrow\}$ и $\Phi \equiv \mathcal{Q}x.\Phi'$ для $\mathcal{Q} \in \{\forall,\exists\}$, доказываются аналогично.

Наконец, для $\Phi \equiv P_i(\overline{x})$ имеем:

$$\mathcal{M} \models [\![\Phi]\!]_{\mathcal{I}}(\overline{a}) \Leftrightarrow \mathcal{M} \models \mathcal{I}(P_i)(\overline{a}) \Leftrightarrow \overline{a} \in \mathcal{M}(\mathcal{I}(P_i)) \Leftrightarrow \overline{a} \in X_i \Leftrightarrow \left(\mathcal{M}, \overline{X}\right) \models P_i(\overline{a})$$

□

Определение 8. Символьная интерпретация $\mathcal{I}$ является *сертификатом выполнимости* системы $\mathcal{S}$, если

$$\mathcal{T} \models \bigwedge_{C \in \mathcal{S}} \forall\, [\![C]\!]_{\mathcal{I}}\,.$$

Эквивалентно, символьная интерпретация $\mathcal{I}$ является сертификатом выполнимости, если она *безопасна* и *индуктивна*:

$$\text{для всех } C \in \mathit{queries},\ \mathcal{T} \models \forall\, [\![C]\!]_{\mathcal{I}} \qquad \text{(безопасность)}$$
$$\text{для всех } P \in \mathcal{R},\ \mathcal{T} \models \forall\, ([\![\mathit{body}(P)]\!]_{\mathcal{I}} \to \mathcal{I}(P)) \qquad \text{(индуктивность)}$$

Сертификаты выполнимости также называют *символьными моделями* системы.

Пример 4. У системы в примере 2 существует сертификат выполнимости

$$\mathcal{I} \equiv \{\mathit{sum} \mapsto z = x + y\}.$$

Утверждение 1. Если существует сертификат выполнимости системы $\mathcal{S}$, то $\mathcal{S}$ выполнима.

Доказательство. По определению выполнимости в теории $\mathcal{T}$, а также по определению сертификата выполнимости, имеем $\mathcal{M} \models [\![C]\!]_{\mathcal{I}}(\overline{a})$ для всех моделей $\mathcal{M}$ теории $\mathcal{T}$, дизъюнктов $C \in \mathcal{S}$ и кортежей $\overline{a} \in |\mathcal{M}|^k$, где k — количество свободных переменных в C. Положим $\overline{X} = \{X_1, \ldots, X_n\}$, где $X_i = \mathcal{M}(\mathcal{I}(P_i))$. По лемме 1 имеем $(\mathcal{M}, \overline{X}) \models C(\overline{a})$ для всех $\overline{a} \in |\mathcal{M}|^k$ и всех дизъюнктов C, а значит

$$(\mathcal{M}, \overline{X}) \models \bigwedge_{C \in \mathcal{S}} \forall C.$$

Следовательно $\mathcal{S}$ выполнима во всех моделях $\mathcal{T}$. □

1.2.3 Невыполнимые системы и резолютивные опровержения

Итак, сертификаты выполнимости представляют в языке ограничений модели систем дизъюнктов, т.е. являются «синтаксическими свидетельствами» выполнимости множества дизъюнктов. Аналогичным инструментом для работы с невыполнимыми системами являются деревья резолютивных опровержений.

Определение 9. Пусть $C \in \mathcal{S}$, $\mathit{body}(C) \equiv \varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m)$. Пусть $D_1, \ldots, D_m$ — ограниченные факты вида следующего вида:

$$D_i \equiv \varphi_i \to R_i(\overline{v}_{R_i}).$$

Без потери общности будем считать, что экзистенциальные переменные всех дизъюнктов $C, D_1, \ldots, D_n$ попарно различны (иначе переименуем их). Тогда *гиперрезольвентой* C *и* $D_1, \ldots, D_m$ называется дизъюнкт следующего вида:

$$\varphi \wedge \varphi_1[\overline{x}_1/\overline{v}_{R_1}] \wedge \ldots \wedge \varphi_m[\overline{x}_m/\overline{v}_{R_m}] \to head(C).$$

Иными словами, гиперрезольвента дизъюнкта C и множества ограниченных фактов — это ограниченный факт, получающийся заменой всех неинтерпретированных атомов в теле C ограничениями фактов с соответствующим переименованием переменных.

Достаточно известен и очевиден следующий факт [59, 60].

Утверждение 2. Гиперрезольвента H дизъюнктов $C, D_1, \ldots, D_n$ является их логическим следствием, т.е. $\models \forall C \wedge \forall D_1 \wedge \ldots \wedge \forall D_n \to \forall H$.

Определение 10. *Выводом системы* $\mathcal{S}$ *снизу-вверх* называется дерево, каждый узел которого принадлежит множеству $\mathcal{S} \times \mathcal{A}$. Листьями этого дерева являются пары $(F, body(F))$, где $F \in \mathcal{S}$ — ограниченный факт. Узел (C,φ) имеет детей $(D_1,\varphi_1), \ldots, (D_m,\varphi_m)$, если $\varphi \to head(C)$ является гиперрезольвентой дизъюнктов C и $\varphi_1 \to head(D_1), \ldots, \varphi_m \to head(D_m)$.

Лемма 2. Пусть дано дерево вывода с корнем (C,φ). Тогда дизъюнкт $\varphi \to head(C)$ является логическим следствием системы $\mathcal{S}$.

Доказательство. Утверждение доказывается индукцией по высоте дерева вывода h, используя утверждение 2.

База $h = 0$**.** Если дерево состоит из единственной вершины, то корень является листом $(F,body(F))$, где F — ограниченный факт. В таком случае, дизъюнкт $(body(F) \to head(F)) \equiv F$ является логическим следствием $\mathcal{S}$ как её элемент.

Переход. Пусть индукционное предположение выполняется для поддеревьев с корнями в детях $(D_1,\varphi_1), \ldots, (D_m,\varphi_m)$, т.е. дизъюнкт $\varphi_i \to D_i$ является логическим следствием $\mathcal{S}$ для всех $i \in \{1, \ldots, m\}$. По определению дерева вывода дизъюнкт $\varphi \to head(C)$ является гиперрезольвентой $C \in \mathcal{S}$ и дизъюнктов $\varphi_1 \to D_1, \ldots, \varphi_m \to D_m$, каждый из которых является логическим следствием $\mathcal{S}$. По утверждению 2 и транзитивности отношения логического следствия получаем, что $\varphi \to head(C)$ является логическим следствием $\mathcal{S}$. $\square$

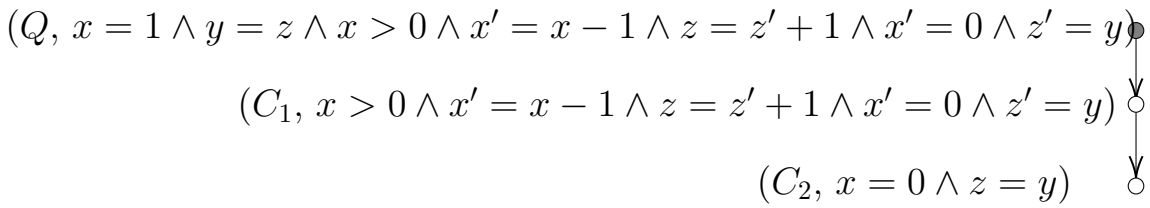


Рис. 1. Гиперрезолютивное опровержение системы из примера 3

Определение 11. *Резолютивное опровержение* системы $\mathcal{S}$ является деревом вывода с корнем (Q, ψ), где $Q \in \mathit{queries}$ и ψ выполнимо в теории $\mathcal{T}$.

Утверждение 3. Система $\mathcal{S}$ невыполнима тогда и только тогда, когда у неё существует резолютивное опровержение.

Доказательство. Данное утверждение следует из корректности и полноты исчисления гиперрезолюций [60]. Необходимость, т.е. полнота исчисления гиперрезолюций, будет доказана в секции 2.4 в качестве применения семантики неподвижной точки.

Чтобы убедиться в достаточности, т.е. корректности исчисления гиперрезолюций, возьмём резолютивное опровержение с корнем (Q,ψ) и предположим, что $\mathcal{S}$ выполнима, т.е. для любой модели $\mathcal{M}$ теории $\mathcal{T}$ существует её обогащение $\left(\mathcal{M}, \overline{X}\right)$, выполняющее все дизъюнкты в $\mathcal{S}$.

По лемме 2 имеем, что $\forall(\psi \to \mathit{head}(Q)) \equiv (\forall\psi \to \bot) \equiv \forall\neg\psi$ является логическим следствием множества дизъюнктов $\mathcal{S}$. Следовательно, любое обогащение $\left(\mathcal{M}, \overline{X}\right)$, выполняющее $\mathcal{S}$, выполняет и $\forall\neg\psi$. Однако это противоречит выполнимости ψ в некоторой модели теории $\mathcal{T}$. Следовательно, $\mathcal{S}$ невыполнима. □

Существует также подход к определению резолютивных опровержений на основе выводов системы сверху-вниз. Для этого используется понятие SLD-резолюции [32]. Однако в данной работе удобно использовать именно гиперрезолютивные деревья вывода.

Пример 5. На рис. 1 представлено дерево опровержения для примера 3. По предложению 3, система невыполнима.

Резолютивные опровержения важны с точки зрения автоматического доказательства невыполнимости систем дизъюнктов Хорна.

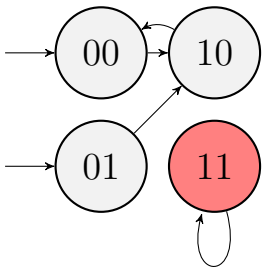


Рис. 2. Пример структуры Крипке с четырьмя состояниями

Утверждение 4. Если множество невыполнимых в теории $\mathcal{T}$ предложений рекурсивно перечислимо, то рекурсивно перечислимо и множество невыполнимых систем дизъюнктов Хорна.

Доказательство. Множество деревьев вывода каждого множества дизъюнктов рекурсивно перечислимо. У корня каждого дерева (Q,ψ) с $Q \in \mathit{queries}$ запустим процедуру определения выполнимости в $\mathcal{T}$; если она остановилась, то по утверждению 3, система невыполнима. □

1.3 От верификации к дизъюнктам Хорна

Дизъюнкты Хорна с ограничениями важны в контексте формальной верификации свойств безопасности. В этом разделе будут коротко описаны способы сведения проблемы доказательства свойств безопасности в различных формализмах к проблеме безопасности систем дизъюнктов Хорна с ограничениями.

1.3.1 Верификация аппаратного обеспечения и протоколов

Как правило, при верификации свойств безопасности аппаратного обеспечения [61] и протоколов [62] модель входной системы (схемы или протокола) представляется в виде *структур Крипке*, а спецификация — в виде формулы *темпоральной логики* (например, CTL) [63]. Существуют теоретико-автоматные методы, позволяющие свести проверку свойств безопасности к проверке свойств вида **AG** φ (φ — формула логики высказываний)[4], т.е. к проверке *инвариантности* пропозиционального свойства φ [64].

[4]Формальные определения синтаксиса и семантики CTL здесь не излагаются. Неформально, **AG** φ означает, что φ выполняется в любом состоянии структуры Крипке.

Один из подходов к верификации системы относительно свойства безопасности состоит в *символьной проверке модели* [65]. Идеей данного подхода служит представление множеств состояний и отношения перехода структуры Крипке формулами логики высказываний от логарифмического числа прозициональных переменных. К примеру, система на рис. 2, имеющая четыре состояния, может быть описана следующими формулами:

$$Init \equiv \neg p_1$$
$$T \equiv (p_1' \leftrightarrow \neg p_1 \vee p_2) \wedge (p_2' \leftrightarrow p_1 \wedge p_2)$$

Каждое состояние описывает интерпретацию переменных p_1 и p_2. Формула $Init$ представляет множество начальных состояний: состояние является начальным, если соответствующая ему интерпретация выполняет $Init$. Аналогично, T представляет отношение перехода: из состояния $\langle s_1, s_2 \rangle$ есть переход в состояние $\langle s_1', s_2' \rangle$ тогда и только тогда, когда выполняется $T(s_1, s_2, s_1', s_2')$. Примером свойства безопасности для системы на рис. 2 является $\mathbf{AG}\ Safe$, где $Safe \equiv \neg p_1 \vee \neg p_2$. Другими словами, в данном примере требуется проверить, что в каждом состоянии, достижимом из начальных, ложен хотя бы один бит.

Итак, задача символьной верификации свойства безопасности модели задаётся тройкой пропозициональных формул $\langle Init(\overline{p}), T(\overline{p}, \overline{p}'), Safe(\overline{p}) \rangle$. В 1990х годах были особо популярны подходы, основывающиеся на *бинарных решающих диаграммах* [65], однако позже популярность приобрели подходы, использующие SAT-решатели на основе алгоритма CDCL [15, 66, 67].

Если свойство безопасности $Safe$ нарушается, то будет существовать *контрпример*, приводящий за k шагов из начального состояния в состояние, выполняющее $\neg Safe$. Другими словами, будет существовать последовательность состояний $\overline{s}, \overline{s}', \ldots, \overline{s}^{(k)}$ таких, что выполняется следующая формула:

$$Init(\overline{s}) \wedge T(\overline{s}, \overline{s}') \wedge \ldots \wedge T(\overline{s}^{(k-1)}, \overline{s}^{(k)}) \wedge \neg Safe(\overline{s}^{(k)}). \tag{1.1}$$

В противном случае будет существовать *безопасный индуктивный инвариант* системы, т.е. пропозициональная формула $Inv(\overline{p})$ такая, что следующие формулы будут общезначимыми:

$$Init(\overline{p}) \rightarrow Inv(\overline{p}) \tag{1.2}$$
$$Inv(\overline{p}) \wedge T(\overline{p}, \overline{p}') \rightarrow Inv(\overline{p}') \tag{1.3}$$
$$Inv(\overline{p}) \rightarrow Safe(\overline{p}) \tag{1.4}$$

Например, вместо Inv подходит формула R, представляющая множество всех состояний, достижимых из начального (такая формула существует по теореме о представимости булевых функций и конечности состояний заданной структуры Крипке). На Inv можно смотреть как на *сертификат корректности* системы относительно свойства $\mathbf{AG}\ Safe$: из общезначимости (1.2)–(1.4) следует, что с помощью Inv тривиальной *индукцией по длине пути* можно доказать недостижимость состояний, выполняющих $\neg Safe$. Таким образом, задача формальной верификации свелась к задаче поиска формулы Inv.

Заметим, что формулы (1.2)–(1.4) напоминают дизъюнкты Хорна с ограничениями. Более формально, рассмотрим сигнатуру языка ограничений с пустым множеством функциональных символов и единственным унарным предикатным символом $true$. Теория $\mathcal{T}$ ограничений будет состоять из единственной интерпретации с носителем {0,1} и интерпретирующей $true$ отношением {1}.

Тогда формулы $Init(\overline{p})$, $T(\overline{p},\overline{p}')$ и $Safe(\overline{p})$ соответствуют *ограничениям* $\varphi_{Init}(\overline{x})$, $\varphi_T(\overline{x},\overline{x}')$ и $\varphi_{Safe}(\overline{x})$: к примеру, формулу языка высказываний $p_1' \leftrightarrow \neg p_1 \vee p_2$ эквивыполнима с ограничением $true(x_1') \leftrightarrow \neg true(x_1) \vee true(x_2)$ в теории $\mathcal{T}$. Введём также неинтерпретированный символ Inv арности $|\overline{p}|$, т.е. положим $\mathcal{R} = \{Inv\}$. В этом случае система дизъюнктов Хорна с ограничениями, представленными ниже, выполнима тогда и только тогда, когда безопасна исходная система.

$$\begin{aligned} \varphi_{Init} &\rightarrow Inv(\overline{x}) \\ \varphi_T \wedge Inv(\overline{x}) &\rightarrow Inv(\overline{x}') \\ \neg\varphi_{Safe} \wedge Inv(\overline{x}) &\rightarrow \bot \end{aligned}$$

При этом сертификаты выполнимости представляют собой искомую формулу Inv. Заметим также, что формулы, которе находятся в корнях всех гиперрезолютивных опровержений данной системы, имеют вид (1.1), т.е. выполняющие оценки свободных переменных листовых ограничений деревьев вывода представляют собой контрпримеры к безопасности.

1.3.2 Верификация программного обеспечения

Существует многочисленные способы сведения задачи верификации свойств безопасности программ к решению систем дизъюнктов Хорна. Один из наиболее очевидных способов состоит в формализации операционной семан-

тики языка программирования в виде *интерпретатора логической программы с ограничениями* с последующей *специализацией* интерпретатора. Например, такой подход реализован в системе VeriMAP [68, 69]. Далее будет показано, как избежать построения такого интерпретатора и порождать систему дизъюнктов по программам на демонстрационных императивных и функциональных языках напрямую.

От верификации императивных программ к дизъюнктам. Рассмотрим задачу сведения доказательства свойств безопасности императивных программ к выполнимости систем дизъюнктов Хорна с ограничениями на примере простого императивного языка программирования `while` [70]. Синтаксис этого языка определяется следующей грамматикой:

$$
\begin{aligned}
P \ ::= \ & \textbf{skip} \\
\mid \ & x \ := \ t \\
\mid \ & P ; P \\
\mid \ & \textbf{if } \varphi \textbf{ then } P \textbf{ else } P \textbf{ end} \\
\mid \ & \textbf{while } \varphi \textbf{ do } P \textbf{ end}
\end{aligned}
$$

Здесь t — это терм языка $\mathcal{A}$, φ — ограничение, x — предметная переменная. Семантика этого языка определяется теорией $\mathcal{T}$ в языке ограничений. Условия цикла `while` являются формулами некоторого языка ограничений $\mathcal{A}$. Семантика языка задаётся стандартно и подробно описана в [70].

Свойства безопасности такой программы можно задавать *тройками Хоара*: $\{\varphi\}\ P\ \{\psi\}$ задаёт *свойство частичной корректности*, где P — программа на языке `while`, φ и ψ — ограничения в языке $\mathcal{A}$, называемые *предусловием* и *постусловием* соответственно. Говорят, что программа P *корректна* относительно φ и ψ, если любое завершающееся вычисление P с начальным состоянием, выполняющим предусловие φ, завершается в состоянии, выполняющем постусловие ψ.

Для данного свойства можно предъявить систему дизъюнктов Хорна $\mathcal{S}$ такую, что программа P *корректна* относительно φ и ψ тогда и только тогда, когда $\mathcal{S}$ выполнима. Описанный метод основан на *исчислении слабейших свободных предусловий* (weakest liberal precondition, WLP) [71, 72].

Опишем процедуру генерации дизъюнктов Хорна. Пусть $\overline{x}$ — множество переменных, используемых в программе, а $\mathcal{R}$ — множество неинтерпретированных символов местности $|\overline{x}|$, число которых больше или равно числу циклов в программе.

Система дизъюнктов порождается функцией $ToHorn$, принимающей на вход тройку Хоара и возвращающей эквивыполнимую систему дизъюнктов:

$$ToHorn(\{\varphi\}\ P\ \{\psi\}) \stackrel{\text{def}}{=} \mathcal{S} \cup \{\varphi \to Q\},\ \text{где}\ \langle Q, \mathcal{S}\rangle \equiv \text{wlp}\,(P,\psi,\varnothing)\,,$$

$$\begin{aligned}
&\text{wlp}\,(\textbf{skip},Q,\mathcal{S}) \stackrel{\text{def}}{=} \langle Q,\mathcal{S}\rangle\\
&\text{wlp}\,(x := t,Q,\mathcal{S}) \stackrel{\text{def}}{=} \langle Q[t/x],\mathcal{S}\rangle\\
&\text{wlp}\,(P_1\,;P_2,Q,\mathcal{S}) \stackrel{\text{def}}{=} \text{wlp}\,(P_1,Q',\mathcal{S}')\,,\ \text{где}\ \langle Q',\mathcal{S}'\rangle \equiv \text{wlp}\,(P_2,Q,\mathcal{S})\\
&\text{wlp}\,(\textbf{if}\ \varphi\ \textbf{then}\ P_1\ \textbf{else}\ P_2\ \textbf{end},Q,\mathcal{S}) \stackrel{\text{def}}{=} \langle(\varphi \wedge Q_1) \vee (\neg\varphi \wedge Q_2), \mathcal{S}_1 \cup \mathcal{S}_2\rangle\,,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{где}\ \langle Q_i, \mathcal{S}_i\rangle \equiv \text{wlp}\,(P_i,Q,\mathcal{S})\\
&\text{wlp}\,(\textbf{while}\ \varphi\ \textbf{do}\ P\ \textbf{end},Q,\mathcal{S}) \stackrel{\text{def}}{=} \langle Inv(\overline{x}), \mathcal{S}' \cup \{C_1,C_2\}\rangle\,,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\text{где}\ Inv\ \text{— новый символ из}\ \mathcal{R},\\
&\qquad\qquad\qquad\qquad\qquad\qquad\langle Q',\mathcal{S}'\rangle \equiv \text{wlp}\,(P,Inv(\overline{x}),\mathcal{S})\\
&\qquad\qquad\qquad\qquad\qquad\qquad C_1 \equiv \varphi \wedge Inv(\overline{x}) \to Q'\\
&\qquad\qquad\qquad\qquad\qquad\qquad C_2 \equiv \neg\varphi \wedge Inv(\overline{x}) \to Q
\end{aligned}$$

Следует отметить, что дизъюнкты, порождаемые определением $ToHorn$, синтаксически отличаются от определения 1. Во-первых, допускается запись дизъюнктов-запросов в виде $B \to \varphi$, где φ — ограничение. Но это не является существенным отклонением от определения 1, поскольку каждый такой дизъюнкт можно переписать в виде $B \wedge \neg\varphi \to \bot$. Во-вторых, допускаются неинтерпретированные атомы вида $Inv(\overline{t})$, где $\overline{t}$ — кортеж произвольных термов (а не предметных переменных, как в определении 1). Однако можно заменить $\overline{t}$ на кортеж новых предметных переменных $\overline{x}$, добавив в ограничение дизъюнкта равенства вида $x_i = t_i$. Например, дизъюнкт

$$i > 0 \wedge Inv(i) \to Inv(i-1)$$

можно превратить в эквивыполнимый дизъюнкт

$$i > 0 \wedge i' = i - 1 \wedge Inv(i) \to Inv(i').$$

Пример 6. Пусть $\mathcal{A}$ — язык линейной целочисленной арифметики. Напомним, в этом языке отсутствует функциональный символ произведения. Рассмотрим следующую программу P:

$$
\begin{array}{l}
i := 0; \\
\textbf{while } i < x \textbf{ do} \\
\qquad i := i + 1;\ \ z_1 := z_1 + y; \\
\textbf{end} \\
i := 0; \\
\textbf{while } i < x \textbf{ do} \\
\qquad i := i + 1;\ \ z_2 := z_2 + y; \\
\textbf{end}
\end{array}
$$

P итеративно добавляет произведение x и y к переменным z_1 и z_2. Очевидно, P корректна относительно предусловия $z_1 = 0 \wedge z_2 = 0$ и постусловия $z_1 = z_2$.

По определению $ToHorn$ имеем следующее:

$$
\begin{array}{l}
ToHorn(\{z_1{=}0 \wedge z_2{=}0\}\ P\ \{z_1{=}z_2\}) \equiv \{ \\
\qquad\qquad z_1 = 0 \wedge z_2 = 0 \to Inv_1(x, y, 0, z_1, z_2) \\
\qquad i < x \wedge Inv_1(x, y, i, z_1, z_2) \to Inv_1(x, y, i + 1, z_1, z_2 + y) \\
\\
\quad \neg(i < x) \wedge Inv_1(x, y, i, z_1, z_2) \to Inv_2(x, y, 0, z_1, z_2) \\
\qquad i < x \wedge Inv_2(x, y, i, z_1, z_2) \to Inv_2(x, y, i + 1, z_1, z_2 + y) \\
\\
\quad \neg(i < x) \wedge Inv_2(x, y, i, z_1, z_2) \to z_1 = z_2 \\
\}
\end{array}
$$

Эта система выполнима: для того, чтобы убедиться в этом, достаточно проинтерпретировать символ Inv_1 следующим отношением:

$$\left\{\langle x,y,i,z_1,z_2\rangle \in \mathbb{Z}^5 \mid 0 \leqslant i \leqslant x, z_1 = i \cdot y, z_2 = 0\right\},$$

а символ Inv_2 — вот таким отношением:

$$\left\{\langle x,y,i,z_1,z_2\rangle \in \mathbb{Z}^5 \mid 0 \leqslant i \leqslant x, z_1 = x \cdot y, z_2 = i \cdot y\right\}.$$

Более эффективные подходы, применяемые для решения задачи сведения доказательства свойств безопасности императивных программ к выполнимости систем дизъюнктов Хорна с ограничениями в контексте более сложных императивных языков описаны в [73–78].

От верификации функциональных программ к дизъюнктам. По сравнению с императивными программами, верификация программ на функциональных языках упрощается неизменяемостью состояний, но осложняются наличием функций высшего порядка. Тем не менее, верификация функциональных программ с функциями высших порядков также сводится к проверке выполнимости дизъюнктов Хорна с ограничениями.

Самые простые, но *некомпозициональные*[5] способы сведения основаны на идее *дефункционализации* [79–81]. Дефункционализация позволяет породить по функциональной программе систему дизъюнктов Хорна над теорией некоторого алгебраического типа данных. Более общий подход, основанный на замыканиях, представлен в дедуктивной системе [82]. Некоторые подходы используют булевы абстракции функциональных программ с функциями высшего порядка [83].

Одним из самых эффективных инструментов построения корректных функциональных программ высших порядков являются *зависимые типы* [84]. Одним из видов зависимых типов являются *уточняющие типы* (refinement types). Уточняющие типы реализованы, например, для таких языков, как Haskell [85], F^* [86], TypeScript [87], Ruby [88] и других языков.

Уточняющие типы позволяют вместе с базовым типом данных указывать *предикат*, дополнительно «отфильтровывающий» некоторые значения этого базового типа. Предикаты формулируются в виде формул некоторого языка ограничений $\mathcal{A}$. Например, вместо обычной аннотации типа функции целочисленного деления

$$\texttt{div::int} \to \texttt{int} \to \texttt{int},$$

[5]Говорят, что подход к верификации программы композионален, если он анализирует каждую подпрограмму «в изоляции» от других, а затем «совмещает» результаты этих анализов. В таком случае результаты верификации всей программы — композиция результатов маленьких её частей. Другими словами, композициональная верификация программ следует идее «разделяй и властвуй»: задача верификации большой системы «распадается» на множество независимых подзадач, результаты решения которых затем совмещаются некоторым образом.

допускающей значение 0 в качестве второго аргумента, что приводит к ошибке времени исполнения, уточняющие типы позволяют написать аннотацию, исключающую такую возможность:

$$\texttt{div::int} \to \{v : \texttt{int} \mid \neg(v = 0)\} \to \texttt{int}.$$

Уточняющие типы позволяют свести статическую проверку свойств безопасности программы к проверке выполнимости формул языка ограничений; на практике для этой проверки выполнимости используются SMT-решатели. Таким образом, для верификации свойства безопасности функциональной программы достаточно *вывести* типовую аннотацию, доказывающую данное свойство.

Для разрешимых теорий языка ограничений проверка корректности типовой аннотации разрешима. Тем не менее, автоматический вывод уточняющих типов, доказывающего заданное свойство безопасности, — неразрешимая проблема даже для разрешимых теорий языка ограничений. Она требует автоматического поиска необходимых предикатов, уточняющих базовые типы. Такие предикаты могут быть получены как символьные модели системы дизъюнктов Хорна с ограничениями, порождённой по исходной программе [89–91].

Кратко опишем суть сведения задачи вывода уточняющих типов к поиску символьных моделей систем дизъюнктов. Рассмотрим, например, систему Liquid Types [90]. В этой системе уточняющий тип имеет вид $\{v : T \mid \varphi(v,\overline{x})\}$, где T — базовый тип (например, `int`), φ — ограничение. *Связывание* переменной и её типа, т.е. утверждение вида $y : \{v : T \mid \varphi(v,\overline{x})\}$, соответствует логическому ограничению $\varphi(y,\overline{x})$.

Правила проверки типов в системе Liquid Types порождают множество *суждений* вида $\Gamma \vdash \{v : T \mid \varphi_1(v,\overline{x})\} \prec \{v : T \mid \varphi_2(v,\overline{x})\}$. Здесь $\prec$ — *отношение подтипирования*, а Γ — *контекст*, являющийся множеством связываний переменных с типами. Задача *вывода* уточняющих типов состоит в нахождении ограничений φ таких, что все суждения будут истинными.

Для этого можно считать, что все ограничения в уточняющих типах заменены на неинтерпретированные символы P_i. В таком случае, суждению

$$\Gamma \vdash \{v : T \mid P_1(v,\overline{x})\} \prec \{v : T \mid P_2(v,\overline{x})\}$$

можно сопоставить дизъюнкт

$$\bigwedge_{y:\{v:T \mid P(v,\overline{z})\} \in \Gamma} P(y,\overline{z}) \wedge P_1(v,\overline{x}) \to P_2(v,\overline{x}).$$

Таким образом, системы проверки уточняющих типов сопоставляют функциональной программе множество суждений, каждому из которых соответствует дизъюнкт Хорна с ограничением, т.е. каждой функциональной программе и свойству её безопасности можно сопоставить систему дизъюнктов Хорна. Существование сертификата выполнимости такой системы будет достаточным условием для корректности программы, а заодно по сертификату выполнимости можно построить типовую аннотацию программы.

1.3.3 Реляционная верификация

Многие интересные свойства программ выходят за рамки «стандартных» свойств безопасности — утверждений о поведении некоторой программы в случае её завершения. Существует целый класс свойств, называемых *реляционными*, которые должны выполняться *совместно несколькими программами*, либо *несколькими запусками одной и той же программы*. Доказательством реляционных свойств набора программ занимается область, называемая *реляционной верификацией*.

Одним из самых простых примеров реляционного свойства является *функциональная эквивалентность программ* [28]: две программы, запущенные и завершившиеся на одних и тех же входных данных, выдадут одни и те же выходные данные. Автоматическая проверка функциональной эквивалентности программ используется, к примеру, в оптимизирующей компиляции [92–94], при обфускации (запутывании текстов программ) [95], а также при автоматическом рефакторинге программного кода [96] и т.д.

Другой пример реляционного свойства — это *невмешательство* [97]: два завершившихся исполнения программы должны выдавать одни и те же выходные данные для одних и тех же ненадёжных входных данных, вне зависимости от надёжных данных, которым оперирует программа. Невмешательство — это важное свойство для программного обеспечения, работающего с конфиденциальными данными, т.к. нарушение этого свойства делает систему уязвимой к *атакам по сторонним каналам*, которые использовались для доступа к конфиденциальным данным, включая пользовательские аккаунты [98–100], криптографические ключи [101–103] и медицинские данные [104].

Другие варианты применения реляционной верификации включают в себя автоматическое регрессионное тестирование [105, 106], разрешение конфликтов

слияния в системах контроля версий [107], автоматический анализ сложности алгоритмов [108], построение срезов программ [109].

Особым видом реляционных свойств безопасности являются так называемые свойства *гипербезопасности* (другое название — *k-свойства*) [25]. В отличие от «стандартных» свойств безопасности, k-свойства представляют собой утверждение о k завершающихся запусках одной и той же программы. Примером k-свойства может служить детерминированность программы, вышеописанные свойства невмешательство вместе с подобными им свойствами так называемого *безопасного потока информации* [110].

Одним из самых распространённых на сегодняшний день подходов к верификации реляционных свойств безопасности состоит в сведении к проверке «обычных» свойств безопасности некоторой программы. Для этого строится *произведение* верифицируемых программ, т.е. программа, моделирующая одновременное исполнение перемножаемых программ, а реляционное свойство безопасности превращается в свойство безопасности программы-произведения [26, 106, 111, 112].

Существуют различные адаптации логики Хоара для доказательства реляционных свойств безопасности императивных программ. Например, таковой является система *декартова логика Хоара* [113–115] и реляционная логика разделения [116].

Проверка реляционных свойств набора программ может быть сведена к проверке выполнимости системы дизъюнктов Хорна с ограничениями [23]. Основная идея такого подхода заключается в следующем. Пусть даны k программ с входами $\bar{i}_1, \ldots, \bar{i}_k$ и выходами $\bar{o}_1, \ldots, \bar{o}_k$, а также реляционное свойство, выраженное в виде формулы φ некоторого языка первого порядка. Например, свойство функциональной эквивалентности этих программ будет выглядеть следующим образом:

$$\bar{i}_1 = \ldots = \bar{i}_k \to \bar{o}_1 = \ldots = \bar{o}_k.$$

Для каждой программы заведём неинтерпретированный символ P_j арности $\left|\bar{i}_j\right| + |\bar{o}_j|$, где j принимает значения от 1 до k. Семантику поведения каждой программы представим в виде множества правил символа P_j, используя методы, описанные в предыдущих разделах. Пусть $\mathcal{S}_j$ — множества таких дизъюнктов для программы j. Итоговая система дизъюнктов, представленная ниже, выполнима тогда и только тогда, когда выполняется реляционное свойство φ.

$$\mathcal{S} \stackrel{\text{def}}{=} \bigcup_{j=1}^{k} \mathcal{S}_j \cup \{Q\}, \texttt{где}\ \ Q \equiv \neg\varphi \wedge P_1(\bar{i}_1,\bar{o}_1) \wedge \cdots \wedge P_k(\bar{i}_k,\bar{o}_k) \to \bot$$

1.4 Автоматическое решение систем дизъюнктов

Итак, проверка разного класса свойств программ на различных языках программирования может быть сведена к проверке выполнимости системы дизъюнктов Хорна с ограничениями.

Существуют инструменты решения систем дизъюнктов Хорна с ограничениями. Такие инструменты, называемые *Хорн-решателями* (Horn solvers), принимают на вход систему дизъюнктов Хорна с ограничениями в некоторой теории, поддерживаемой решателем, и выводят `SAT` вместе с сертификатом выполнимости, если система выполнима, либо `UNSAT`, если невыполнима. Так как для многих теорий задача определения выполнимости системы дизъюнктов Хорна с огранчениями алгоритмически неразрешима, такие решатели могут не завершиться; некоторые решатели могут выдавать ответ `UNKNOWN` в некоторых случаях, когда входная система гарантированно не может быть им решена.

Наиболее известными и эффективными Хорн-решателями на сегодняшний день являются Spacer [6], Eldarica [7], HoIce [9], Ultimate TreeAutomizer [117], Sally [118], Qarmc [8], TransfHORNer [24], FreqHorn [10].

1.4.1 Подходы к решению систем дизъюнктов Хорна.

Существует большое количество подходов к решению дизъюнктов Хорна. Здесь будут перечислены лишь те, которые являются наиболее эффективными на сегодняшний день.

Использование синтаксических трансформаций . Существует большое количество подходов, построенных на последовательных переписываниях дизъюнктов Хорна [21, 23, 24, 119]. Примером решателя, основанного на синтаксических трансформациях, является TransfHORNer. Обзор таких методов представлен в [120].

Синтаксически-направляемый синтез инвариантов. Несколько лет назад появилось соревнование SyGuS (Syntax-Guided Synthesis) [121]. Соревнующиеся инструменты должны синтезировать формулу, имея на входе два вида ограничений: семантические, заданные формулой некоторого языка ограничений, и синтаксические, заданные контекстно-свободной грамматикой. Очевидно, задача выполнимости систем дизъюнктов Хорна с ограничениями сводится к задаче, решаемой на соревнованиях SyGuS, однако решатели SyGuS общего назначения оказались не достаточно эффективными для синтеза символьных моделей дизъюнктов Хорна. В результате была разработана более удачная адаптация подходов SyGuS для синтеза символьных моделей систем дизъюнктов, реализованная в Хорн-решателе FreqHorn [10].

Уточнение абстракции по контрпримерами. Уточнение абстракции по контрпримерам [122, 123] (counterexample-guided abstraction refinement, CEGAR) является одним из наиболее успешных подходов в современной верификации программного обеспечения [14]. Общая идея заключается в следующем: начав с наиболее грубой абстракции модели

$$\mathcal{I} \stackrel{\text{def}}{=} P_i \mapsto \top$$

, т.е тривиальной символьной интерпретации, сопоставляющей каждому символу тождественную истину, проверяется выполнимость в $\mathcal{T}$ ограничения

$$\bigvee C \in \mathcal{S} \neg [\![C]\!]_{\mathcal{I}} .$$

Если ограничение невыполнимо, то $\mathcal{I}$ является сертификатом безопасности. В противном случае SMT-решатель выдаёт выполняющую модель и оценку свободных переменных, которые используются для *уточнения* (refinement) $\mathcal{I}$, т.е. для усиления формул образа. Формулы усиливаются таким образом, чтобы исключить появление той же модели при повторном запросе. Этот процесс повторяется итеративно до тех пор, пока либо $\mathcal{I}$ не станет сертификатом выполнимости, либо очередной контрпример не будет выполнять корень какого-либо гиперрезолютивного вывода системы.

Подход CEGAR реализован, например, в решателях Eldarica и Qarmc.

Использование обучения с учителем. В 2014 году был предложен метод ICE, использующий обучение с учителем для построения инвариантов цик-

Solver	Score	#SAT	#UNSAT	Avg time
Spacer	279	194	85	28.90
Rebus	267	188	79	41.85
Eldarica	**209**	129	80	24.55
Ultimate Unihorn Automizer	**133**	63	70	23.05
Hoice	**129**	65	64	7.09
Ultimate Tree Automizer	**107**	42	65	29.15
PCSat	**45**	33	12	23.74

Табл. 1. CHC-Comp 2019: LIA-Lin

лов [22]. Позднее, в 2018 году, подход был обобщён на дизъюнкты Хорна с ограничениями [9]. Этот подход был реализован в решателе Hoice.

Достижимость, направляемая свойством. Достижимость, направляемая свойством (property-directed reachability, PDR) — подход, изначально предложенный Аароном Брэдли и назанный им IC3, для верификации аппаратного обеспечения [15] с помощью SAT-решателей. Далее подход был обобщён на случай языков первого порядка [124] и адаптирован для решения дизъюнктов Хорна с ограничениями [6, 125].

PDR является адаптацией похода CEGAR. Главное его отличие состоит в последовательном построении серии *индуктивных усилений* отрицания ограничений дизъюнктов-запросов. Подход удачно совмещает процедуры поиска гиперрезолютивного опровержения с шагами к построению сертификата выполнимости, используя информацию, полученную от одной процедуры в другой и наоборот. PDR будет подробно описан в главе 4.

PDR реализован в решателях Spacer и Sally.

1.4.2 Сравнение решателей дизъюнктов Хорна

Существует ежегодное соревнование Хорн-решателей CHC-Comp. В табл. 1, 2, 3 и 4 представлены результаты последнего соревнования CHC-Comp 2019 [126]. Таблицы отражают результаты различных секций этого со-

Solver	Score	#SAT	#UNSAT	Avg time
Spacer	270	153	117	5.04
Eldarica	**234**	131	103	15.93
Ultimate Unihorn Automizer	**177**	96	81	36.94
Hoice	**176**	110	66	9.85
PCSat	**123**	81	42	24.69
Ultimate Tree Automizer	**73**	29	44	4.85

Табл. 2. CHC-Comp 2019: LIA-Nonlin

Solver	Score	#SAT	#UNSAT	Avg time
Spacer	159	76	83	9.60
Ultimate Unihorn Automizer	**90**	44	46	28.47
Ultimate Tree Automizer	**71**	39	32	44.14
Hoice	**35**	24	11	0.06
Eldarica	**20**	20	0	100.14

Табл. 3. CHC-Comp 2019: LIA+Array-Lin

ревнования: различные секции содержат тесты с системами дизъюнктов Хорна над различными языками ограничений различного вида[6]. Системы в секции LIA-Lin (рис. 1) являются линейными над линейной целочисленной арифметикой, в секции LIA-Nonlin (рис. 2) — нелинейными над линейной целочисленной арифметикой, LIA+Array-Lin (рис. 3) — линейными над комбинацией эктенсиональной теории массивов с линейной целочисленной арифметикой, LRA-TS (рис. 4) — линейные системы с одним неинтерпретированным символом над линейной вещественной арифметикой. Первая колонка каждой таблицы содержит название решателя, вторая — общее количество очков, набранное

[6]Таблицы взяты из `https://chc-comp.github.io/2019/chc-comp19.pdf` (дата обращения: 1.05.2020).

Solver	Score	#SAT	#UNSAT	Avg time
Sally-y2o2-decomposed-itp	**194**	150	44	43.71
Sally-y2o2-Farkas-itp	**194**	150	44	44.34
Rebus	**190**	137	53	53.24
Sally-y2o2-dual-Farkas-itp	**188**	144	44	53.46
Sally-y2m5	**179**	135	44	40.07
Spacer	**173**	126	47	46.19
Sally-y2o2-dual-decomposed-itp	**157**	118	39	67.67
Ultimate Tree Automizer	**93**	73	20	55.15
Ultimate Unihorn Automizer	**67**	50	17	22.21

Табл. 4. CHC-Comp 2019: LRA-TS

решателем, третья — количество решённых выполнимых систем, четвёртая — количество решённых невыполнимых систем, пятая — среднее время работы в секундах.

Как видно из таблиц, наиболее эффективными на сегодняшний день решателей являются Spacer и Sally. Оба этих решателя используют подход направляемой свойством достижимости. Поэтому можно считать, что на сегодняшний день именно направляемая свойством достижимость является наиболее перспективным подходом к решению систем дизъюнктов Хорна с ограничениями.

Итак, по утверждению 3, любая невыполнимая система имеет резолютивное опровержение, по факту являющееся синтаксическим сертификатом невыполнимости системы. К сожалению, в случае с сертификатами выполнимости это не так: *не любая выполнимая система имеет сертификат выполнимости.*

Пример 7. Рассмотрим следующую систему дизъюнктов Хорна с ограничениями в линейной целочисленной арифметике:

$$x = 0 \wedge z = 0 \to mul(x,y,z)$$
$$x > 0 \wedge x' = x - 1 \wedge z = z' + y \wedge mul(x',y,z') \to mul(x,y,z)$$
$$x = x' \wedge y = y' \wedge z \neq z' \wedge mul(x,y,z) \wedge mul(x',y',z') \to \bot$$

Она выполнима: к примеру, символ mul можно проинтерпретировать отношением

$$\left\{\langle x,y,z\rangle \in \mathbb{Z}^3 \mid z = x \cdot y\right\}.$$

Тем не менее, у этой системы не существует ни одного сертификата выполнимости. Это строго показано в разделе 2.6, но основная причина этого состоит в том, что умножение не представимо в линейной целочисленной арифметике.

Пример 8. Аналогично, система в примере 1 выполнима, но у неё не существует сертификата выполнимости. Неформальное объяснение следующее: эта система представляет условия верификации программы, которая (1) вычисляет сумму элементов в узлах двоичного дерева T, (2) вычисляет сумму элементов двоичного дерева, полученного из T увеличением каждого элемента на 2 и (3) проверяет, что вторая сумма равна первой плюс удвоенное количество элементов в дереве T. Язык ограничений недостаточно выразителен для того, чтобы описать эти ограничения.

Заметим, что системы в примерах 7 и 8 нелинейны. Неформально говоря, непредставимость сертификатов выполнимости можно объяснить следующим образом. Ограничения нелинейных дизъюнктов формулируют утверждение о переменных сразу нескольких неинтерпретированных атомов. Поэтому для того, чтобы быть сертификатом выполнимости, символьная интерпретация должна выполнять правила сразу нескольких символов, т.е. «резюмировать» формулами языка ограничений правила нескольких символов, и при этом эти «резюме» должны быть достаточно сильными для доказательства утверждения, формулируемого ограничением нелинейного дизъюнкта. Существуют и примеры выполнимых линейных систем без сертификатов выполнимости [127], но на практике такие системы встречаются реже [126].

Проблема непредставимости сертификатов выполнимости в языке ограничений носит фундаментальный характер. Любая теория, выполнимость ограничений в которой разрешима, но выполнимость дизъюнктов Хорна над которой неразрешима[7], будет невыразительной в том смысле, что будут существовать выполнимые системы дизъюнктов Хорна без сертификата выполнимости.

Действительно, как будет показано в разделе 2.4.4, любая невыполнимая система имеет гиперрезолютивное опровержение. Так как выполнимость огра-

[7]Таковыми являются все теории, описанные в разделе 1.1.4, кроме теории алгебраического типа $List(T)$ и теории битовых векторов [127, 128]

ничений разрешима, множество гиперрезолютивных опровержений рекурсивно перечислимо. Но если бы сертификаты выполнимости существовали бы для любой такой системы, их множество также было бы перечислимо.

Эта закономерность хорошо видна в примере 7. Выполнимость ограничений в линейной целочисленной арифметике разрешима, но выполнимость систем дизъюнктов Хорна с линейно-арифметическими ограничениями неразрешима [127]. Заметим, что если бы мы хотели увеличить представимость языка ограничений, добавив в него умножение, то выполняющая интерпретация станет представима, но выполнимость ограничений станет неразрешимой. Таким образом, имеет место *компромисс* между разрешимостью языка ограничений и его выразительностью.

Получив выполнимую систему без сертификата выполнимости, любой современный решатель дизъюнктов Хорна с ограничениями не завершится, так как на сегодняшний день отсутствуют методы автоматического обнаружения такой ситуации.

1.5 Выводы

На основе проделанного обзора сделаны следующие выводы.

- Задача автоматического вывода символьных моделей систем дизъюнктов Хорна с ограничениями важна для формальной верификации программ, поскольку к ней сводится большинство проблем доказательства корректности кода на различных языках программирования. При этом важны нелинейные системы дизъюнктов, позволяющие описывать реляционные спецификации корректности программ, необходимые для многих практических задач.
- Достижимость, направляемая свойством, на текущий момент является наиболее эффективным подходом к автоматическому решению систем дизъюнктов Хорна.
- Фундаментальным препятствием на пути к решению систем дизъюнктов является непредставимость моделей в языке решателя. В таких ситуациях любой существующий алгоритм автоматического построения символьных моделей не завершается. При этом обогащение языка может сделать проверку выполнимости его формул неразрешимой, что

сделает невозможной в общем случае даже автоматическую проверку корректности решений-кандидатов.

Таким образом, современным направлением формальной верификации программ является поиск подхода, который преодолевает проблему непредставимости моделей. Адаптация такого подхода в контексте достижимости, направляемой свойством, является важной на практике задачей.

Глава 2. Синхронизация дизъюнктов

В данной главе представлено понятие *синхронизации дизъюнктов Хорна*. Синхронизация позволяет частично решить проблему непредставимости моделей. Корректность понятия синхронизации дизъюнктов установлено двумя способами: синтаксическим (см. раздел 2.2) и семантическим (см. раздел 2.5).

В разделе 2.7 описан алгоритм, переписывающий входную систему дизъюнктов таким образом, что если у входной системы существует сертификат выполнимости, то он существует и у переписанной, но обратное в общем случае не верно. Выполнен формальный анализ этого алгоритма, доказана его корректность.

2.1 Определение синхронизации дизъюнктов Хорна

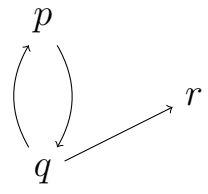
Пусть $\mathcal{S}$ является системой дизъюнктов Хорна с ограничениями. Тогда *графом зависимостей* $\mathcal{S}$ назовём ориентированный граф $\langle \mathcal{R}, E \rangle$, где $\langle P,Q \rangle \in E$ тогда и только тогда, когда символ Q появляется в теле некоторого правила символа P. Символы P и Q называются *рекурсивными*, если P и Q лежат в одной компоненте сильной связности графа зависимостей $\langle \mathcal{R}, E \rangle$.

Пример графа зависимостей для системы дизъюнктов Хорна с ограничениями представлен на рис. 3. В этом примере символ p рекурсивен с q, а символ r достижим из p и q, но не наоборот.

Пусть C — дизъюнкт Хорна с ограничением. Неинтерпретированный атом $R(\overline{x})$ в теле C называется *рекурсивным*, если $C \in rules(P)$, и P и R рекурсивны. В противном случае, атом называется *нерекурсивным*. Другими словами,

$$
\begin{aligned}
p(x) \wedge q(y) &\to \bot \\
\varphi_1 &\to p(x) \\
\varphi_2 \wedge q(y) &\to p(x) \\
\varphi_3 \wedge p(x) \wedge r(x) &\to q(y) \\
\psi &\to r(x)
\end{aligned}
$$

а) Система дизъюнктов $\mathcal{S}$



б) Граф зависимостей для $\mathcal{S}$

Рис. 3. Пример системы дизъюнктов и соответствующего графа зависимостей.

атом $R(\overline{x})$ нерекурсивен, если он входит в тело запроса, или R нерекурсивен с символом в заголовке C.

К примеру, атом $P(\overline{x})$ в дизъюнкте $\varphi \wedge P(\overline{x}) \to P(\overline{x}')$ рекурсивен, а в дизъюнкте $\varphi \wedge P(\overline{x}) \to \bot$ нерекурсивен.

Множество рекурсивных атомов в теле C обозначим $Rec\,(C)$, а множество нерекурсивных атомов — $NRec\,(C)$. Таким образом, тело любого дизъюнкта Хорна C с ограничением φ можно записать следующим образом:

$$\varphi \wedge \bigwedge NRec\,(C) \wedge \bigwedge Rec\,(C)\,.$$

Определение 12. Непустые множества $A_1, \ldots, A_n$ *накрываются* множеством A, что обозначается $A \in \lceil A_1, \ldots, A_n \rceil$, если выполнены следующие условия.

1. $A \subseteq A_1 \times \cdots \times A_n$.
2. Каждый элемент каждого A_i появляется в позиции i, как минимум, одного кортежа $x \in A$.

К примеру, справедливы следующие утверждения:

$$\{(1,3,5),(2,4,5)\} \in \lceil \{1,2\},\{3,4\},\{5\} \rceil$$
$$\{(1,3,5),(2,3,5)\} \notin \lceil \{1,2\},\{3,4\},\{5\} \rceil$$

Множество мультимножеств на X, т.е. множество всех отображений X на множество натуральных чисел, обозначим $\mathbb{N}^X$. Мультимножества частично упорядочены отношением включения, т.е. для $m_1, m_2 \in \mathbb{N}^X$, $m_1 \subseteq m_2$, если $\forall x \in X, m_1(x) \leqslant m_2(x)$.

Будем ассоциировать кортеж $\overline{x} = \langle x_1, \ldots, x_m \rangle$ с мультимножеством $\{x_i \mapsto \#x_i\}$, где $\#x_i$ — это число вхождений x_i в $\overline{x}$. Таким образом, для функции $f : \mathbb{N}^X \to Y$ будем допускать запись $f(\langle x_1, \ldots, x_m \rangle)$, подразумевая $f(\{x_i \mapsto \#x_i\})$.

Введём счётное множество предикатных символов $\mathcal{R}'$ и биективное отображение $G : \mathbb{N}^{\mathcal{R}} \to \mathcal{R}'$ такие, что для всех $\langle R_1, \ldots, R_m \rangle \in \mathbb{N}^{\mathcal{R}}$ выполнено

$$ar\,(G\,(\langle R_1, \ldots, R_m \rangle)) = \sum_{i=1}^{m} ar\,(R_i)\,.$$

Потребуем, чтобы $\mathcal{R} \subseteq \mathcal{R}'$, и для всех $P \in \mathcal{R}$ было выполнено $G(\{P \mapsto 1\}) = P$.

Неформально, символы в $\mathcal{R}'$ будут обозначать «слияния» символов их G-прообраза. Будем записывать имена этих символов в виде «произведений»

имён соответствующих символов. Например, символ $G(\{P \mapsto 2, Q \mapsto 1\})$ будем записывать как P^2Q, при этом $ar\left(P^2Q\right) = 2ar\,(P) + ar\,(Q)$.

Теперь перейдём к определению синхронизации дизъюнктов. Для это дадим два вспомогательных определения — произведение неинтерпретированных атомов и произведение дизъюнктов.

Определение 13. Пусть "$\leqslant$" — линейный порядок на $\mathcal{R}$, и пусть $R_1,\ldots,R_m \in \mathcal{R}$, $R \stackrel{\text{def}}{=} G(\langle R_1,\ldots,R_m\rangle)$. *Произведением неинтерпретированных атомов* $R_1(\overline{x}_1),\ldots,R_m(\overline{x}_m)$ будем называть

$$\prod_{i=1}^{m} R_i(\overline{x}_i) \stackrel{\text{def}}{=} R(\overline{x}_{j_1},\ldots,\overline{x}_{j_m}),$$

где $(j_1,\ldots,j_m)$ — такая перестановка чисел $1,\ldots,m$, что $R_{j_1} \leqslant \ldots \leqslant R_{j_m}$.

Пусть, например, $\mathcal{R} = \{P,Q\}$ и $P \leqslant Q$. Произведением неинтерпретированных атомов $Q(y)$ и $P(x)$ является атом $PQ(x,y)$. Далее, без потери общности будем считать, что в любом произведении $\prod_{i=1}^{m} R_i(\overline{x}_i)$ символы R_i проиндексированы таким образом, что $R_1 \leqslant \ldots \leqslant R_m$, и в таком случае

$$\prod_{i=1}^{m} R_i(\overline{x}_i) \stackrel{\text{def}}{=} R(\overline{x}_1,\ldots,\overline{x}_m).$$

Введём следующее обозначение для дизъюнкта Хорна C:

$$Rec_{/head}\,(C) \stackrel{\text{def}}{=} \begin{cases} Rec\,(C) & \text{если } Rec\,(C) \neq \varnothing, \\ \{head(C)\} & \text{если } Rec\,(C) = \varnothing. \end{cases}$$

Определение 14. Пусть $C_1,\ldots,C_m$ — дизъюнкты Хорна с ограничениями, $R_1,\ldots,R_m$ — неинтерпретированные символы (возможно, повторяющиеся) и такие, что $C_i \in rules(R_i)$ для всех $i \in \{1,\ldots,m\}$. Дизъюнкт C над $\mathcal{R}'$ называется *произведением дизъюнктов* $C_1,\ldots,C_m$ (обозначается $C = C_1 \times \cdots \times C_m$),

если выполнены следующие утверждения:

$$
\begin{aligned}
head(C) &= \prod_{i=1}^{m} head(C_i) \\
body(C) &= \varphi \wedge \bigwedge NRec\,(C) \wedge \bigwedge Rec\,(C) \\
NRec\,(C) &= \bigcup_{i=1}^{m} NRec\,(C_i) \\
A &\in \left\lceil Rec_{/head}\,(C_1), \ldots, Rec_{/head}\,(C_m) \right\rceil \\
Rec\,(C) &= \Big\{ \prod_{i=1}^{m} a_i \;\Big|\; \langle a_1, \ldots a_m \rangle \in A \Big\} \setminus \{head(C)\}.
\end{aligned} \tag{2.1}
$$

При этом ограничение дизъюнкта C будет $\varphi = \bigwedge_{i=1}^{m} \varphi_i$, где φ_i — это ограничение дизъюнкта C_i.

К примеру, если $C_1 \equiv \varphi_1 \to P(\overline{x})$ и $C_2 \equiv \varphi_2 \wedge Q(\overline{y}') \to Q(\overline{y})$, то

$$C_1 \times C_2 \equiv \varphi_1 \wedge \varphi_2 \wedge PQ(\overline{x}, \overline{y}') \to PQ(\overline{x}, \overline{y}).$$

Заметим, что в определении 14 не требуется, чтобы каждый символ в дизъюнктах $C_1, \ldots, C_m$ являлся элементом $\mathcal{R}$. Однако произведения атомов определено только в случае, когда символы в заголовке дизъюнктов $C_1, \ldots, C_m$ и их рекурсивных атомах являются элементами $\mathcal{R}$.

Каждому символу $R \in \mathcal{R}'$ сопоставим множество правил $rules(R)$, заданных следующим образом. Пусть $G^{-1}(R) = \{P_1 \mapsto k_1, \ldots, P_n \mapsto k_n\}$. Тогда для всех $i \in \{1, \ldots, n\}$ и $j \in \{1, \ldots, k_i\}$ определим $rules^{(j)}(P_i)$ как копию $rules(P_i)$ с переименованными переменными таким образом, что переменные в $rules^{(j)}(P_i)$ и $rules^{(j')}(P_{i'})$ различны для $j \neq j'$ или $i \neq i'$; пусть также $k \stackrel{\text{def}}{=} k_1 + \ldots + k_n$. Тогда

$$rules(R) = \left\{ C_1 \times \cdots \times C_k \,\middle|\, \langle C_1, \ldots, C_k \rangle \in rules^{(1)}(P_1) \times \cdots \times rules^{(k_n)}(P_n) \right\}. \tag{2.2}$$

Пример 9. Вернёмся к системе дизъюнктов в примере 7, у которой

$$
\begin{aligned}
rules(mul) = \{ & x{=}0 \wedge z{=}0 \to mul(x,y,z), \\
& x > 0 \wedge x'{=}x - 1 \wedge z{=}z'{+}y \wedge mul(x',y,z') \to mul(x,y,z)\}.
\end{aligned}
$$

В таком случае, правилами символа $mul^2 \stackrel{\text{def}}{=} G(\{mul \mapsto 2\})$ являются

$$\begin{aligned} rules(mul^2) = \{&x_1 = 0 \wedge z_1 = 0 \wedge x_2 = 0 \wedge z_2 = 0 \rightarrow mul^2(x_1,y_1,z_1,x_2,y_2,z_2),\\ &x_1 > 0 \wedge x_1' = x_1 - 1 \wedge z_1 = y_1 + z_1' \wedge\\ &\wedge x_2 = 0 \wedge z_2 = 0 \wedge mul^2(x_1',y_1,z_1',x_2,y_2,z_2) \rightarrow mul^2(x_1,y_1,z_1,x_2,y_2,z_2),\\ &x_2 > 0 \wedge x_2' = x_2 - 1 \wedge z_2 = y_2 + z_2' \wedge\\ &\wedge x_1 = 0 \wedge z_1 = 0 \wedge mul^2(x_1,y_1,z_1,x_2',y_2,z_2') \rightarrow mul^2(x_1,y_1,z_1,x_2,y_2,z_2),\\ &x_1 > 0 \wedge x_1' = x_1 - 1 \wedge z_1 = y_1 + z_1' \wedge x_2 > 0 \wedge\\ &\wedge x_2' = x_2 - 1 \wedge z_2 = y_2 + z_2' \wedge mul^2(x_1',y_1,z_1',x_2',y_2,z_2') \rightarrow mul^2(x_1,y_1,z_1,x_2,y_2,z_2)\}. \end{aligned}$$

Определение 15. Пусть $\mathcal{S}$ — система дизъюнктов Хорна с ограничениями, и пусть для некоторого дизъюнкта $C \in \mathcal{S}$ выполнено следующее условие:

$$C \equiv \varphi \wedge \bigwedge NRec\,(C) \wedge \bigwedge Rec\,(C) \rightarrow head(C)$$
$$NRec\,(C) = \{R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)\}.$$

Синхронизацией дизъюнкта C назовём множество дизъюнктов $\mathcal{S}'$ над $\mathcal{R}'$, полученных из $\mathcal{S}$ добавлением новых правил и заменой в C атомов $NRec\,(C)$ на их произведение:

$\mathcal{S}' \stackrel{\text{def}}{=} \mathcal{S} \setminus \{C\} \cup \{C'\} \cup \bigcup\limits_{R \in N} rules(R)$, где

$C' \stackrel{\text{def}}{=} \varphi \wedge \prod\limits_{i=1}^{m} R_i(\overline{x}_i) \wedge \bigwedge Rec\,(C) \rightarrow head(C)$,

N — наименьшее по включению множество символов, содержащее $G(\langle R_1, \ldots, R_m \rangle)$ и все символы из $\mathcal{R}'$, входящие в рекурсивные атомы правил символов из N

Пример 10. Для системы $\mathcal{S}$ в примере 7 с запросом

$$C \equiv x = x' \wedge y = y' \wedge z \neq z' \wedge mul(x,y,z) \wedge mul(x',y',z') \rightarrow \bot$$

синхронизацией C является система $\mathcal{S}' \stackrel{\text{def}}{=} rules(mul) \cup rules(mul^2) \cup \{C'\}$, где

$$C' \equiv x = x' \wedge y = y' \wedge z \neq z' \wedge mul^2(x,y,z,x',y',z') \rightarrow \bot.$$

Заметим, что в этом примере система $\mathcal{S}'$ выполнима, также как и $\mathcal{S}$. Однако в отличие от $\mathcal{S}$, система $\mathcal{S}'$ имеет сертификат выполнимости в линейной целочисленной арифметике:

$$\mathcal{I} \stackrel{\text{def}}{=} \{mul \mapsto \top, mul^2 \mapsto (x_1 = x_2 \wedge y_1 = y_2 \rightarrow z_1 = z_2)\}.$$

Очевидно, что множество N в определении 15 является множеством вершин в компоненте сильной связности символа $G(\langle R_1, \ldots, R_m \rangle)$ графа зависимостей синхронизированной системы, т.е. N содержит все символы, рекурсивные с $G(\langle R_1, \ldots, R_m \rangle)$ в синхронизированной системе. Рассмотрим это более подробно на следующем примере.

Пример 11. Пусть $\mathcal{S}$ — следующая система дизъюнктов над $\mathcal{R} = \{even, odd\}$:

$$\begin{aligned} x = 0 &\to even\,(x) \\ x > 0 \wedge odd\,(x-1) &\to even\,(x) \\ even\,(y-1) &\to odd\,(y) \\ even\,(x) \wedge odd\,(x) &\to \bot \end{aligned}$$

Пусть на $\mathcal{R}$ зафиксирован линейный порядок $even < odd$. Тогда синхронизацией запроса системы $\mathcal{S}$ является следующая система:

$$\begin{aligned} x = 0 &\to even\,(x) \\ x > 0 \wedge odd\,(x-1) &\to even\,(x) \\ even\,(y-1) &\to odd\,(y) \\ x = 0 \wedge eveneven\,(x, y-1) &\to evenodd\,(x, y) \\ x > 0 \wedge evenodd\,(y-1, x-1) &\to evenodd\,(x, y) \\ x_1 = 0 \wedge x_2 = 0 &\to eveneven\,(x_1, x_2) \\ x_1 > 0 \wedge x_2 = 0 \wedge evenodd\,(x_2, x_1-1) &\to eveneven\,(x_1, x_2) \\ x_1 = 0 \wedge x_2 > 0 \wedge evenodd\,(x_1, x_2-1) &\to eveneven\,(x_1, x_2) \\ x_1 = 0 \wedge x_2 = 0 \wedge oddodd\,(x_1-1, x_2-1) &\to eveneven\,(x_1, x_2) \\ eveneven\,(x_1-1, x_2-1) &\to oddodd\,(x_1, x_2) \\ evenodd\,(x, x) &\to \bot \end{aligned}$$

Множество N из определения 15 в данном случае выглядит следующим образом: $\{evenodd, eveneven, oddodd\}$.

Оставшаяся часть данного раздела будет посвящена доказательству того, что синхронизация любого дизъюнкта является конечным множеством дизъюнктов.

Размером неинтерпретированного символа $R \in \mathcal{R}'$ назовём мощность мультимножества в его G-прообразе, т.е.

$$size\,(R) \stackrel{\text{def}}{=} \sum_{P \in \mathcal{R}} r(P), \text{ где } r = G^{-1}(R).$$

Заметим, что по определению G, для всех символов $P \in \mathcal{R}$ справедливо $G(\{P \mapsto 1\}) = P$, т.е. размеры всех символов в $\mathcal{R}$ равны 1.

Лемма 3. Пусть $C_1, \ldots, C_m \in \mathcal{S} \setminus queries$ и пусть $C = C_1 \times \ldots \times C_m$. Тогда размеры всех символов, входящих в рекурсивные атомы дизъюнкта C, равны m.

Доказательство. По определению 14 имеем следующее:

$$Rec\,(C) = \Big\{ \prod_{i=1}^{m} a_i \;\Big|\; \langle a_1, \ldots a_m \rangle \in A \Big\} \setminus \{head(C)\},$$
$$\text{где } A \in \big\lceil Rec_{/head}\,(C_1), \ldots, Rec_{/head}\,(C_m) \big\rceil.$$

По определению произведения неинтерпретированных атомов очевидно, что если $P(\overline{x}) \equiv \prod_{i=1}^{m} R_i(\overline{x}_i)$ для некоторых неинтерпретированных атомов $R_1(\overline{x}_1), \ldots, R_1(\overline{x}_m)$ системы $\mathcal{S}$, то $size\,(P) = size\,(G(\langle R_1, \ldots, R_m \rangle)) = m$. Так как и заголовок, и рекурсивные атомы являются произведениями m атомов, то размер символа в заголовке C равен m, а также размеры всех рекурсивных символов в теле C равны m. □

Теорема 1. Система $\mathcal{S}'$ из определения 15 является системой дизъюнктов Хорна.

Доказательство. Нужно показать, что $\mathcal{S}'$ — конечное множество дизъюнктов. Очевидно, для этого достаточно показать, что множество N содержит конечное число символов.

Из леммы 3 и (2.2) следует, что каким бы ни был символ $R \in \mathcal{R}'$, размеры всех рекурсивных символов для всех дизъюнктов в $rules(R)$ равны $size\,(R)$. Следовательно, размеры всех символов в множестве N равны $size\,(G\langle R_1, \ldots, R_m \rangle) = m$.

Поскольку существует лишь конечное множество мультимножеств символов из конечного множества $\mathcal{R}$ мощности не более m, и $G : \mathbb{N}^{\mathcal{R}} \to \mathcal{R}'$ является биекцией, то существует лишь конечное число символов из $\mathcal{R}'$ размера m. □

2.2 Деревья выводов синхронизированных систем

В данном разделе обсуждается связь между деревьями гиперрезолютивного вывода системы и её синхронизацией. Наличие этой связи будет формально доказано в теоремах 2 и 3. Схема этих доказательств проиллюстрирована на рис. 4: синхронизация "склеивает" различные ветви деревьев, не меняя самих формул в родителях "склеиваемых" узлов. Важным следствием наличия этой связи является эквивыполнимость системы и её синхронизации (теорема 4).

Теорема 2. Пусть $R_1, \ldots, R_m \in \mathcal{R}$, $C_1 \in \mathit{rules}(R_1), \ldots, C_m \in \mathit{rules}(R_m)$. Тогда $(C_1, \psi_1), \ldots, (C_m, \psi_m)$ будут корнями некоторых деревьев вывода тогда и только тогда, когда $(C_1 \times \ldots \times C_m, \psi)$ для $\psi \sim \psi_1 \wedge \cdots \wedge \psi_m$ является корнем некоторого дерева вывода.

Доказательство. ***Необходимость.*** Пусть $T_1, \ldots, T_m$ — деревья вывода с корнями $(C_1, \psi_1), \ldots, (C_m, \psi_m)$ соответственно. Пусть h_i — высота дерева T_i, т.е. число рёбер в самом длинном пути от корня до листа в дереве T_i. Докажем утверждение теоремы, воспользовавшись возвратной индукцией по $h = \sum_{i=1}^{m} h_i$.

База. Высота дерева равна нулю $h = 0$. Если $\sum_{i=1}^{m} h_i$, то $h_i = 0$ для всех i, т.е. корни $(C_1, \psi_1), \ldots, (C_m, \psi_m)$ являются листьями. По определению дерева вывода, для всех $i \in \{1, \ldots, m\}$ справедливо $C_i \equiv \psi_i \to R_i(\overline{v}_{R_i})$. По определению произведения дизъюнктов имеем:

$$C \stackrel{\text{def}}{=} C_1 \times \cdots \times C_m \equiv (\psi_1 \wedge \cdots \wedge \psi_m \to R\,(\overline{v}_R)) \text{ для } R = G(\langle R_1, \ldots, R_m \rangle).$$

По определению дерева вывода $(C, \mathit{body}(C)) = (C_1 \times \ldots \times C_m, \psi_1 \wedge \cdots \wedge \psi_m)$ является корнем и листом дерева вывода высоты 0.

Переход. Пусть $C_i \equiv \big(\varphi_i \wedge R_1^i(\overline{x}_1^i) \wedge \cdots \wedge R_{k_i}^i(\overline{x}_{k_i}^i) \to R_i(\overline{v}_{R_i})\big)$. Пусть $(C_1^i, \psi_1^i), \ldots, (C_{k_i}^i, \psi_{k_i}^i)$ — дочерние узлы корня в дереве вывода T_i, такие, что $C_j^i \in \mathit{rules}(R_j^i)$. По определению дерева вывода имеем:

$$\psi_i \equiv \varphi_i \wedge \psi_1^i[\overline{x}_1^i / \overline{v}_{R_1^i}] \wedge \cdots \wedge \psi_{k_i}^i[\overline{x}_{k_i}^i / \overline{v}_{R_{k_i}^i}] \tag{2.3}$$

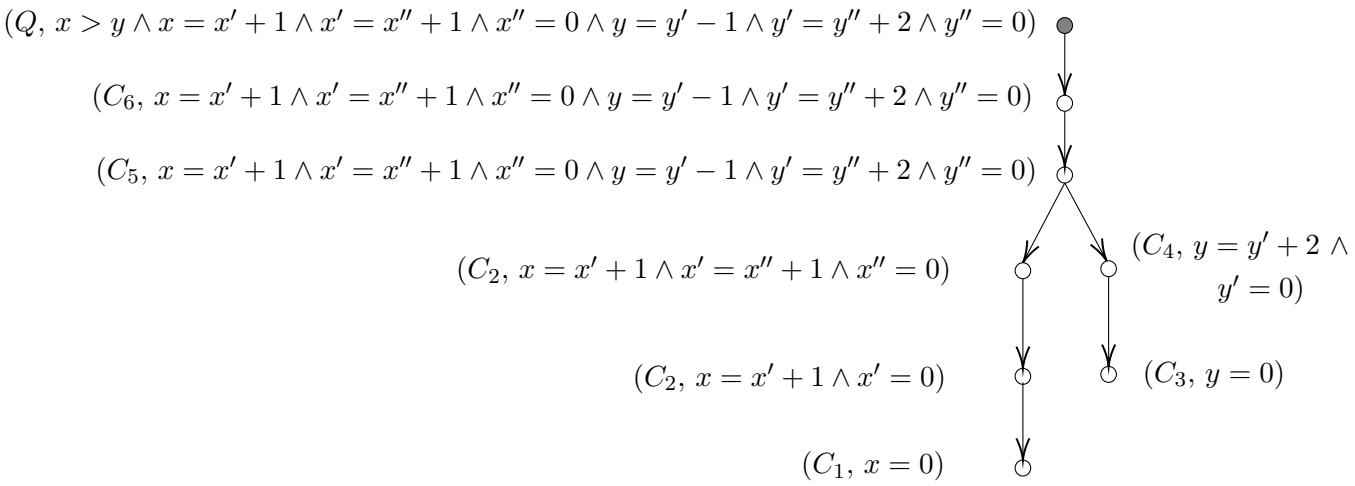
Сопоставим каждому неинтерпретированному атому в теле дизъюнкта C дерево вывода высоты не более h. Пусть $P(\overline{x})$ — неинтерпретированный атом

$$C_1 \equiv x = 0 \to P(x)$$
$$C_2 \equiv x = x' + 1 \wedge P(x') \to P(x)$$
$$C_3 \equiv y = 0 \to Q(y)$$
$$C_4 \equiv y = y' + 2 \wedge Q(y') \to Q(y)$$
$$C_5 \equiv P(x) \wedge Q(y) \to R(x,y)$$
$$C_6 \equiv y = y' - 1 \wedge R(x,y') \to R(x,y)$$
$$Q \equiv x > y \wedge R(x,y) \to \bot$$

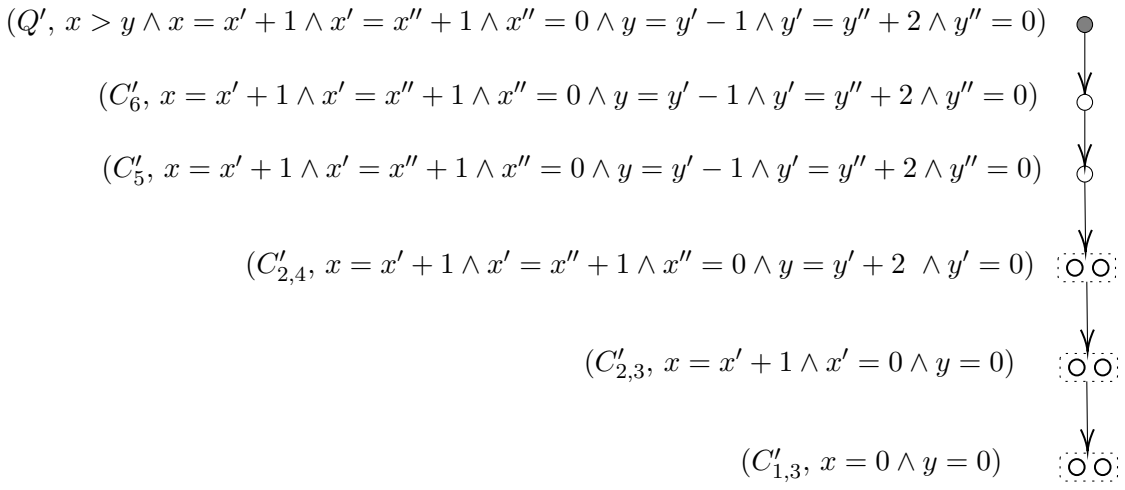
а) Система $\mathcal{S}$

$$C'_{1,3} \equiv x = 0 \wedge y = 0 \to PQ(x,y)$$
$$C'_{2,3} \equiv x = x' + 1 \wedge y = 0 \wedge PQ(x',y) \to PQ(x,y)$$
$$C'_{1,4} \equiv x = 0 \wedge y = y' + 2 \wedge PQ(x,y') \to PQ(x,y)$$
$$C'_{2,4} \equiv x = x' + 1 \wedge y = y' + 2 \wedge PQ(x',y') \to PQ(x,y)$$
$$C'_5 \equiv PQ(x,y) \to R(x,y)$$
$$C'_6 \equiv y = y' - 1 \wedge R(x,y') \to R(x,y)$$
$$Q' \equiv x > y \wedge R(x,y) \to \bot$$

б) Синхронизация дизъюнкта C_5



в) Гиперрезолютивное опровержение $\mathcal{S}$



г) Гиперрезолютивное опровержение синхронизации

Рис. 4. Гиперрезолютивные опровержения системы и её синхронизации

в теле дизъюнкта $C \stackrel{\text{def}}{=} C_1 \times \cdots \times C_m$. Тогда либо $P(\overline{x}) \in NRec\,(C)$, либо $P(\overline{x}) \in Rec\,(C)$.

Пусть $P(\overline{x}) \in NRec\,(C)$. Тогда по определению 14 $NRec\,(C) = \bigcup_{i=1}^{m} NRec\,(C_i)$, следовательно, $P(\overline{x}) = R^i_j(\overline{x}^i_j)$ для некоторых i и j. Сопоставим этому атому $P(\overline{x})$ поддерево с корнем (C^i_j, ψ^i_j).

Пусть $P(\overline{x}) \in Rec\,(C)$. По определению 14 имеем:

$$P(\overline{x}) = \prod_{i=1}^{m} R^i_{j_i}(\overline{x}^i_{j_i}) \text{ для таких } j_1, \ldots, j_m, \text{ что } R^i_{j_i}(\overline{x}^i_{j_i}) \in Rec_{/head}\,(C_i)\,,$$

причём хотя бы один из $R^i_{j_i}(\overline{x}^i_{j_i}) \in Rec\,(C_i)$.

Рассмотрим следующие деревья вывода $U_1, \ldots, U_m$: если $R^i_{j_i} \in Rec\,(C_i)$, то пусть U_i будет поддеревом с корнем $(C^i_{j_i}, \psi^i_{j_i})$; если $R^i_{j_i}(\overline{x}^i_{j_i}) = head(C_i)$, то положим U_i деревом T_i. Корень дерева U_i будем обозначать как $(D_i^{P(\overline{x})}, \eta_i^{P(\overline{x})})$, т.е.

$$(D_i^{P(\overline{x})}, \eta_i^{P(\overline{x})}) \stackrel{\text{def}}{=} \begin{cases} (C^i_{j_i}, \psi^i_{j_i}) & \text{если } R^i_{j_i}(\overline{x}^i_{j_i}) \in Rec\,(C_i)\,, \\ (C_i, \psi_i) & \text{если } R^i_{j_i}(\overline{x}^i_{j_i}) = head(C_i). \end{cases}$$

Тогда для всех i верно, что $D_i^{P(\overline{x})} \in rules(R^i_{j_i})$. Поскольку $head(C_1 \times \ldots \times C_m) = \prod_{i=1}^{m} head(C_i) \notin Rec\,(C_1 \times \ldots \times C_m)$, имеем, что для некоторого i справедливо $R^i_{j_i}(\overline{x}^i_{j_i}) \in Rec\,(C_i)$. Следовательно, как минимум, одно из деревьев U_i является собственным поддеревом T_i, т.е. сумма высот деревьев $U_1, \ldots, U_m$ строго меньше h.

По индукционному предположению существует дерево вывода U с корнем $(D_1^{P(\overline{x})} \times \cdots \times D_m^{P(\overline{x})}, \eta_1^{P(\overline{x})} \wedge \cdots \wedge \eta_m^{P(\overline{x})})$. Поскольку $P = G(\langle R^1_{j_1}, \ldots, R^m_{j_m} \rangle)$, то в силу (2.2) имеем $D_1^{P(\overline{x})} \times \cdots \times D_m^{P(\overline{x})} \in rules(P)$. Сопоставим атому $P(\overline{x})$ дерево U.

Итак, каждому $P(\overline{x})$ в теле $C_1 \times \cdots \times C_m$ сопоставлено дерево вывода U с корнем (D, ψ), где $D \in rules(P)$. Для произведения дизъюнктов $C_1 \times \cdots \times C_m$ в силу определения 14 ограничением является $\varphi_1 \wedge \cdots \wedge \varphi_m$. Тело ψ гиперрезольвенты дизъюнкта $C_1 \times \cdots \times C_m$ и дизъюнктов $\psi \to head(D)$ всех таких

деревьев U выглядит следующим образом:

$$\varphi_1 \wedge \cdots \wedge \varphi_m \wedge \bigwedge_{i=1}^{m} \bigwedge_{R_j^i(\overline{x}_j^i) \in NRec(C_i)} \psi_j^i[\overline{x}_j^i/\overline{v}_{R_j^i}] \wedge \bigwedge_{P(\overline{x}) \in Rec(C)} (\eta_1^{P(\overline{x})} \wedge \cdots \wedge \eta_m^{P(\overline{x})})[\overline{x}/\overline{v}_P] \sim$$
$$\sim \varphi_1 \wedge \cdots \wedge \varphi_m \wedge \bigwedge_{i=1}^{m} \bigwedge_{R_j^i(\overline{x}_j^i) \in NRec(C_i)} \psi_j^i[\overline{x}_j^i/\overline{v}_{R_j^i}] \wedge \bigwedge_{i=1}^{m} \bigwedge_{R_j^i(\overline{x}_j^i) \in Rec(C_i)} \psi_j^i[\overline{x}_j^i/\overline{v}_{R_j^i}]. \tag{2.4}$$

Из (2.3) получаем, что тело гиперрезольвенты ψ, заданное в (2.4), можно записать так: $\psi_1 \wedge \cdots \wedge \psi_m$. Следовательно, дерево с корнем $(C_1 \times \cdots \times C_m, \psi)$ и поддеревьями, сопоставленными неинтерпретированным атомам в теле $C_1 \times \cdots \times C_m$, является деревом вывода.

Достаточность. Доказательство достаточности аналогично доказательству необходимости, поэтому будет представлена лишь его схематичная версия. Доказательство проводится с помощью индукции по высоте дерева T с корнем $(C_1 \times \ldots \times C_m, \psi)$. Если $h = 0$, то $C_1 \times \ldots \times C_m$ — ограниченный факт, следовательно, дизъюнкты C_i — тоже ограниченные факты $\varphi_i \to R_i(\overline{v}_{R_i})$, и (C_i,φ_i) — деревья вывода; при этом $\psi = \varphi_1 \wedge \cdots \wedge \varphi_m$.

Для доказательства индукционного перехода построим деревья $T_1, \ldots, T_m$ следующим образом: каждому нерекурсивному атому дизъюнкта C_i сопоставим поддерево T, соответствующее этому атому в теле $C_1, \ldots, C_m$. Для каждого атома $R_j^i(\overline{x}_j^i) \in Rec\,(C_i)$ выберем какой-нибудь атом $P(\overline{x}) = \prod_{k=1}^{m} Q_i(\overline{x}_j) \in Rec\,(C_1 \times \ldots \times C_m)$ для некоторого $Q_i = R_j^i$ (такой атом существует, т.к. G-прообразы рекурсивных символов в теле $C_1 \times \cdots \times C_m$ накрывают рекурсивные атомы дизъюнктов $C_1, \ldots, C_m$). Воспользуемся индукционным предположением для поддерева вывода U дерева T, соответствующего атому $P(\overline{x})$, и получим дерево вывода с корнем (D_i,η_i) для $D_i \in rules(R_j^i)$, которое сопоставим атому $R_j^i(\overline{x}_j^i)$. Корень (C_i,ψ_i) искомого дерева вывода получим гиперрезольвентой дизъюнктов, соответствующих деревьям, сопоставленным атомам в теле C_i. Конъюнкция всех таких ψ_i является ψ, как было показано в доказательстве необходимости. □

Теорема 3. Пусть $\mathcal{S}$ — система дизъюнктов Хорна с ограничениями, $C \in \mathcal{S}$, $\mathcal{S}'$ — синхронизация дизъюнкта C. У системы $\mathcal{S}$ существует резолютивное опровержение тогда и только тогда, когда такое же опровержение существует у системы $\mathcal{S}'$.

Доказательство. Пусть C' — дизъюнкт Хорна из определения 15, который получен из C, т.е. если

$$C \equiv \varphi \wedge R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m) \wedge \bigwedge Rec\,(C) \to head(C),$$

где $NRec\,(C) = \{R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)\}$, то

$$C' \equiv \varphi \wedge R(\overline{x}) \wedge \bigwedge Rec\,(C) \to head(C), \text{ где } R(\overline{x}) = \prod_{i=1}^{m} R_i(\overline{x}_i).$$

Докажем более сильное утверждение: у системы $\mathcal{S}$ существует дерево вывода с корнем (D,ψ) тогда и только тогда, когда у системы $\mathcal{S}'$ существует дерево вывода с корнем (D',ψ') такое, что $\psi \sim \psi'$, и выполнено следующее утверждение:

$$D' = \begin{cases} D & \text{если } D \neq C \\ C' & \text{если } D = C. \end{cases}$$

Утверждение теоремы легко следует из этого утверждения: D — запрос тогда и только тогда, когда D' — запрос, а ψ и ψ' равносильны, поэтому, выполнимы или невыполнимы в теории $\mathcal{T}$ одновременно. Докажем только необходимость, достаточность доказывается аналогично. Пусть T — дерево вывода с корнем (D,ψ). Выполним доказательство индукцией по высоте дерева T. Если $h = 0$, то D — ограниченный факт, и тогда даже если $D = C$, то $C' = C = D$, следовательно, (D,ψ) — дерево вывода системы $\mathcal{S}'$. Пусть утверждение верно для всех поддеревьев $T_1, \ldots, T_k$ с узлами в детях $(C_1,\psi_1), \ldots, (C_k,\psi_k)$ корня дерева T, т.е. существуют деревья вывода $(C'_1,\psi'_k), \ldots, (C'_i,\psi'_k)$ такие, что $\psi_i \sim \psi'_i$, и либо $C'_i = C_i$, либо $C'_i = C'$. По определению дерева вывода $\psi \to head(D)$ является гиперрезольвентой D и дизъюнктов $\psi_1 \to head(C_1), \ldots, \psi_k \to head(C_k)$.

Рассмотрим два случая: $D \neq C$ и $D = C$. Если $D \neq C$, то положим ψ' ограничением гиперрезольвенты D и дизъюнктов $\psi'_1 \to head(C'_1), \ldots, \psi'_k \to head(C'_k)$. Очевидно, что $\psi' \sim \psi$. Тогда дерево с корнем (D,ψ') и дочерними поддеревьями $T_1, \ldots, T_k$ является искомым деревом вывода системы $\mathcal{S}'$. Пусть $D = C$. Разобьём поддеревья $T_1, \ldots, T_k$ на два множества: множество $\{T_{i_1}, \ldots, T_{i_m}\}$ соответствует нерекурсивным атомам $R(\overline{x}_1), \ldots, R(\overline{x}_m)$ дизъюнкта C, и $\{T_j \mid j \notin \{i_1, \ldots, i_m\}\}$ — остальные поддеревья. По теореме 3 существует дерево вывода с корнем $(C_{i_1} \times \cdots \times C_{i_m}, \eta)$ такое, что $\eta \sim \psi_{i_1} \wedge \cdots \wedge \psi_{i_m}$. В силу (2.2) имеем $C_{i_1} \times \cdots \times C_{i_m} \in rules(R)$. Следовательно, у дизъюнкта C' и дизъюнктов $\eta \to R(\overline{v}_R)$ и $\psi'_j \to head(C'_j)$, где $j \notin \{i_1, \ldots, i_m\}$, существует гиперрезольвента $\psi' \to head(C)$. При этом очевидно, что $\psi' \sim \psi$, а значит дерево с

корнем (C', ψ') и множеством дочерних поддеревьев $\{U\} \cup \{T_j \mid j \notin \{i_1, \ldots, i_m\}\}$ есть искомое дерево вывода системы $\mathcal{S}'$. □

Теорема 4. Пусть $\mathcal{S}$ — система дизъюнктов Хорна с ограничениями, $C \in \mathcal{S}$, $\mathcal{S}'$ — синхронизация C. Система $\mathcal{S}$ выполнима тогда и только тогда, когда выполнима система $\mathcal{S}'$.

Эта теорема является легко доказывается на основании предыдущей теоремы, а также утверждения 3 о корректности и полноте исчисления гиперрезолюций.

2.3 Сертификаты выполнимости синхронизированных систем

Синхронизирующее преобразование обладает важным свойством, которое будет доказано в этом разделе: если у системы существует сертификат выполнимости, то он существует также и у любой её синхронизации (как будет показано в разделе 2.6, обратное не верно). Таким образом, можно отказаться от исходной системы и искать символьные модели для её синхронизаций; при этом ситуация с представимостью моделей системы гарантированно не ухудшится.

Пусть $\mathcal{S}'$ — синхронизация дизъюнкта $C \in \mathcal{S}$, т.е. $\mathcal{S}' = \mathcal{S} \setminus \{C\} \cup \{C'\} \cup \bigcup_{R \in N} \mathit{rules}(R)$, где C' и N — дизъюнкт и множество символов из определения 15 соответственно.

Введём следующее обозначение: $\varphi \overset{+}{\wedge} \psi$ будет означать конъюнкцию φ и ψ, которая гарантирует, что свободные переменные операндов в $\varphi \wedge \psi$ различны:

$$\varphi(\overline{x}) \overset{+}{\wedge} \psi(\overline{y}) \stackrel{\text{def}}{=} (\varphi \wedge \psi)\,(\overline{x} \uplus \overline{y})\,.$$

Теорема 5. Пусть $\mathcal{I}$ — сертификат выполнимости $\mathcal{S}$. Тогда следующая символьная интерпретация является сертификатом выполнимости для системы $\mathcal{S}'$:

$$\mathcal{I}'(R) \stackrel{\text{def}}{=} \bigoverset{+}{\bigwedge_{P \in \mathcal{R}}} \bigoverset{+}{\bigwedge_{i=1}^{r(P)}} \mathcal{I}(P).$$

Доказательство. Прежде всего заметим, что для каких бы то ни было атомов $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)$ с $R_i \in \mathcal{R}$ верно следующее утверждение:

$$\left[\!\!\left[\prod_{i=1}^{m} R_i(\overline{x}_i) \right]\!\!\right]_{\mathcal{I}'} = [\![R_1(\overline{x}_1)]\!]_{\mathcal{I}} \wedge \cdots \wedge [\![R_m(\overline{x}_m)]\!]_{\mathcal{I}}\,. \tag{2.5}$$

По определению 8 справедливо следующее утверждение: $\mathcal{T} \models \bigwedge_{D\in\mathcal{S}} \forall[\![D]\!]_{\mathcal{I}}$. Поскольку $G(\{R \mapsto 1\}) = R$ для каждого $R \in \mathcal{R}$, заметим, что $\mathcal{I}'(R) = \mathcal{I}(R)$. В частности, $\mathcal{T} \models \bigwedge_{D\in\mathcal{S}\setminus\{C\}} \forall[\![D]\!]_{\mathcal{I}'}$. Таким образом осталось показать, что $\mathcal{T} \models \forall[\![C']\!]_{\mathcal{I}'}$, а также, что для всех $P \in N$ выполнено следующее утверждение:

$$\mathcal{T} \models \bigwedge_{D\in rules(P)} \forall[\![D]\!]_{\mathcal{I}'}. \tag{2.6}$$

Первое утверждение следует в силу того, что по (2.5) справедливо $[\![R(\overline{x})]\!]_{\mathcal{I}'} = [\![R_1(\overline{x}_1)]\!]_{\mathcal{I}} \wedge \cdots \wedge [\![R_m(\overline{x}_m)]\!]_{\mathcal{I}}$, и поэтому $[\![C']\!]_{\mathcal{I}'} = [\![C]\!]_{\mathcal{I}}$ (обозначения неинтерпретированных атомов взяты из определения 15). Для доказательства второго утверждения докажем более общий факт — (2.6) выполняется для любого $R \in \mathcal{R}'$. Пусть $D \in rules(R)$. Тогда в силу (2.2) имеем, что $D = C_1 \times \cdots \times C_m$ для некоторых $C_1, \ldots, C_m \in \mathcal{S}$, возможно с переменными, переименованными таким образом, что $C_1 \overset{+}{\wedge} \ldots \overset{+}{\wedge} C_m = C_1 \wedge \cdots \wedge C_m$. Заметим, что по определению 14 и (2.6) справедливо следующее:

$$\begin{aligned} &[\![head(D)]\!]_{\mathcal{I}'} = [\![head(C_1)]\!]_{\mathcal{I}} \wedge \cdots \wedge [\![head(C_m)]\!]_{\mathcal{I}}\,, \text{ и} \\ &[\![body(D)]\!]_{\mathcal{I}'} = [\![body(C_1)]\!]_{\mathcal{I}} \wedge \cdots \wedge [\![body(C_m)]\!]_{\mathcal{I}}\,. \end{aligned} \tag{2.7}$$

Но так как $\mathcal{I}$ — это сертификат выполнимости $\mathcal{S}$, то имеем $\mathcal{T} \models \bigwedge_{i=1}^{m} \forall[\![C_i]\!]_{\mathcal{I}}$, а значит, что $\mathcal{T} \models \forall \bigwedge_{i=1}^{m} ([\![body(C_i)]\!]_{\mathcal{I}} \to [\![head(C_i)]\!]_{\mathcal{I}})$. Следовательно, справедливо следующее утверждение:

$$\mathcal{T} \models \forall \bigwedge_{i=1}^{m} \Big(\Big[\!\Big[\bigwedge_{j=1}^{m} body(C_j) \Big]\!\Big]_{\mathcal{I}} \to [\![head(C_i)]\!]_{\mathcal{I}} \Big).$$

Это утверждение может быть переписано следующим образом:

$$\mathcal{T} \models \forall \Big(\Big[\!\Big[\bigwedge_{j=1}^{m} body(C_j) \Big]\!\Big]_{\mathcal{I}} \to [\![\bigwedge_{i=1}^{m} head(C_i)]\!]_{\mathcal{I}} \Big).$$

Наконец, в силу (2.7) имеем $\mathcal{T} \models \forall([\![body(D)]\!]_{\mathcal{I}} \to [\![head(D)]\!]_{\mathcal{I}})$, то есть $\mathcal{T} \models \forall[\![D]\!]_{\mathcal{I}}$. □

2.4 Семантика неподвижной точки

В данном разделе обсуждается стандартный подход к заданию денотационной семантики систем дизъюнктов Хорна как поиска наименьшей неподвижной точки её отношений переходов. Этот подход является продуктивным:

с помощью него будет показана теорема о полноте исчисления резолюций (см. раздел 2.4.4), доказана непредставимость сертификатов выполнимости системы в примере 7 (см. раздел 2.6), и, главное, синхронизация дизъюнктов получит более простое, семантическое толкование (см. раздел 2.5).

2.4.1 Отношение переходов системы дизъюнктов

Пусть фиксированы система $\mathcal{S}$ над неинтерпретированными символами $\mathcal{R} = \{P_1, \ldots, P_n\}$ и $\mathcal{M}$ — некоторая модель теории $\mathcal{T}$. Пусть k_i является местностью символа P_i, где $i = 1, \ldots, n$. Введём следующее обозначение: $\mathcal{P} \stackrel{\text{def}}{=} 2^{|\mathcal{M}|^{k_1}} \times \cdots \times 2^{|\mathcal{M}|^{k_n}}$.

Определим оператор $T : \mathcal{P} \to \mathcal{P}$ для $X_i \subseteq |\mathcal{M}|^{k_i}$ следующим образом:

$$T(\langle X_1, \ldots, X_n \rangle) \stackrel{\text{def}}{=} \langle Y_1, \ldots, Y_n \rangle, \text{ где}$$
$$Y_i \stackrel{\text{def}}{=} (\mathcal{M}, \langle X_1, \ldots, X_n \rangle)\, (\exists \overline{\ell}_{P_i}.\mathit{body}(P_i)). \tag{2.8}$$

При этом нотация $\left(\mathcal{M}, \overline{X}\right)(\exists \overline{\ell}_{P_i}.\mathit{body}(P_i))$ означает множество кортежей из $|\mathcal{M}|^{k_i}$, которые являются выполняющими оценками переменных $\overline{v}_{P_i}$ в структуре $\left(\mathcal{M}, \overline{X}\right)$ даже в том случае, если *не все переменные из* $\overline{v}_{P_i}$ *входят в* $\mathit{body}(P_i)$. Например, можно считать, что эта запись используется обычным образом, но все переменные x из $\overline{v}_{P_i}$, отсутствующие в $\mathit{body}(P_i)$, добавляются туда приписыванием конъюнкции $\ldots \wedge x = x$.

Заметим, что при фиксированной структуре $\mathcal{M}$ оператор T определяется только множеством *правил* системы $\mathcal{S}$ и не зависит от множества её *запросов*. На практике множество правил $\mathcal{S}$ является логическим описанием поведения некоторой программы с модулями $P_1, \ldots, P_n$ (см. раздел 1.3). Поэтому T задаёт *отношения переходов* этой программы со значениями переменных из $|\mathcal{M}|$: Y_i представляет собой множество состояний, в которые может попасть программа P_i за один шаг при условии, что дочерние модули (т.е. модули P_j, входящие в $\mathit{body}(P_i)$) достигают состояний из X_j.

Важная особенность оператора T состоит в том, что он определён на всех отношениях из $\mathcal{P}$. Тем не менее, поведение оператора T на *представимых* отношениях $\overline{X}$ можно описать обычной подстановкой формул вместо неинтерпретированных атомов, что утверждает следующая лемма.

Лемма 4. Пусть $\mathcal{I}$ — символьная интерпретация системы $\mathcal{S}$, $\overline{X} = \{X_1, \ldots, X_n\}$, где $X_i = \mathcal{M}(\mathcal{I}(P_i))$. Тогда справедливо следующее утверждение:

$$T(\overline{X}) = \langle Y_1, \ldots, Y_n \rangle \text{, где}$$
$$Y_i = \mathcal{M}\left(\exists \overline{\ell}_{P_i}. [\![body(P_i)]\!]_{\mathcal{I}}\right).$$

Доказательство. Утверждение, фактически, непосредственно следует из леммы 1. Заметим, что для того, чтобы подстановка $\left[\!\left[\exists \overline{\ell}_{P_i}.body(P_i)\right]\!\right]_{\mathcal{I}}$ была определена, необходимо потребовать, чтобы для всех $P_i, P_j \in \mathcal{R}$ ни одна из переменных $\overline{\ell}_{P_i}$ не входила в $\overline{v}_{P_j}$. Без потери общности, можно считать, что это так и есть, в противном случае переименуем экзистенциальные переменные.

Далее, по определению T в (2.8) имеем следующее:

$$\overline{a} \in Y_i \quad \Leftrightarrow \quad \left(\mathcal{M}, \overline{X}\right) \models \left(\exists \overline{\ell}_{P_i}.body(P_i)\right)(\overline{a}).$$

По лемме 1 и определению подстановки $[\![\cdot]\!]_{\mathcal{I}}$ имеем:

$$\left(\mathcal{M}, \overline{X}\right) \models \left(\exists \overline{\ell}_{P_i}.body(P_i)\right)(\overline{a}) \quad \Leftrightarrow \quad \mathcal{M} \models \left(\exists \overline{\ell}_{P_i} [\![body(P_i)]\!]_{\mathcal{I}}\right)(\overline{a}).$$

Таким образом, мы получили, что $\overline{a} \in Y_i$ тогда и только тогда, когда $\overline{a} \in \mathcal{M}\left(\exists \overline{\ell}_{P_i}. [\![body(P_i)]\!]_{\mathcal{I}}\right)$. □

2.4.2 Неподвижные точки отношения переходов

Известно, что для каждого $k \in \mathbb{N}$ система $(2^{|\mathcal{M}|^k}, \subseteq)$ является полной решёткой: для каждого $W \subseteq \mathcal{P}$ существует наименьшая верхняя грань $\bigcup W$, равная объединению элементов W, и наибольшая нижняя грань $\bigcap W$, равная пересечению элементов W.

Система $(\mathcal{P}, \subseteq)$ также является полной решёткой, так как получается произведением полных решёток; отношение частичного порядка $\subseteq$ на $\mathcal{P}$ определяется как $\langle X_1, \ldots, X_n \rangle \subseteq \langle Y_1, \ldots, Y_n \rangle \Leftrightarrow X_1 \subseteq Y_1$ и $\ldots$ и $X_n \subseteq Y_n$; наименьшая верхняя (наибольшая нижняя) грань множества кортежей — как кортеж наименьших верхних (наибольших нижних) граней соответствующих проекций, т.е. $\bigcup W = \langle \bigcup W_1, \ldots, \bigcup W_n \rangle$, где W_i — множество i-х членов кортежей-элементов W.

Определение 16. $F \in \mathcal{P}$ называется *неподвижной точкой* оператора T, если $T(F) = F$. Далее, F называется *преднеподвижной точкой* T, если

$T(F) \subseteq F$. Наконец, F называется *наименьшей* (пред)неподвижной точкой T, если F является (пред)неподвижной точкой T и для всех $F' \in \mathcal{P}$, если F' — (пред)неподвижная точка T, то $F \subseteq F'$.

Определение 17. Оператор T называется *монотонным*, если для всех $\overline{X}$ и $\overline{X}' \in \mathcal{P}$, из того, что $\overline{X} \subseteq \overline{X}'$, следует, что $T(\overline{X}) \subseteq T(\overline{X}')$.

Утверждение 5. Оператор T является монотонным.

Доказательство. Это утверждение следует из того факта, что все неинтерпретированные символы в $body(P_i)$ входят туда *позитивно*, т.е. без отрицания. Действительно, если предикатный символ P появляется в формуле Φ позитивно, то при увеличении интерпретирующего отношения формула Φ из выполнимой не может стать невыполнимой. □

По теореме Кнастера-Тарского у монотонного оператора на полной решётке существует наименьшая неподвижная точка, т.е. такое отношение $F \subseteq |\mathcal{M}|^k$, что $T(F) = F$. Тем не менее, мы выполним конструктивное доказательство этого факта, проделав шаги для построения F в явном виде.

Определение 18. Для $W \subseteq \mathcal{P}$ обозначим через $T(W)$ образ множества W при отображении T. Назовём T *непрерывным*, если для всех возрастающих цепочек в $\mathcal{P}$ (т.е. таких множеств $W = \{w_1, w_2, \ldots\}$, что для всех $i \geqslant 1$, $w_i \subseteq w_{i+1}$) выполнено следующее:

$$T(\bigcup W) = \bigcup T(W).$$

Утверждение 6. Оператор T является непрерывным.

Доказательство. Во-первых, для любого монотонного оператора верно $\bigcup T(W) \subseteq T(\bigcup W)$. Действительно, по определению верхней грани для всех $w \in W$ имеем $w \subseteq \bigcup W$. В силу монотонности оператора T имеем, что $T(w) \subseteq T(\bigcup W)$ для всех $w \in W$. Другими словами, $T(\bigcup W)$ является верхней гранью множества $T(W)$. Но так как $\bigcup T(W)$ — это наименьшая верхняя грань $T(W)$, то имеем $\bigcup T(W) \subseteq T(\bigcup W)$.

Чтобы показать, что выполнено $T(\bigcup W) \subseteq \bigcup T(W)$, возьмём $\overline{a} = \langle \overline{a}_1, \ldots, \overline{a}_n \rangle \in T(\bigcup W)$ и покажем, что $\overline{a} \in \bigcup T(W)$. Пусть $T(\bigcup W) = \langle Y_1, \ldots, Y_n \rangle$, т.е. для всех i, $\overline{a}_i \in Y_i$.

По определению оператора T имеем следующее:

$$Y_i = (\mathcal{M}, \cup W)\,(\exists \overline{\ell}_{P_i}.body(P_i)) \text{ для всех } i.$$

Значит, для свободных переменных $body(P_i)$ существует оценка $\overline{b}$, пригодная для оценивания переменных $\overline{v}_{P_i}$ в $\overline{a}_i$ и такая, что

$$\left(\mathcal{M}, \bigcup W\right) \models body(P_i)(\overline{b}) \equiv \bigvee_{C \in rules(P_i)} body(C)(\overline{b}).$$

Следовательно, $(\mathcal{M}, \bigcup W) \models C(\overline{b})$ для некоторого $C \in rules(P_i)$. Пусть теперь $body(C) \equiv \varphi \wedge P_{j_1}(\overline{x}_1) \wedge \cdots \wedge P_{j_m}(\overline{x}_m)$. А также, пусть $\overline{b}$ оценивает $\overline{x}_1$ в $\overline{b}_1$, …, $\overline{x}_m$ — в $\overline{b}_m$. Наконец, пусть $\bigcup W = \langle \bigcup W_1, \ldots, \bigcup W_n \rangle$. Тогда выполнено $\overline{b}_1 \in \bigcup W_{j_1}, \ldots, \overline{b}_m \in \bigcup W_{j_m}$, поскольку очевидно, что справедливо $(\mathcal{M}, \langle \bigcup W_1, \ldots, \bigcup W_n \rangle) \models (\varphi \wedge P_{j_1}(\overline{x}_1) \wedge \cdots \wedge P_{j_m}(\overline{x}_m))\,(\overline{b})$.

Поскольку для всех $k \in \{1, \ldots, m\}$ наименьшая верхняя грань $\bigcup W_{j_k}$ есть объединение элементов W_{j_k}, имеем $\overline{b}_k \in X_k$ для некоторого $X_k \in W_{j_k}$.

Далее, поскольку таких X_k конечное число, и элементы W образуют возрастающую цепочку, то существует кортеж $\overline{Z} = \langle Z_1, \ldots, Z_n \rangle \in W$ такой, что $X_k \subseteq Z_{j_k}$. В частности, $\overline{b}_k \in Z_{j_k}$ для всех $k \in \{1, \ldots, m\}$. Следовательно, имеем $\left(\mathcal{M}, \overline{Z}\right) \models body(C)(\overline{b})$, а это означает, что $\left(\mathcal{M}, \overline{Z}\right) \models \left(\exists \overline{\ell}_{P_i}.body(P_i)\right)(\overline{a}_i)$. Таким образом, очевидно, что $\overline{a}_i$ принадлежит i-й компоненте $T(\overline{Z})$.

Вспомним теперь, что $\overline{a} = \langle \overline{a}_1, \ldots, \overline{a}_n \rangle \in T(\bigcup W)$. По предыдущему рассуждению, существуют кортежи $\overline{Z}_1, \ldots, \overline{Z}_n \in W$, что $\overline{a}_i$ принадлежит принадлежит i-й компоненте $T(\overline{Z}_i)$. Поскольку W — возрастающая цепочка, существует кортеж $\overline{Z} \in W$, такой, что $\overline{Z}_i \subseteq \overline{Z}$ для всех $i \in \{1, \ldots, n\}$. По монотонности, $T(\overline{Z}_i) \subseteq T(\overline{Z})$, поэтому $\overline{a} \in T(\overline{Z}) \subseteq \bigcup T(W)$. □

Теперь можно показать, что наименьшая неподвижная точка является наименьшей верхней гранью счётного числа множеств.

Для любого множества $X \in \mathcal{P}$ и $i > 0$ положим

$$T^0(X) \stackrel{\text{def}}{=} X$$
$$T^i(X) \stackrel{\text{def}}{=} T(T^{i-1}(X)).$$

Обозначим также $\overline{\varnothing} \stackrel{\text{def}}{=} \langle \underbrace{\varnothing, \ldots, \varnothing}_{n \text{ шт.}} \rangle$.

Теорема 6. Пусть $L \stackrel{\text{def}}{=} \bigcup_{i \geqslant 0} T^i(\overline{\varnothing})$. Тогда L является наименьшей неподвижной и преднеподвижной точкой оператора T.

Доказательство. Покажем, что $T(L) = L$ (в частности, из этого будет следовать, что L является и преднеподвижной точкой T). Во-первых, докажем, что $\{T^i(\overline{\varnothing})\}_{i \geqslant 0}$ — возрастающая цепочка в $\mathcal{P}$, воспользовавшись индукцией по i. Для $i = 0$, $T^0(\overline{\varnothing}) = \overline{\varnothing} \subseteq T^1(\overline{\varnothing})$. Пусть теперь $T^{i-1}(\overline{\varnothing}) \subseteq T^i(\overline{\varnothing})$. Так как оператор T монотонен, то $T(T^{i-1}(\overline{\varnothing})) \subseteq T(T^i(\overline{\varnothing}))$, т.е. $T^i(\overline{\varnothing}) \subseteq T^{i+1}(\overline{\varnothing})$. В силу непрерывности оператора T имеем следующее:

$$T(L) = T(\bigcup_{i \geqslant 0} T^i(\overline{\varnothing})) = \bigcup_{i \geqslant 0} T(T^i(\overline{\varnothing})) = \bigcup_{i \geqslant 1} T^i(\overline{\varnothing}) = \bigcup_{i \geqslant 0} T^i(\overline{\varnothing}) = L.$$

Теперь докажем, что L является наименьшей преднеподвижной точкой (в частности, из этого будет следовать, что L является и наименьшей неподвижной точкой T). Пусть $T(F) \subseteq F$. Покажем, что для всех $i \geqslant 0$ $T^i(\overline{\varnothing}) \subseteq F$. Воспользуемся индукцией по i. Очевидно, что для $i = 0$ $\overline{\varnothing} \subseteq F$. Пусть $T^i(\overline{\varnothing}) \subseteq F$. По монотонности оператора T имеем $T(T^i(\overline{\varnothing})) \subseteq T(F)$. Таким образом, мы получили, что $T^{i+1}(\overline{\varnothing}) \subseteq T(F) \subseteq F$. Итак, для всех $i \geqslant 0$, $T^i(\overline{\varnothing}) \subseteq F$, т.е. F является верхней гранью множества $\{T^i(\overline{\varnothing})\}_{i \geqslant 0}$. Но так как L — наименьшая верхняя грань этого множества, то выполнено $L \subseteq F$. □

Теорема 6 важна для понимания вычислительной интуиции, которая стоит за системами дизъюнктов Хорна с ограничениями. Наименьшие неподвижные точки отношений переходов системы являются её *денотационной семантикой* [129]. Более формально, $L = \bigcup_{i \geqslant 0} T^i(\overline{\varnothing})$ является денотационной семантикой системы $\mathcal{S}$. Как будет показано далее, $T^i(\overline{\varnothing})$ описывает всевозможные деревья выводов высоты $i - 1$. Поэтому, неформально говоря, L является объединением всех возможных сценариев «поведения», описываемых правилами $\mathcal{S}$. При такой интуиции запросы в $\mathcal{S}$ формируют спецификацию, множество состояний которой должно содержать все состояния в L.

Далее, в этой и следующих главах будет несколько раз перегружаться нотация семантических скобок $[\![\cdot]\!]$. С учётом леммы 4, соединяющей денотационную семантику систем с символьными подстановками, эти перегрузки достаточно естественны и не будут специально комментироваться.

Пусть $L = \bigcup_{i=0} T^i(\overline{\varnothing})$. Тогда i-й элемент L обозначим как $[\![P_i]\!]_{\mathcal{M}}$ и будем называть *семантикой символа* P_i. Другими словами,

$$\langle [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}} \rangle \stackrel{\text{def}}{=} \bigcup_{i=0} T^i(\overline{\varnothing}).$$

Будем называть i-й элемент $T^{b+1}(\overline{\varnothing})$ *ограниченной семантикой символа* P_i *уровня* b и обозначать $[\![P_i]\!]^b_{\mathcal{M}}$:

$$\left\langle [\![P_1]\!]^b_{\mathcal{M}}, \ldots, [\![P_n]\!]^b_{\mathcal{M}} \right\rangle \stackrel{\text{def}}{=} T^{b+1}(\overline{\varnothing}).$$

Таким образом, семантика символа P_i оказывается объединением его ограниченных семантик:

$$[\![P_i]\!]_{\mathcal{M}} = \bigcup_{b \geqslant 0} [\![P_i]\!]^b_{\mathcal{M}}.$$

Неформально говоря, $[\![P_i]\!]^b_{\mathcal{M}}$ есть множество $\mathcal{M}$-состояний, достижимых в программе с главной функцией P_i с условиями верификации $\mathcal{S}$ не более, чем за b шагов, а $[\![P_i]\!]_{\mathcal{M}}$ — множество всех достижимых $\mathcal{M}$-состояний.

Как следует из доказательства теоремы 6, $T^b(\overline{\varnothing})$ формируют возрастающую цепочку. Это означает, что для всех i выполнено следующее:

$$[\![P_i]\!]^0_{\mathcal{M}} \subseteq [\![P_i]\!]^1_{\mathcal{M}} \subseteq [\![P_i]\!]^2_{\mathcal{M}} \subseteq \ldots \subseteq [\![P_i]\!]_{\mathcal{M}}.$$

2.4.3 Неподвижные точки и сертификаты выполнимости[1]

Рассмотрим выполнимую систему дизъюнктов $\mathcal{S}$. По определению выполнимости, каждый дизъюнкт этой системы выполняется в некотором обогащении любой модели $\mathcal{M}$ теории $\mathcal{T}$; в частности, выполняется каждое правило системы. Это можно формализовать следующим образом:

$$\left(\mathcal{M}, \overline{X}\right) \models \forall \overline{v}_{P_i}. \left(\exists \overline{\ell}_{P_i}. body(P_i)\right) \rightarrow P_i(\overline{v}_{P_i}).$$

При этом $\overline{X} = \langle X_1, \ldots, X_n \rangle \in \mathcal{P}$ и $i \in \{1, \ldots, n\}$. Это, в свою очередь, можно переписать так:

$$\left(\mathcal{M}, \overline{X}\right) \left(\exists \overline{\ell}_{P_i}. body(P_i)\right) \subseteq X_i.$$

.

[1]Представленные в данной главе утверждения являются адаптацией результатов, полученных в [130] для логики Хоара.

Используя определение T получаем, что *все правила* $\mathcal{S}$ *выполняются в* $(\mathcal{M}, \overline{X})$ *тогда и только тогда*, когда выполнено следующее условие:

$$T(\overline{X}) \subseteq \overline{X}.$$

А это означает, что X *является преднеподвижной точкой* T.

С этого момента и до конца диссертации будем считать, что *переменные всех неинтерпретированных атомов каждого запроса различны.* Это не ограничивает общности рассуждений. Например, пусть мы имеем следующий запрос:

$$\varphi \wedge P_1(x) \wedge P_2(x) \to \bot.$$

Его можно его переписать в эквивыполнимом виде следующим образом:

$$\varphi \wedge x = y \wedge P_1(x) \wedge P_2(y) \to \bot.$$

Далее заметим, что любой запрос $Q \equiv \varphi \wedge P_{i_1}(\overline{x}_1) \wedge \ldots \wedge P_{i_m}(\overline{x}_m) \to \bot$ можно переписать в эквивалентном виде так:

$$P_{i_1}(\overline{x}_1) \wedge \ldots \wedge P_{i_m}(\overline{x}_m) \to (\forall \overline{y}.\neg\varphi),$$

где $\overline{y}$ — все свободные переменные φ, не входящие в $\overline{x}_1, \ldots, \overline{x}_m$. При этом можно, не теряя общности, считать, что все переменные $\overline{x}_1, \ldots, \overline{x}_m$ появляются в φ (иначе добавим отсутствующие переменные x приписыванием $\ldots \wedge x = x$ к φ).

При тот факт, что переменные $\overline{x}_1, \ldots, \overline{x}_m$ попарно различны, означает следующее:

$$(\mathcal{M}, \langle X_1, \ldots, X_n\rangle)\,(P_{i_1}(\overline{x}_1) \wedge \ldots \wedge P_{i_m}(\overline{x}_m)) = X_{i_1} \times \cdots \times X_{i_m}.$$

Всё вышесказанное доказывает следующую теорему.

Теорема 7.

$$(\mathcal{M}, \langle X_1, \ldots, X_n\rangle) \models \bigwedge_{C \in \mathcal{S}} \forall C$$

тогда и только тогда, когда $\langle X_1, \ldots, X_n\rangle$ — преднеподвижная точка оператора T, и для всех запросов $\varphi(\overline{y}, \overline{x}_1, \ldots, \overline{x}_m) \wedge P_{i_1}(\overline{x}_1) \wedge \ldots \wedge P_{i_m}(\overline{x}_m) \to \bot$ выполнено следующее:

$$X_{i_1} \times \cdots \times X_{i_m} \subseteq \mathcal{M}(\forall \overline{y}.\neg\varphi).$$

Из теорем 7 и 6 вытекает следующая характеристика выполнимых систем.

Теорема 8. Система $\mathcal{S}$ выполнима тогда и только тогда, когда для всякой модели $\mathcal{M}$ теории $\mathcal{T}$ и всякого запроса $\varphi(\overline{y},\overline{x}_1,\ldots,\overline{x}_m)\wedge P_{i_1}(\overline{x}_1)\wedge\ldots\wedge P_{i_m}(\overline{x}_m)\to\bot$ истинно следующее:

$$[\![P_{i_1}]\!]_{\mathcal{M}}\times\cdots\times[\![P_{i_m}]\!]_{\mathcal{M}}\subseteq\mathcal{M}(\forall\overline{y}.\neg\varphi).$$

Теорема 7 даёт ключ к пониманию проблемы представимости сертификатов выполнимости систем дизъюнктов Хорна. По определению, символьная интерпретация $\mathcal{I}$ является сертификатом выполнимости, если $\mathcal{T}\models\bigwedge_{C\in\mathcal{S}}\forall[\![C]\!]_{\mathcal{I}}$. Это можно переформулировать так, что для любой модели $\mathcal{M}$ теории $\mathcal{T}$ выполнено следующее:

$$(\mathcal{M},\langle\mathcal{M}(\mathcal{I}(P_1)),\ldots,\mathcal{M}(\mathcal{I}(P_n))\rangle)\models\bigwedge_{C\in\mathcal{S}}\forall C.$$

По теореме 7, это эквивалентно тому, что $\langle\mathcal{M}(\mathcal{I}(P_1)),\ldots,\mathcal{M}(\mathcal{I}(P_n))\rangle$ — такая преднеподвижная точка T, что для всех запросов $\varphi(\overline{y},\overline{x}_1,\ldots,\overline{x}_m)\wedge P_{i_1}(\overline{x}_1)\wedge\ldots\wedge P_{i_m}(\overline{x}_m)\to\bot$ выполнено следующее:

$$\mathcal{M}(\mathcal{I}(P_{i_1}))\times\cdots\times\mathcal{M}(\mathcal{I}(P_{i_m}))\subseteq\mathcal{M}(\forall\overline{y}.\neg\varphi).$$

В частности, *любой сертификат выполнимости является представлением некоторой неподвижной точки* T. Поэтому проблема вывода сертификатов выполнимости является проблемой поиска представимой неподвижной точки, достаточно маленькой для того, чтобы не пересекаться ни с одним ограничением запросов системы. Эта интуиция позволит лучше понять идею реляционных инвариантов (см. главу 3) и алгоритмы построения реляционных инвариантов (см. главу 4).

2.4.4 Ограниченные семантики и резолютивные опровержения

В данном разделе будет показана связь между семантикой неподвижных точек и резолютивными опровержениями.

Теорема 9. Пусть $P\in\mathcal{R}$, $\mathcal{M}$ — Σ-структура. Тогда

$$[\![P]\!]^h_{\mathcal{M}}=\bigcup_{\substack{(C,\psi)\text{ — корень дерева вывода}\\ \text{высоты не более } h,\\ C\in rules(P),\\ \overline{y}\text{ — все свободные}\\ \text{переменные }\psi\text{, кроме }\overline{v}_P.}}\mathcal{M}(\exists\overline{y}.\psi).$$

Доказательство. Докажем это утверждение индукцией по высоте h. **База** $h = 0$**.** По определению, деревья вывода высоты 0 — это вершины $(C,body(C))$, где D — это ограниченный факт. Возьмём структуру $\mathcal{M}$. По определению ограниченной семантики имеем следующее: $[\![P]\!]^0_{\mathcal{M}} = T(\langle\varnothing,\ldots,\varnothing\rangle)$. Заметим, что $\varnothing = \mathcal{M}(\bot)$. Определим символьную интерпретацию $\mathcal{I}_\bot(R) \stackrel{\text{def}}{=} \bot$. Тогда по лемме 4 i-й член $T(\langle\varnothing,\ldots,\varnothing\rangle)$ равен $\mathcal{M}(\exists\overline{\ell}_P.\,[\![body(P)]\!]_{\mathcal{I}_\bot})$. Вспомним, что $body(P) \equiv \bigvee_{C\in rules(P)} body(C)$. Заметим, что подстановка лжи $[\![body(P)]\!]_{\mathcal{I}_\bot}$ «уничтожает» те элементы дизъюнкции, в которых встречаются неинтерпретированные атомы, т.е. имеем следующее:

$$[\![body(P)]\!]_{\mathcal{I}_\bot} \sim \bigvee_{\substack{C\in rules(P),\\ C\text{ — ограниченный факт}}} body(C).$$

Таким образом, мы получили, что истинно следующее:

$$[\![P]\!]^0_{\mathcal{M}} = \bigcup_{\substack{C\in rules(P_i),\\ C\text{ — ограниченный факт}}} \mathcal{M}(\exists\overline{\ell}_P.body(C)).$$

Это доказывает базу индукции.

Переход. Пусть $C \in rules(P)$ и $C \equiv \varphi\wedge R_1(\overline{x}_1)\wedge\ldots\wedge R_m(\overline{x}_m) \to P(\overline{v}_P)$. По определению дерева вывода (C,ψ) является корнем некоторого дерево вывода высоты h тогда и только тогда, когда существуют такие деревья вывода с корнями $(D_1,\varphi_1),\ldots,(D_m,\varphi_m)$, имеющими высоту не более $h-1$, что $\psi \to head(C)$ является гиперрезольвентой дизъюнктов C и $\varphi_1 \to head(D_1),\ldots,\varphi_m \to head(D_m)$. По определению гиперрезольвенты имеем:

$$\psi \equiv \varphi \wedge \varphi_1[\overline{x}_1/\overline{v}_{R_1}] \wedge \ldots \wedge \varphi_m[\overline{x}_m/\overline{v}_{R_m}].$$

Пусть $\overline{y}_i$ — свободные переменные ψ_i, которые не входят в $\overline{v}_{R_i}$. При этом по оговорке, сделанной при определении гиперрезольвенты, можно считать, что экзистенциальные переменные в C и $\overline{y}_1,\ldots,\overline{y}_n$ попарно различны. Пусть $\overline{y}$ — это свободные переменные ψ, не входящие в $\overline{v}_P$. Тогда имеем следующее:

$$\exists\overline{y}.\psi \sim \exists\overline{\ell}_P.(\varphi \wedge \exists\overline{y}_1.\varphi_1[\overline{x}_1/\overline{v}_{R_1}] \wedge \ldots \wedge \exists\overline{y}_m.\varphi_m[\overline{x}_m/\overline{v}_{R_m}]). \tag{2.9}$$

Пусть k является местностью символа P. Определим оператор $U_C : \mathcal{P} \to 2^{|\mathcal{M}|^k}$ следующим образом:

$$U_C(\overline{X}) \stackrel{\text{def}}{=} \left(\mathcal{M},\overline{X}\right)(\exists\overline{\ell}_P.body(C)) = \left(\mathcal{M},\overline{X}\right)(\exists\overline{\ell}_P.(\varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m))).$$

Поскольку неинтерпретированные атомы входят в $body(C)$ позитивно, то оператор U_C монотонен. По индукционному предположению имеем $\mathcal{M}(\exists\overline{y}_i.\varphi_i) \subseteq [\![R_i]\!]^{h-1}_{\mathcal{M}}$, следовательно, по монотонности U_C и (2.9) выполнено следующее:

$$\mathcal{M}(\exists\overline{y}.\psi) \subseteq U_C\left(\left\langle [\![P_1]\!]^{h-1}_{\mathcal{M}}, \ldots, [\![P_n]\!]^{h-1}_{\mathcal{M}}\right\rangle\right).$$

Поскольку при фиксированном C все деревья вывода с корнями (C,ψ) высоты не более h получаются из всевозможных деревьев вывода R_i высоты не более $h-1$ (в объединении дающих $[\![R_i]\!]^{h-1}_{\mathcal{M}}$), а интерпретация символов $\mathcal{R}\setminus\{R_1,\ldots,R_m\}$ не влияет на истинность $body(C)$, то мы получаем следующее:

$$U_C\left(\left\langle [\![P_1]\!]^{h-1}_{\mathcal{M}}, \ldots, [\![P_n]\!]^{h-1}_{\mathcal{M}}\right\rangle\right) = \bigcup_{\substack{(C,\psi)\text{ — корень дерева вывода}\\ \text{высоты не более } h,\\ \overline{y}\text{ — все свободные}\\ \text{переменные }\psi\text{, кроме }\overline{v}_P.}} \mathcal{M}(\exists\overline{y}.\psi).$$

Более того, поскольку $body(P) = \bigvee_{C\in rules(P)} body(C)$, то для любого $\overline{X} \in \mathcal{P}$ имеем:

$$\bigcup_{C\in rules(P)} U_C(\overline{X}) = Y_i, \quad \text{где } i \text{ — номер } P \text{ в } \mathcal{R}, \tag{2.10}$$

$$T(\overline{X}) = \langle Y_1, \ldots, Y_n\rangle.$$

По определению ограниченной семантики имеем следующее:

$$T\left(\left\langle [\![P_1]\!]^{h-1}_{\mathcal{M}}, \ldots, [\![P_n]\!]^{h-1}_{\mathcal{M}}\right\rangle\right) = \left\langle [\![P_1]\!]^{h}_{\mathcal{M}}, \ldots, [\![P_n]\!]^{h}_{\mathcal{M}}\right\rangle.$$

В силу вышесказанного очевидно следующее утверждение:

$$[\![P]\!]^h_{\mathcal{M}} = \bigcup_{C\in rules(P)} U_C\left(\left\langle [\![P_1]\!]^{h-1}_{\mathcal{M}}, \ldots, [\![P_n]\!]^{h-1}_{\mathcal{M}}\right\rangle\right).$$

□

Поскольку $h \in \mathbb{N}$, то множество деревьев вывода высоты не более h конечно, и из теоремы 9 следует представимость всех $[\![P]\!]^h_{\mathcal{M}}$ в $\mathcal{M}$. Следовательно, дизъюнкция всех ограничений в корнях соответствующих деревьев вывода представляет $[\![P]\!]^h_{\mathcal{M}}$.

Теорема 9 является важной для понимания связи между синтаксической версией доказательства корректности синхронизации (теорема 4) и семантической версией, которая будет получена в разделе 2.5 (теорема 14). Из этой же

теоремы следует полнота исчисления гиперрезолюций для систем дизъюнктов Хорна с ограничениями (утверждение 3). Завершим доказательство последнего факта.

Теорема 10. Если система дизъюнктов Хорна с ограничениями невыполнима, то у неё существует гиперрезолютивное опровержение.

Доказательство. По теореме 8 существует такие модель $\mathcal{M}$ теории $\mathcal{T}$ и запрос $Q \equiv \varphi(\overline{y},\overline{x}_1,\ldots,\overline{x}_m) \wedge P_{i_1}(\overline{x}_1) \wedge \ldots \wedge P_{i_m}(\overline{x}_m) \to \bot$, что $(\llbracket P_{i_1} \rrbracket_{\mathcal{M}} \times \cdots \times \llbracket P_{i_m} \rrbracket_{\mathcal{M}}) \cap \mathcal{M}(\exists \overline{y}.\varphi) \neq \varnothing$.

Другими словами, существуют такие $\overline{a}_1 \in |\mathcal{M}|^{ar\left(P_{i_1}\right)}, \ldots, \overline{a}_n \in |\mathcal{M}|^{ar(P_{i_m})}$, что $\overline{a}_j \in \llbracket P_{i_j} \rrbracket_{\mathcal{M}}$ для всех $j \in \{1,\ldots,m\}$ и $\mathcal{M} \models \exists \overline{y}.\varphi(\overline{a}_1,\ldots,\overline{a}_m)$.

Но $\llbracket P_{i_j} \rrbracket_{\mathcal{M}} = \bigcup_{b \geqslant 0} \llbracket P_{i_j} \rrbracket^b_{\mathcal{M}}$, поэтому существуют такие $b_1,\ldots,b_m \in \mathbb{N}$ такие, что для всех $j \in \{1,\ldots,m\}$ выполнено $\overline{a}_j \in \llbracket P_{i_j} \rrbracket^{b_j}_{\mathcal{M}}$.

В свою очередь, по теореме 9 для каждого j существует дерево вывода с корнем (C_j,φ_j) и высотой не более b_j, такое, что $\mathcal{M} \models \exists \overline{y}_j.\varphi_j(\overline{a}_j)$, где $\overline{y}_j$ — все (кроме $\overline{v}_{P_{i_j}}$) свободные переменные φ_j. Поэтому выполнима и гиперрезольвента:

$$\psi \stackrel{\text{def}}{=} \varphi \wedge \bigwedge_{j=1}^{m} \varphi_j[\overline{x}_j/\overline{v}_{P_{i_j}}],$$

т.е. существует дерево вывода с корнем (Q,ψ) с выполнимым по модулю $\mathcal{T}$ ограничением ψ. □

2.5 Семантика синхронизации дизъюнктов Хорна

В данном разделе будет показана связь между отношениями переходов системы и её синхронизацией. В конце будет дана семантическая версия доказательства теоремы 4, не оперирующая гиперрезолютивными опровержениями.

Пусть $\mathcal{S}$ — система дизъюнктов Хорна с ограничениями над $\mathcal{R} = \{P_1,\ldots,P_n\}$, $\mathcal{S}'$ — синхронизация дизъюнкта $C \in \mathcal{S}$, т.е. $\mathcal{S}'$ является следую-

щей системой:

$$\mathcal{S}' \stackrel{\text{def}}{=} \mathcal{S} \setminus \{C\} \cup \{C'\} \cup \bigcup_{R \in N} rules(R), \text{ где}$$

$$C \equiv \varphi \wedge \bigwedge NRec\,(C) \wedge \bigwedge Rec\,(C) \to head(C)$$

$$NRec\,(C) = \{P_{i_1}(\overline{x}_1), \ldots, P_{i_m}(\overline{x}_m)\},$$

$$C' \stackrel{\text{def}}{=} \varphi \wedge \prod_{j=1}^{m} P_{i_j}(\overline{x}_j) \wedge \bigwedge Rec\,(C) \to head(C),$$

$$R \stackrel{\text{def}}{=} G(\langle P_{i_1}, \ldots, P_{i_m} \rangle).$$

При этом N — это наименьшее по включению множество символов, содержащее R и все символы из $\mathcal{R}'$, входящие в рекурсивные атомы правил символов из N.

Для простоты изложения будем считать, что $N = \{R\}$; всё, что представлено ниже, легко адаптировать для общего случая. В таком случае, в $\mathcal{S}'$ входят лишь неинтерпретированные символы из множества $\mathcal{R} \cup \{R\}$, где $ar\,(R) = ar\,(P_{i_1}) + \ldots + ar\,(P_{i_m})$.

Пусть $\mathcal{P} \stackrel{\text{def}}{=} 2^{|\mathcal{M}|^{ar(P_1)}} \times \cdots \times 2^{|\mathcal{M}|^{ar(P_n)}}$ и $\mathcal{P}' \stackrel{\text{def}}{=} \mathcal{P} \times 2^{|\mathcal{M}|^{ar(R)}}$.

Сформулируем и докажем два вспомогательных утверждения.

Лемма 5. Для любых неинтерпретированных атомов $R_1(\overline{y}_1), \ldots, R_m(\overline{y}_m)$, если все переменные в кортежах $\overline{y}_i$ и $\overline{y}_j$ попарно различны при $i \neq j$, то для любой интерпретации $\mathcal{M}$ и отношений $X_1 \subseteq |\mathcal{M}|^{ar(R_1)}, \ldots, X_m \subseteq |\mathcal{M}|^{ar(R_m)}$,

$$(\mathcal{M}\{R_1 \mapsto X_1\} \ldots \{R_m \mapsto X_m\})\,(R_1(\overline{y}_1) \wedge \cdots \wedge R_m(\overline{y}_m)) = X_1 \times \cdots \times X_m.$$

Доказательство. Утверждение теоремы следует из более общего: для любых двух формул $\varphi(x_1, \ldots, x_k)$ и $\psi(y_1, \ldots, y_\ell)$ и любой интерпретации $\mathcal{M}$, если $\mathcal{M}(\varphi) = X$, $\mathcal{M}(\psi) = Y$, и переменная x_i отлична от y_j для всех i и j, то $\mathcal{M}(\varphi(x_1, \ldots, x_k) \wedge \psi(y_1, \ldots, y_\ell)) = X \times Y$.

Это верно, т.к.

$$\varphi(x_1, \ldots, x_k) \wedge \psi(y_1, \ldots, y_\ell) \sim (\varphi(x_1, \ldots, x_k) \wedge y_1 = y_1 \wedge \cdots \wedge y_\ell = y_\ell) \wedge$$
$$\wedge\, (x_1 = x_1 \wedge \cdots \wedge x_k = x_k \wedge \psi(y_1, \ldots, y_\ell)),$$

а $\mathcal{M}(\varphi(x_1, \ldots, x_k) \wedge y_1 = y_1 \wedge \ldots \wedge y_\ell = y_\ell) = X \times |\mathcal{M}|^{\ell}$, и $\mathcal{M}(x_1 = x_1 \wedge \ldots \wedge x_k = x_k \wedge \psi(y_1, \ldots, y_\ell)) = |\mathcal{M}|^{k} \times Y$. Но для любых двух формул A и B с одинаковым набором свободных переменных и любой структуры $\mathcal{M}$ верно $\mathcal{M}(A \wedge B) = \mathcal{M}(A) \cap \mathcal{M}(B)$, а значит $\mathcal{M}(\varphi \wedge \psi) = \left(X \times |\mathcal{M}|^{\ell}\right) \cap \left(|\mathcal{M}|^{k} \times Y\right) = X \times Y$. $\square$

Лемма 6. Пусть $C_1 \in \mathit{rules}(P_{i_1}), \ldots, C_m \in \mathit{rules}(P_{i_m})$, причём все переменные в $C_1, \ldots, C_m$ попарно различны, $\mathcal{M}$ — интерпретация, $\langle X_1, \ldots, X_n \rangle \in \mathcal{P}$. Тогда

$$(\mathcal{M}, \langle X_1, \ldots, X_n \rangle)\,(\mathit{body}(C_1) \wedge \cdots \wedge \mathit{body}(C_m)) = \\ = (\mathcal{M}, \langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m} \rangle)\,(\mathit{body}(C_1 \times \cdots \times C_m)).$$

Доказательство. Без потери общности можем считать, что переменные всех атомов всех дизъюнктов $C_1, \ldots, C_m$ попарно различны, иначе переименуем их, добавив нужные равенство в ограничение. Так как в тело произведения дизъюнктов состоит из атомов перемножаемых дизъюнктов и их произведений, данная лемма является прямым следствием предыдущей. □

Пусть $T : \mathcal{P} \to \mathcal{P}$ — оператор отношения переходов системы $\mathcal{S}$, $T' : \mathcal{P}' \to \mathcal{P}'$ — оператор отношения переходов системы $\mathcal{S}'$ (см. раздел 2.4.1). Их связь отражена в следующей теореме.

Теорема 11. Для всех $\langle X_1, \ldots, X_n \rangle \in \mathcal{P}$, если $T(\langle X_1, \ldots, X_n \rangle) = \langle Y_1, \ldots, Y_n \rangle$, то

$$T'(\langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m} \rangle) = \langle Y_1, \ldots, Y_n, Y_{i_1} \times \cdots \times Y_{i_m} \rangle\,.$$

Доказательство. По определению T имеем следующее:

$$Y_i = (\mathcal{M}, \langle X_1, \ldots, X_n \rangle)\,(\exists \overline{\ell}_{P_i}.\mathit{body}(P_i)).$$

Положим $\langle Z_1, \ldots, Z_n, Z \rangle \stackrel{\text{def}}{=} T'(\langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m} \rangle)$. Если $C \notin \mathit{rules}(P_i)$, то правила символа P_i в $\mathcal{S}$ и в $\mathcal{S}'$ не отличаются, и следовательно, $Z_i = Y_i$. Если $C \in \mathit{rules}(P_i)$. Тогда воспользуемся утверждением из леммы 6

$$(\mathcal{M}, \langle X_1, \ldots, X_n \rangle) \left(\bigwedge_{j=1}^{m} P_{i_j}(\overline{y}_j) \right) = (\mathcal{M}, \langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m} \rangle) \left(\prod_{j=1}^{m} P_{i_j}(\overline{y}_j) \right).$$

Тогда имеем:

$$(\mathcal{M}, \langle X_1, \ldots, X_n \rangle)\,(\mathit{body}(C_i)) = (\mathcal{M}, \langle X_1, \ldots, X_n \rangle) \left(\varphi \wedge \bigwedge_{j=1}^{m} P_{i_j}(\overline{x}_j) \wedge \bigwedge \mathit{Rec}\,(C) \right) =$$

$$= (\mathcal{M}, \langle X_1, \ldots, X_n, X_1 \times \ldots \times X_n \rangle) \left(\varphi \wedge \prod_{j=1}^{m} P_{i_j}(\overline{x}_j) \wedge \bigwedge \mathit{Rec}\,(C) \right) =$$

$$= (\mathcal{M}, \langle X_1, \ldots, X_n, X_1 \times \ldots \times X_n \rangle)\,(\mathit{body}(C'))\,.$$

Следовательно, и в этом случае $Z_i = Y_i$. Наконец, в силу (2.2) получаем:

$$body(R) = \bigvee_{\langle C_1,\ldots,C_k\rangle \in rules(P_{i_1})\times\ldots\times rules(P_{i_m})} body(C_1 \times \cdots \times C_m).$$

По лемме (6) справедливо следующее:

$$\begin{aligned}
&(\mathcal{M}, \langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m}\rangle)\,(body(R)) = \\
&= (\mathcal{M}, \langle X_1, \ldots, X_n\rangle)\,(\bigvee_{\langle C_1,\ldots,C_k\rangle \in rules(P_{i_1})\times\cdots\times rules(P_{i_m})} body(C_1) \wedge \cdots \wedge body(C_m)) = \\
&= (\mathcal{M}, \langle X_1, \ldots, X_n\rangle) \left(\bigwedge_{j=1}^{m} \bigvee_{C \in rules(P_{i_j})} body(C) \right) = \\
&\qquad = (\mathcal{M}, \langle X_1, \ldots, X_n\rangle)\,(body(P_{i_1}) \wedge \cdots \wedge body(P_{i_m})).
\end{aligned}$$

Следовательно, $Z = Y_{i_1} \times \cdots \times Y_{i_m}$. □

Теперь всё готово для демонстрации важнейшего семантического свойства синхронизации: синхронизация *сохраняет произведение* интерпретаций.

Теорема 12. Для $\langle X_1, \ldots, X_n\rangle \in \mathcal{P}$ и $b \geqslant 1$, если выполнено $T^b(\langle X_1, \ldots, X_n\rangle) = \langle Y_1, \ldots, Y_n\rangle$, то выполнено и

$$T'^{\,b}(\langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m}\rangle) = \langle Y_1, \ldots, Y_n, Y_{i_1} \times \cdots \times Y_{i_m}\rangle\,.$$

Доказательство. Выполним доказательство теоремы индукцией по b. Для $b = 1$ утверждение верно по теореме 11. Пусть теперь $\langle Y_1, \ldots, Y_n\rangle \stackrel{\text{def}}{=} T^b(\langle X_1, \ldots, X_n\rangle)$, и $\langle Z_1, \ldots, Z_n\rangle \stackrel{\text{def}}{=} T^{b+1}(\langle X_1, \ldots, X_n\rangle) = T(\langle Y_1, \ldots, Y_n\rangle)$. По индукционному предположению, $T'^{\,b}(\langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m}\rangle) = \langle Y_1, \ldots, Y_n, Y_{i_1} \times \cdots \times Y_{i_m}\rangle$. Тогда

$$T'^{\,b+1}(\langle X_1, \ldots, X_n, X_{i_1} \times \cdots \times X_{i_m}\rangle) = T'(\langle Y_1, \ldots, Y_n, Y_{i_1} \times \cdots \times Y_{i_m}\rangle).$$

Поскольку $T(\langle Y_1, \ldots, Y_n\rangle) = \langle Z_1, \ldots, Z_n\rangle$, то по теореме 11 имеем $T'(\langle Y_1, \ldots, Y_n, Y_{i_1} \times \cdots \times Y_{i_m}\rangle) = \langle Z_1, \ldots, Z_n, Z_1 \times \cdots \times Z_n\rangle$. □

Теорема 11 позволяет говорить об ограниченных семантиках символа P_i без уточнения, идёт ли речь о правилах системы $\mathcal{S}$ или системы $\mathcal{S}'$. Действительно, по определению ограниченной семантики,

$$\left\langle [\![P_1]\!]^b_{\mathcal{M}}, \ldots, [\![P_n]\!]^b_{\mathcal{M}} \right\rangle \stackrel{\text{def}}{=} T^{b+1}(\overline{\varnothing}).$$

Но по теореме 12 имеем следующее:

$$T'^{\,b+1}(\overline{\varnothing}) = \left\langle [\![P_1]\!]^b_{\mathcal{M}}, \ldots, [\![P_n]\!]^b_{\mathcal{M}}, [\![P_{i_1}]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_{i_m}]\!]^b_{\mathcal{M}} \right\rangle .$$

В частности, из этого следует корректный способ задания ограниченной семантики произведения символов, сохраняющий все полученные ранее результаты об ограниченных семантиках для синхронизаций систем:

$$[\![G(\langle R_1, \ldots, R_m \rangle)]\!]^b_{\mathcal{M}} \stackrel{\text{def}}{=} [\![R_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![R_m]\!]^b_{\mathcal{M}} . \tag{2.11}$$

Завершим исследование семантики произведения дизъюнктов, показав, что семантика произведения символов является произведениями семантик символов.

Теорема 13. $\langle [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}} \rangle$ — наименьшая неподвижная точка T тогда и только тогда, когда $\langle [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}}, [\![P_{i_1}]\!]_{\mathcal{M}} \times \cdots \times [\![P_{i_n}]\!]_{\mathcal{M}} \rangle$ — наименьшая неподвижная точка T'.

Доказательство. Наименьшей неподвижной точкой оператора T является

$$\langle [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}} \rangle = \bigcup_{b \geqslant 0} T^b(\overline{\varnothing}) = \left\langle \bigcup_{b \geqslant 0} [\![P_1]\!]^b_{\mathcal{M}}, \ldots, \bigcup_{b \geqslant 0} [\![P_n]\!]^b_{\mathcal{M}} \right\rangle .$$

Вместе с тем, наименьшей неподвижной точкой оператора T' является

$$\langle [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}}, [\![R]\!]_{\mathcal{M}} \rangle = \bigcup_{b \geqslant 0} T'^{\,b}(\overline{\varnothing}) =$$

$$= \left\langle \bigcup_{b \geqslant 0} [\![P_1]\!]^b_{\mathcal{M}}, \ldots, \bigcup_{b \geqslant 0} [\![P_n]\!]^b_{\mathcal{M}}, \bigcup_{b \geqslant 0} \left([\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^b_{\mathcal{M}} \right) \right\rangle .$$

По определению частичного порядка на $\mathcal{P}$ имеем следующее:

$$\bigcup_{b \geqslant 0} \left([\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^b_{\mathcal{M}} \right) \subseteq \bigcup_{b \geqslant 0} [\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times \bigcup_{b \geqslant 0} [\![P_n]\!]^b_{\mathcal{M}} .$$

Как показано в разделе 2.4.2, для $i \in \{1, \ldots, n\}$ и $b \geqslant 0$ имеем:

$$[\![P_i]\!]^b_{\mathcal{M}} \subseteq [\![P_i]\!]^{b+1}_{\mathcal{M}} .$$

Тогда для $b_1, \ldots, b_n \geqslant 0$ и $b \stackrel{\text{def}}{=} \max_{i=1}^{n} b_i$ имеем следующее:

$$[\![P_1]\!]^{b_1}_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^{b_n}_{\mathcal{M}} \subseteq [\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^b_{\mathcal{M}} .$$

Теперь возьмём $\langle x_1,\ldots,x_n\rangle \in \bigcup_{b\geqslant 0}[\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times \bigcup_{b\geqslant 0}[\![P_n]\!]^b_{\mathcal{M}}$. Существуют $b_1,\ldots,b_n$ такие, что

$$x_1 \in [\![P_1]\!]^{b_1}_{\mathcal{M}}, \ldots, x_n \in [\![P_n]\!]^{b_n}_{\mathcal{M}}.$$

Следовательно, для $b = \max_{i=1} b_i$ имеем:

$$\langle x_1,\ldots,x_n\rangle \in [\![P_1]\!]^b_{\mathcal{M}} \times \ldots \times [\![P_n]\!]^b_{\mathcal{M}}.$$

Из этого можно заключить, что

$$\bigcup_{b\geqslant 0}[\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times \bigcup_{b\geqslant 0}[\![P_n]\!]^b_{\mathcal{M}} \subseteq \bigcup_{b\geqslant 0}[\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^b_{\mathcal{M}}.$$

Таким образом имеем следующее:

$$[\![R]\!]_{\mathcal{M}} = \bigcup_{b\geqslant 0}[\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^b_{\mathcal{M}} = \bigcup_{b\geqslant 0}[\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times \bigcup_{b\geqslant 0}[\![P_n]\!]^b_{\mathcal{M}} = [\![P_1]\!]_{\mathcal{M}} \times \cdots \times [\![P_n]\!]_{\mathcal{M}}.$$

□

Завершим данный раздел доказательством семантической версии теоремы 4.

Теорема 14. Система $\mathcal{S}$ выполнима тогда и только тогда, когда выполнима система $\mathcal{S}'$.

Доказательство. По теореме 8 система $\mathcal{S}$ выполнима тогда и только тогда, когда для всякой модели $\mathcal{M}$ теории $\mathcal{T}$ и всякого запроса $Q \equiv \varphi(\overline{y},\overline{z}_1,\ldots,\overline{z}_m) \wedge P_{j_1}(\overline{z}_1) \wedge \ldots \wedge P_{j_k}(\overline{z}_k) \to \bot$ справедливо следующее:

$$[\![P_{j_1}]\!]_{\mathcal{M}} \times \cdots \times [\![P_{j_k}]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\forall \overline{y}.\neg\varphi). \tag{2.12}$$

Если $Q \neq C$, то $Q \in \mathcal{S}'$, но поскольку, в силу теоремы 13, семантика символов $P_{j_1},\ldots,P_{j_k}$ в $\mathcal{S}'$ не изменились, то (2.12) справедливо и для системы $\mathcal{S}'$.

Если $Q = C$, то в системе $\mathcal{S}'$ дизъюнкт Q заменён следующим дизъюнктом:

$$Q' \equiv \varphi(\overline{y},\overline{z}_1,\ldots,\overline{z}_m) \wedge \prod_{i=1}^{k} P_{j_i}(\overline{z}_i) \to \bot.$$

Поэтому условие теоремы 8 для Q' можно представить так:

$$[\![G(\langle P_{j_1},\ldots,P_{j_k}\rangle)]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\forall \overline{y}.\neg\varphi). \tag{2.13}$$

По теореме 13 имеем $[\![G(\langle P_{j_1},\ldots,P_{j_k}\rangle)]\!]_{\mathcal{M}} = [\![P_{j_1}]\!]_{\mathcal{M}} \times \cdots \times [\![P_{j_k}]\!]_{\mathcal{M}}$. Из этого следует равносильность утверждений (2.13) и (2.12). □

2.6 Непредставимость сертификатов выполнимости некоторых систем

Вернёмся к системе $\mathcal{S}$ из примера 7:

$$x=0 \wedge z=0 \to mul(x,y,z)$$
$$x>0 \wedge x'=x-1 \wedge z=z'+y \wedge mul(x',y,z') \to mul(x,y,z)$$
$$x=x' \wedge y=y' \wedge z\neq z' \wedge mul(x,y,z) \wedge mul(x',y',z') \to \bot$$

Как было показано, эта система выполнима. Семантика неподвижной точки даёт простой способ доказательства непредставимости моделей этой системы в линейной целочисленной арифметике.

Утверждение 7. Система $\mathcal{S}$ выполнима, но у неё не существует сертификата выполнимости.

Доказательство. Класс моделей $\mathcal{T}_{LIA}$ состоит из единственной модели $\mathcal{N}$. Отношение переходов системы $\mathcal{S}$ в интерпретации $\mathcal{N}$ является оператор $T : 2^{\mathbb{Z}^3} \to 2^{\mathbb{Z}^3}$. Для этого оператора справедливо следующее:

$$T(A) = \{\langle 0,y,0\rangle \mid y \in \mathbb{Z}\} \cup \{\langle x+1,y,z+y\rangle \mid \langle x,y,z\rangle \in A\}.$$

Как показано в разделе 2.4.2, наименьшая преднеподвижная точка этого оператора равна $[\![mul]\!]_{\mathcal{N}} = \bigcup_{i\geqslant 0} T^i(\varnothing)$. Вычислим её:

$$\begin{aligned}
&T^0(\varnothing) = \varnothing\\
&T^1(\varnothing) = \{\langle 0,y,0\rangle \mid y \in \mathbb{Z}\}\\
&T^2(\varnothing) = T(T^1(\varnothing)) = \{\langle 1,y,y\rangle \mid y \in \mathbb{Z}\}\\
&T^3(\varnothing) = T(T^2(\varnothing)) = \{\langle 2,y,2y\rangle \mid y \in \mathbb{Z}\}\\
&T^4(\varnothing) = T(T^3(\varnothing)) = \{\langle 3,y,3y\rangle \mid y \in \mathbb{Z}\}\\
&\quad\ldots\\
&T^k(\varnothing) = \{\langle k-1,y,(k-1)y\rangle \mid y \in \mathbb{Z}\}\\
&\quad\ldots
\end{aligned}$$

Тогда очевидно следующее:

$$[\![mul]\!]_{\mathcal{N}} = \bigcup_{i\geqslant 0} T^i(\varnothing) = \{\langle x,y,z\rangle \in \mathbb{Z}^3 \mid x \geqslant 0, z = x\cdot y\}.$$

Теперь предположим, что $\mathcal{N}\{mul \mapsto X\} \models \bigwedge_{C \in \mathcal{S}} \forall C$. Нужно показать, что отношение X не представимо в линейной целочисленной арифметике.

Разобьем множество X на два подмножества X^+ и X^-, заданные следующим образом:

$$X^+ \stackrel{\text{def}}{=} \{\langle x,y,z\rangle \in X \mid x \geqslant 0\}; \qquad X^- \stackrel{\text{def}}{=} \{\langle x,y,z\rangle \in X \mid x < 0\}.$$

Очевидно, что $X^+ \cap X^- = \varnothing$. Докажем, что $X^+ = [\![mul]\!]_{\mathcal{N}}$. По теореме 7 множество X является преднеподвижной точкой оператора T, и для него справедливо следующее:

$$X \times X \subseteq \{\langle x,y,z,x',y',z'\rangle \in \mathbb{Z}^6 \mid x \neq x' \text{ или } y \neq y' \text{ или } z = z'\}. \tag{2.14}$$

Но поскольку $[\![mul]\!]_{\mathcal{N}}$ — наименьшая преднеподвижная точка, то выполнено $[\![mul]\!]_{\mathcal{N}} \subseteq X$, а поскольку $[\![mul]\!]_{\mathcal{N}} \cap X^- = \varnothing$, то выполнено $[\![mul]\!]_{\mathcal{N}} \subseteq X^+$. Пусть $\langle x,y,z_1\rangle \in X^+$ и $\langle x,y,z_2\rangle \in X^+$. Тогда $\langle x,y,z_1,x,y,z_2\rangle \in X \times X$, и в силу (2.14) $z_1 = z_2$, т.е. для всех $x \geqslant 0$ и $y \in \mathbb{Z}$ существует не более одного $z \in \mathbb{Z}$, что $\langle x,y,z\rangle \in X^+$. Но $\{\langle x,y,z\rangle \in \mathbb{Z}^3 \mid x \geqslant 0, z = x \cdot y\} \subseteq X^+$, следовательно, $X^+ = [\![mul]\!]_{\mathcal{N}}$.

Вне зависимости от X^- отношение $X = X^- \cup X^+$ непредставимо в $\mathcal{N}$.

Докажем это от противного. Пусть формула $\varphi(x,y,z)$ представляет отношение X. Тогда формула $x \geqslant 0 \wedge \varphi(x,y,z)$ представляет X^+, а формула $(x \geqslant 0 \wedge \varphi(x,y,z)) \vee (x < 0 \wedge \varphi(-x,y,-z))$ задаёт отношение умножения $\{\langle x,y,z\rangle \in \mathbb{Z}^3 \mid z = x \cdot y\}$, которое не представимо в линейной целочисленной арифметике. □

Пример 7 и утверждение 7 можно обобщить следующим образом. Возьмём любую частично вычислимую функцию $f : \mathbb{Z}^k \to \mathbb{Z}$, которая не представима в линейной целочисленной арифметике, но область определения которой представима формулой $\varphi(x_1,\ldots,x_k)$. Очевидно, для неё верно $x_1 = x'_1 \wedge \cdots \wedge x_k = x'_k \to f(x_1,\ldots,x_k) = f(x'_1,\ldots,x'_k)$. Запишем условия верификации этого утверждения в виде следующей системы дизъюнктов Хорна с ограничениями:

$$\cdots \to P_f(x_1,\ldots,x_k,z)$$
$$x_1 = x'_1 \wedge \cdots \wedge x_k = x'_k \wedge z = z' \wedge P_f(x_1,\ldots,x_k,z) \wedge P_f(x'_1,\ldots,x'_k,z') \to \bot$$

Сужение любой модели этой системы на область определения функции f есть график этой функции[2], следовательно, из представимости этой модели формулой $\psi(x_1,\ldots,x_k,z)$ будет следовать представимость графика f формулой $\varphi(x_1,\ldots,x_k) \wedge \psi(x_1,\ldots,x_k,z)$, но этого не может быть. Используя этот метод, можно порождать бесконечное число примеров нелинейных систем дизъюнктов, модели которых не представимы в линейной целочисленной арифметике.

Более фундаментальное описание причин того, что модели таких систем не представимы, приведено в главе 3.

2.7 Алгоритм CHCprodуct

Целесообразность синхронизирующего преобразования можно аргументировать следующим образом: синхронизация позволяет "уменьшить нелинейность" системы, заменив несколько нерекурсивных атомов одним с помощью введения новых неинтерпретированных символов. Однако это верно лишь для одного, синхронизируемого, дизъюнкта в системе. Вместе с тем, в правилах нововведённых неинтерпретированных символов может оказаться ещё больше нерекурсивных атомов, так как эти правила являются произведениями дизъюнктов синхронизируемой системы.

В данном разделе представлен алгоритм, который итеративно выполняет синхронизации нужных дизъюнктов, выдавая на выходе систему $\mathcal{S}'$, для которой верно следующее утверждение:

$$\text{для всех } C \in \mathcal{S}' \, |NRec\,(C)| \leqslant 1.$$

Трассировка алгоритма на примере представлена в разделе 2.7.2. В разделе 2.7.3 доказывается корректность алгоритма (теорема 4), а также анализируется его завершаемость и сложность (теоремы 17 и 20).

2.7.1 Алгоритм CHCprodуct

В листинге 1 приведён псевдокод процедуры синхронизации под названием CHCprodуct. Она принимает на вход систему дизъюнктов $\mathcal{S}$ и выдаёт синхронизированную систему $\mathcal{S}'$. Алгоритм поддерживает очередь дизъюнктов Q и инициализирует Q запросами системы (строка 2). На каждой итерации ал-

[2]Графиком частичной функции $f : X \to Y$ называется отношение $\{\langle x, f(x)\rangle \mid x \in dom(f)\}$.

Листинг 1: Алгоритм CHCproduct

Вход : Система дизъюнктов $\mathcal{S}$

Выход : Система дизъюнктов $\mathcal{S}'$

Данные: Очередь дизъюнктов Q, множество посещённых символов V

```
1  S' := ∅; V := ∅;
2  Q := queries;
3  пока Q ≠ ∅
4      выбрать C ∈ Q; Q := Q \ {C};
5      {R_1(x̄_1), ..., R_m(x̄_m)} := NRec(C);
6      если m = 0 то
7          S' := S' ∪ {C};
8          перейти на строку 3;
9      (φ ∧ ⋀ Rec(C) ∧ ⋀ NRec(C) → head(C)) := C;
10     R := G(⟨R_1, ..., R_m⟩);
11     C' := (φ ∧ R(x̄_1, ..., x̄_m) ∧ Rec(C) → head(C));
12     S' := S' ∪ {C'};
13     N := наименьшее по включению подмножество R', содержащее R
           и все рекурсивные символы всех правил символов из N;
14     для всех P ∈ N
15         если P ∉ V то
16             Q := Q ∪ rules(P);
17             V := V ∪ {P};
```

горитм вынимает из Q первый в порядке очереди дизъюнкт из Q (строка 4), модифицирует его (строка 11) и добавляет его в $\mathcal{S}'$ (строка 12).

Для каждого дизъюнкта C алгоритм определяет нерекурсивные атомы $\langle R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)\rangle$ для тела C (строка 5) и заменяет их произведением (строка 11).

Правила, которые алгоритм добавляет в Q в строке 16, строятся по определению (2.2) (очевидно, что существует алгоритм, вычисляющий произведение m любых дизъюнктов за линейное от их размера время).

Листинг 2: Вычисление N в алгоритме CHCprodуct

```
1 N := {R}; Δ := {R};
2 пока Δ ≠ ∅
3 |  Δ' := ∅;
4 |  для всех R' ∈ Δ, D ∈ rules(R'), P(ȳ) ∈ Rec(D)
5 |  |  если P ∉ N' то
6 |  |  |  N := N ∪ P; Δ' := Δ' ∪ P;
7 |  Δ := Δ';
```

Множество символов N, вычисляемое в строке 13, вычисляется процедурой в листинге 2. Можно очевидным образом интегрировать псевдокод листинга 2 в листинг 1, избежав явного построения множества N. Однако представленная здесь версия проще для доказательства корректности, хотя и является менее эффективной.

Важно, что тела новых правил, добавленные алгоритмом в Q (строка 16), могут содержать множественные неинтерпретированные атомы других неинтерпретированных символов, которые также могут быть синхронизированы. Алгоритм рассматривает их на следующих итерациях.

Заметим, что любой дизъюнкт в системе $\mathcal{S}'$, возвращаемой алгоритмом, добавляется туда в либо в строке 7, либо в строке 12. В первом случае $|NRec(C)| = m = 0$, для второго справедливо $|NRec(C')| = 1$. Следовательно, верно, что для всех $C \in \mathcal{S}' \, |NRec(C)| \leqslant 1$.

2.7.2 Пример работы алгоритма CHCprodуct

Рассмотрим систему дизъюнктов Хорна, описывающие условия верификации рекурсивного вычисления n^n и $n!$ для $n > 1$.

$$
\begin{aligned}
a = 0 \wedge c = 0 &\to mul(a,b,c)\\
a > 0 \wedge a' = a-1 \wedge c = b + c' \wedge mul(a',b,c') &\to mul(a,b,c)\\
n = 0 \wedge m = 1 &\to fact(n,m)\\
n > 0 \wedge n' = n-1 \wedge fact(n',m') \wedge mul(m',n,m) &\to fact(n,m)\\
x = 0 \wedge z = 1 &\to pwr(x,y,z)\\
x > 0 \wedge x' = x-1 \wedge pwr(x',y,z') \wedge mul(z',y,z) &\to pwr(x,y,z)\\
fact(n,a) \wedge pwr(n,n,b) \wedge a > b &\to \bot
\end{aligned}
$$

В этой системе используются три неинтерпретированных символа mul, $fact$ и pwr для задания условий верификации рекурсивного вычисления умножения, факториала и возведения в степень. Дизъюнкт-запрос задаёт свойство этих вычислений: система выполнима, если и только если для всех натуральных $n > 1$ верно $\neg(n! \geqslant n^n)$. По причинам, разъяснённым в разделе 2.6, у этой системы не существует представимого в линейной целочисленной арифметике сертификата выполнимости. Алгоритм CHCprodUCT преобразует эту систему таким образом, что у системы появляется сертификат выполнимости.

Алгоритм CHCprodUCT начинает работу, помещая единственный запрос в очередь Q (строка 2). В строке 5 алгоритм получает список нерекурсивных атомов $\{fact(n,x), pwr(n,n,y)\}$ в теле запроса. Пусть $fp \stackrel{\text{def}}{=} G(\langle fact, pwr\rangle)$. В строках 11 и 12 в преобразованную систему добавляется следующий новый запрос:

$$fp(n,a,n,n,b) \wedge n > 1 \wedge a \geqslant b \to \bot.$$

Процедура в листинге 2 возвращает $N = \{fp\}$. При этом в очередь Q добавляются дизъюнкты-правила нововведённого символа fp (см. (2.2)). Таким образом, после первой итерации очередь Q содержит следующие дизъюнкты:

$$
\begin{aligned}
n = 0 \wedge m = 1 \wedge x = 0 \wedge z = 1 &\rightarrow fp(n,m,x,y,z) \\
n > 0 \wedge n' = n-1 \wedge x = 0 \wedge z = 1 &\wedge \\
\wedge fp(n',m',x,y,z) \wedge mul(m',n,m) &\rightarrow fp(n,m,x,y,z) \\
n = 0 \wedge m = 1 \wedge x > 0 \wedge x' = x-1 &\wedge \\
\wedge fp(n,m,x',y,z') \wedge mul(z',y,z) &\rightarrow fp(n,m,x,y,z) \\
n > 0 \wedge n' = n-1 \wedge m > 0 \wedge m' = m-1 &\wedge \\
\wedge fp(n',m',x',y,z') \wedge mul(m',n,m) \wedge mul(z',y,z) &\rightarrow fp(n,m,x,y,z)
\end{aligned}
$$

На следующих итерациях алгоритм CHCPRODUCT обрабатывает все дизъюнкты в Q, включая следующий:

$$
\begin{aligned}
n > 0 \wedge n' = n-1 \wedge m > 0 \wedge m' = m-1 &\wedge \\
\wedge fp(n',m',x',y,z') \wedge mul(m',n,m) \wedge mul(z',y,z) &\rightarrow fp(n,m,x,y,z)
\end{aligned}
$$

Множество $\{mul(a,n,x'),mul(b,y,y')\}$ является нерекурсивными атомами этого дизъюнкта, и оно строится в строке 5. Алгоритм вводит символ $mul^2 \stackrel{\text{def}}{=} G(\langle mul,mul\rangle)$, получает $N = \{mul^2\}$ и добавляет в очередь Q правила для mul^2. В конце своей работы алгоритм возвращает следующую систему $\mathcal{S}'$:

$$
\begin{aligned}
a = 0 \wedge c = 0 &\to mul(a,b,c)\\
mul(a',b,c') \wedge a > 0 \wedge a' = a-1 \wedge c = c' + b &\to mul(a,b,c)\\
a_1 = 0 \wedge c_1 = 0 \wedge a_2 = 0 \wedge c_2 = 0 &\to mul^2(a_1,b_1,c_1,a_2,b_2,c_2)\\
a_1 > 0 \wedge a_1' = a_1-1 \wedge c_1 = b_1 + c_1' \wedge & \\
\wedge a_2 = 0 \wedge c_2 = 0 \wedge mul^2(a_1',b_1,c_1',a_2,b_2,c_2) &\to mul^2(a_1,b_1,c_1,a_2,b_2,c_2)\\
a_2 > 0 \wedge a_2' = a_2-1 \wedge c_2 = b_2 + c_2' \wedge & \\
\wedge a_1 = 0 \wedge c_1 = 0 \wedge mul^2(a_1,b_1,c_1,a_2',b_2,c_2') &\to mul^2(a_1,b_1,c_1,a_2,b_2,c_2)\\
a_1 > 0 \wedge a_1' = a_1-1 \wedge c_1 = b_1 + c_1' \wedge a_2 > 0 \wedge & \\
\wedge a_2' = a_2-1 \wedge c_2 = b_2 + c_2' \wedge mul^2(a_1',b_1,c_1',a_2',b_2,c_2') &\to mul^2(a_1,b_1,c_1,a_2,b_2,c_2)\\
n = 0 \wedge m = 1 \wedge x = 0 \wedge z = 1 &\to fp(n,m,x,y,z)\\
n > 0 \wedge n' = n-1 \wedge x = 0 \wedge z = 1 \wedge & \\
\wedge fp(n',m',x,y,z) \wedge mul(m',n,m) &\to fp(n,m,x,y,z)\\
n = 0 \wedge m = 1 \wedge x > 0 \wedge x' = x - 1 \wedge & \\
\wedge fp(n,m,x',y,z') \wedge mul(z',y,z) &\to fp(n,m,x,y,z)\\
n > 0 \wedge n' = n-1 \wedge x > 0 \wedge x' = x-1 \wedge & \\
\wedge fp(n',m',x',y,z') \wedge mul^2(m',n,m,z',y,z) &\to fp(n,m,x,y,z)\\
fp(n,a,n,n,b) \wedge a > b &\to \bot
\end{aligned}
$$

Заметим, что в этой системе полностью отсутствуют неинтерпретированные символы $fact$ и pwr, входившие в изначальную систему. Таким образом, алгоритм обладает практически важной особенностью: он не добавляет «ненужные» правила изначальной системы, тем самым уменьшая размер результирующей системы.

Также в теле каждого дизъюнкта системы $\mathcal{S}'$ содержится не более одного нерекурсивного атома, в частности, дизъюнкт-запрос стал линейным (очевидно, это будет происходить для всех дизъюнктов-запросов). У этой системы существует сертификат выполнимости в линейной целочисленной арифметике:

$$\mathcal{I} \equiv \{fp \mapsto n = x \wedge n \leqslant y \to m \leqslant z,$$
$$mul^2 \mapsto a_1 \leqslant a_2 \wedge 0 < b_1 \leqslant b_2 \to c_1 \leqslant c_2,$$
$$mul \mapsto \top\}.$$

2.7.3 Анализ алгоритма CHCprodUCT

В данном разделе с помощью $\mathcal{S}$ будем обозначать произвольную систему дизъюнктов Хорна, а с помощью $\mathcal{S}'$ — систему, построенную алгоритмом CHCprodUCT по системе $\mathcal{S}$.

Для начала заметим, что конкретный порядок выбора очередного дизъюнкта из Q не определён (строка 4). Однако очевидно, что результат работы алгоритма не зависит от порядка выбора очередного дизъюнкта: этот порядок влияет только на порядок добавления дизъюнктов в $\mathcal{S}'$ (строка 12), но поскольку $\mathcal{S}'$ — это множество, то порядок добавления элементов в множество не имеет значения.

Напомним, что область определения отображения G — это $\mathbb{N}^{\mathcal{R}}$. Поэтому требуется показать, что в строке 10 нашего алгоритма G применяется только к кортежу символов из $\mathcal{R}$. Это следует из следующей леммы.

Лемма 7. Для любого дизъюнкта D, добавляемого алгоритмом CHCprodUCT в очередь Q, верно, что для всех $R(\overline{x}) \in NRec\,(D)$ выполнено $R \in \mathcal{R}$.

Доказательство. Любой дизъюнкт добавляется в Q либо в строке 2, либо в строке 16. В первом случае добавляются запросы системы $\mathcal{S}$, для которых утверждение леммы тривиально выполнено. Во втором случае в Q добавляются правила некоторого символа $P \in \mathcal{R}'$. В силу (2.2), любой добавляемый в Q дизъюнкт D представим следующим образом:

$$D = C_1 \times \cdots \times C_m \text{ для некоторых } C_1, \ldots, C_m \in \mathcal{S}.$$

Но по определению 14 произведения дизъюнктов имеем:

$$NRec\,(D) = \bigcup_{i=1}^{m} NRec\,(C_i)\,.$$

Из последнего факта утверждение леммы следует очевидным образом. □

Докажем ещё одно вспомогательное утверждение. Напомним, что *граф зависимостей* системы $\mathcal{S}$ — это ориентированный граф $\langle \mathcal{R}, E \rangle$, где $\langle P,R \rangle \in E$ тогда и только тогда, когда символ R появляется в теле некоторого правила символа P системы $\mathcal{S}$.

Лемма 8. Пусть $\mathcal{S}$ — произвольная система дизъюнктов Хорна, $\mathcal{R}_{queries}$ — это множество неинтерпретированных символов из $\mathcal{R}$, входящих в запросы системы $\mathcal{S}$, и $\mathcal{R}_{reach}$ — множество символов из $\mathcal{R}$, достижимых из вершин $\mathcal{R}_{queries}$ в графе зависимостей $\langle \mathcal{R}, E \rangle$ системы $\mathcal{S}$. Система $\mathcal{S}$ выполнима тогда и только тогда, когда выполнима система

$$\mathcal{S}_{reach} \stackrel{\text{def}}{=} queries \cup \bigcup_{P \in \mathcal{R}_{reach}} rules(P).$$

Доказательство. Воспользуемся теоремой о полноте исчисления гиперрезолюций: система $\mathcal{S}$ невыполнима тогда и только тогда, когда у неё существует гиперрезолютивное опровержение. Но любое гиперрезолютивное опровержение системы $\mathcal{S}$ является также и опровержением системы $\mathcal{S}_{reach}$, поскольку некоторый узел (C,ψ) может быть родителем узла (C',ψ') при условии, что $C' \in rules(R)$, если и только если $C \in queries$ и $R \in \mathcal{R}_{queries}$, либо $C \in rules(P)$ и $\langle P,R \rangle \in E$, поскольку R входит в тело C. □

Теперь всё готово для того, чтобы показать, что CHCPRODUCT преобразует входную систему дизъюнктов Хорна в эквивыполнимую.

Теорема 15. Система $\mathcal{S}'$, возвращаемая алгоритмом CHCPRODUCT, выполнима тогда и только тогда, когда выполнима $\mathcal{S}$.

Доказательство. Рассмотрим итерацию i алгоритма. Обозначим посредством $\mathcal{S}'_i, Q_i, \mathcal{S}'_{i+1}, Q_{i+1}$ значения переменных $\mathcal{S}'$ и Q в начале и в конце итерации i соответственно. И пусть C и C' — дизъюнкты, получаемые в строках 4 и 11.

По определению 15 верно следующее утверждение:

$$\mathcal{S}'_{i+1} \cup Q_{i+1} \cup \bigcup_{P \in \mathcal{R}} rules(P)$$

является синхронизацией дизъюнкта C системы

$$\mathcal{S}'_i \cup Q_i \cup \bigcup_{P \in \mathcal{R}} rules(P).$$

В этом не сложно убедиться, поскольку один шаг алгоритма удаляет из Q дизъюнкт C (строка 4), добавляет в $\mathcal{S}'$ дизъюнкт C' с заменёнными нерекурсивными атомами (строка 12), а также добавляет в Q правила для символов из множества N, в точности совпадающим с аналогичным множеством из определения 15 (строки 13 и 16).

В начале первой итерации $Q_1 = \mathit{queries}$ и $\mathcal{S}'_1 = \varnothing$, т.е. $\mathcal{S}'_1 \cup Q_1 \cup \bigcup_{P \in \mathcal{R}} \mathit{rules}(P) = \mathcal{S}$. В конце последней итерации (пусть она имеет номер $F-1$) $Q_F = \varnothing$, а $\mathcal{S}'_F$ возвращается как результат $\mathcal{S}'$, т.е. $\mathcal{S}'_F \cup Q_F \cup \bigcup_{R \in \mathcal{R}} \mathit{rules}(P) = \mathcal{S}' \cup \bigcup_{R \in \mathcal{R}} \mathit{rules}(P)$. Согласно теореме 4 система и любая её синхронизация эквивыполнимы. Поэтому очевидной индукцией по количеству итераций алгоритма CHCPRODUCT[3] легко показать, что система $\mathcal{S}_{reach}$ выполнима тогда и только тогда, когда выполнима система $\mathcal{S}'' \stackrel{\text{def}}{=} \mathcal{S}' \cup \bigcup_{R \in \mathcal{R}} \mathit{rules}(P)$. Рассмотрим зависимостей $\langle \mathcal{R}', E' \rangle$ системы $\mathcal{S}'$. По его построению очевидно, что пользуясь обозначениями леммы 8, мы имеем $\mathcal{S}'_{reach} = \mathcal{S}'$: алгоритм добавляет в Q правило любого символа, который появляется в теле любого дизъюнкта, добавляемого в $\mathcal{S}'$, и только такое правило. Более того, поскольку $\mathcal{R} \subseteq \mathcal{R}'$, имеем $\mathcal{S}''_{reach} = \mathcal{S}'$. По лемме 8, применённой к системе $\mathcal{S}''$, системы $\mathcal{S}''$ и $\mathcal{S}'$ эквивыполнимы. □

В доказательстве этой теоремы можно заменить применение теоремы 4 на применение теоремы 5, и тем самым доказать следующее утверждение.

Теорема 16. Если у системы $\mathcal{S}$ существует сертификат выполнимости, то он существует также и у системы $\mathcal{S}'$.

Теперь докажем завершаемость алгоритма и оценим размер системы $\mathcal{S}'$.

Определение 19. Пусть $\mathcal{D} \stackrel{\text{def}}{=} \langle \mathcal{R}, E \rangle$ — граф зависимостей $\mathcal{D}$ системы дизъюнктов Хорна с ограничениями $\mathcal{S}$. *Рангом* символа $P \in \mathcal{R}$ назовём натуральное число $rank\,(P)$ такое, что справедливо следующее:

- если K — это количество сильно связных компонент графа $\mathcal{D}$, то тогда $1 \leqslant rank\,(P) \leqslant K$;
- для $P, P' \in \mathcal{R}$, $rank\,(P) \leqslant rank\,(P')$ тогда и только тогда, когда в графе $\mathcal{D}$ вершина P' достижима из вершины P;
- для P,$P' \in \mathcal{R}$, $rank\,(P) = rank\,(P')$ тогда и только тогда, когда P и P' принадлежат одной и той же сильно связной компоненте графа $\mathcal{D}$, т.е. P и P' рекурсивны.

[3]Итерациями нашего алгоритма будем называть итерации цикла `while` в строке 3.

Неформально говоря, ранг вершины графа зависимостей задаёт линейный порядок на его сильно связных компонентах графа. Его можно получить, например, топологической сортировкой конденсации графа $\mathcal{D}$.

К примеру, для системы на рис. 3 раздела 2.1 $rank\,(p) = rank\,(q) = 1, rank\,(r) = 2$.

Расширим понятие ранга на символы из $\mathcal{R}'$ и дизъюнкты над $\mathcal{R}'$.

Определение 20. Пусть $R \in \mathcal{R}'$. Вспомним, что $G : \mathbb{N}^{\mathcal{R}} \to \mathcal{R}'$ — биекция. Определим ранг символа R как минимальный ранг символа из G-прообраза P:

$$rank\,(R) \stackrel{\text{def}}{=} \min_{\substack{P \in \mathcal{R} \\ r(P) \neq 0}} r(P), \text{ где } r \equiv G^{-1}(R).$$

Определение 21. *Рангом дизъюнкта* C над $\mathcal{R}'$ назовём ранг символа в его заголовке, либо 0 для дизъюнктов-запросов:

$$rank\,(C) \stackrel{\text{def}}{=} \begin{cases} 0, & head(C) = \bot \\ rank\,(R), & C \in rules(R) \end{cases}$$

Определение 22. Будем говорить, что дизъюнкт C над $\mathcal{R}'$ *ранжирован*, если

- для всех $P(\overline{x}) \in Rec\,(C)$ выполняется $rank\,(P) = rank\,(C)$.
- для всех $P(\overline{x}) \in NRec\,(C)$ выполняется $rank\,(P) > rank\,(C)$;

По определению ранга очевидно, что все дизъюнкты системы $\mathcal{S}$ ранжированы. Докажем лемму, о сохранении свойства ранжированности произведением дизъюнктов. Эта лемма будет использована далее для доказательства того факта, что несмотря на увеличение количества символов и правил в $\mathcal{S}'$, в $\mathcal{S}'$ остаётся K различных рангов.

Лемма 9. Пусть $C_1, \ldots, C_m$ — ранжированные дизъюнкты над $\mathcal{R}'$. Тогда дизъюнкт $C_1 \times \ldots \times C_m$ также ранжирован.

Доказательство. Пусть r — мультимножество неинтерпретированных символов, являющихся заголовками дизъюнктов $C_1, \ldots, C_m$, а символ R является заголовком дизъюнкта C. По определению 14 $head(C) = \prod_{i=1}^{m} head(C_i)$, следовательно, $G^{-1}(R) = r$. Имеем

$$rank\,(C) = rank\,(R) = \min_{\substack{P \in \mathcal{R} \\ r(P) \neq 0}} r(P) = \min_{i=1}^{m} \big(rank\,(C_i) \big).$$

Поэтому для всех $i \in \{1,\ldots,m\}$ справедливо следующее:

$$rank\,(C_i) \geqslant rank\,(C)\,.$$

Пусть $P(\overline{x}) \in NRec\,(C)$. По определению 14 $NRec\,(C) = \bigcup\limits_{i=1}^{m} NRec\,(C_i)$. Следовательно, найдётся такое $i \in \{1,\ldots,m\}$, что $P(\overline{x}) \in NRec\,(C_i)$. В силу ранжированности C_i имеем:

$$rank\,(P) > rank\,(C_i) \geqslant rank\,(C)\,.$$

Пусть $P(\overline{x}) \in Rec\,(C)$. По определению 14 $Rec\,(C) = \left\{ \prod_{i=1}^{m} R_i(\overline{x}_i) \;\middle|\; \langle R_1(\overline{x}_1), \ldots R_m(\overline{x}_m)\rangle \in A \right\} \setminus \{head(C)\}$, где $A \in \left\lceil Rec_{/head}\,(C_1)\,, \ldots, Rec_{/head}\,(C_m)\,\right\rceil$. Следовательно, найдутся $R_1(\overline{x}_1) \in Rec_{/head}\,(C_1)\,, \ldots R_m(\overline{x}_m) \in Rec_{/head}\,(C_m)$ такие, что $P(\overline{x}) \equiv \prod_{i=1}^{m} R_i(\overline{x}_i)$, т.е. $P = G(\langle R_1, \ldots, R_m\rangle)$.

В силу ранжированности C_i для всех $R_i(\overline{x}_i) \in Rec_{/head}\,(C_i)$ выполняется $rank\,(R_i) = rank\,(C_i)$.

Итак, мы получили, что выполняется следующее:

$$rank\,(P) = \min_{i=1}^{m}\left(rank\,(R_i)\right) = \min_{i=1}^{m}\left(rank\,(C_i)\right) = rank\,(C)\,.$$

□

Лемма 10. Пусть C — дизъюнкт, выбранный алгоритмом CHCPRODUCT в строке 4 в начале i-й итерации. Для всякого дизъюнкта D, добавляемого в Q в конце итерации (строка 16), справедливо $rank\,(D) > rank\,(C)\,$.

Доказательство. Для начала заметим, что на любой итерации алгоритма все дизъюнкты в Q ранжированы. Это может быть легко доказано индукцией по количеству итераций: в начале первой итерации ранги всех дизъюнктов в Q равны нулю, а при добавлении $rules(R)$ в Q в строке 16 ранжированность сохраняется по лемме 9.

Выполним доказательство леммы индукцией по номеру итерации i.

База $i = 1$**.** В строке 2 очередь Q инициализируется запросами системы. Значит, в начале первой итерации $rank\,(C) = 0$, и утверждение леммы справедливо.

Переход $i \to i+1$. В строке 5 алгоритм получает множество $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_1)$ нерекурсивных атомов произвольно выбранного дизъюнкта C, для которых по ранжированности C выполняется $rank\,(R_j) > rank\,(C)$. Так как $R = G(\{\langle R_1, \ldots, R_m\rangle\})$, то для всех $j \in \{1, \ldots, m\}$ имеем:

$$rank\,(R) = \min_{j=1}^{m} \big(rank\,(R_j)\big) > rank\,(C)\,. \tag{2.15}$$

По построению множество N содержит символ R и рекурсивные с ним символы. Следовательно, в силу ранжированности C для всех $P \in N$ имеем $rank\,(P) = rank\,(R)$. Поэтому для каждого дизъюнкта $D \in rules(P)$, добавляемого в строке 16 в Q, имеем:

$$rank\,(D) = rank\,(P) > rank\,(C)\,.$$

□

Только что доказанная лемма, в сущности, показывает, что алгоритм CHCprodUCT строит систему $\mathcal{S}'$ в порядке обхода её графа зависимостей.

Пусть w — *ширина* системы $\mathcal{S}$, т.е. максимальное количество неинтерпретированных атомов в теле дизъюнктов системы $\mathcal{S}$.

Определение 23. *Размером дизъюнкта* C над $\mathcal{R}'$ назовём размер символа в его заголовке, либо 0, если C является запросом, т.е.:

$$size\,(C) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{, если } head(C) \equiv \bot \\ size\,(R)\,, & \text{если } C \in rules(R). \end{cases}$$

Лемма 11. На любой итерации алгоритма CHCprodUCT для любого дизъюнкта $C \in Q$ верно

$$size\,(C) \leqslant w^{rank(C)} \text{ и } |NRec\,(C)| \leqslant w^{rank(C)+1}. \tag{2.16}$$

Доказательство. Выполним доказательство индукцией по количеству итераций i алгоритма.

База $i = 1$. В начале первой итерации Q содержит только запросы в $\mathcal{S}$, для которых утверждение тривиально выполняется.

Переход $i \to i{+}1$**.** Пусть в начале i-й итерации для всех дизъюнктов в Q справедливо (2.16). Тогда для дизъюнкта C, выбираемого из Q (строка 4), также справедливо (2.16). В строке 5 из C извлекается множество его неинтерпретированных атомов $\{R_1(\overline{x}_1),\ldots,R_m(\overline{x}_m)\}$, которое используется для построения символа $R = G(\langle R_1,\ldots,R_m\rangle)$. По (2.16) имеем $m \leqslant w^{rank(C)+1}$. По лемме 7 для всех $i \in \{1,\ldots,m\}$ выполнено $R_i \in \mathcal{R}$. Следовательно, $size\,(R) = m$. Далее алгоритм CHCprodUCT строит множество символов N, рекурсивных с R. Как следует из доказательства леммы 3, суммарный размер всех рекурсивных символов в произведении дизъюнктов равен размеру символа в заголовке этого произведения, поэтому для всех $P \in N$ имеем $size\,(P) = size\,(R) = m$. По лемме 10 для каждого дизъюнкта D, добавляемого в Q (строка 16), имеем $rank\,(D) > rank\,(C)$, т.е. $rank\,(D) \geqslant rank\,(C) + 1$. Получили, что для каждого такого $D \in rules(P)$ справедливо следующее:

$$size\,(D) = size\,(P) = size\,(R) = m \leqslant w^{rank(C)+1} \leqslant w^{rank(D)}.$$

Поскольку $size\,(P) = m$, то в виду (2.2) имеем $D = C_1 \times \cdots \times C_m$ для некоторых $C_1,\ldots,C_m$ из $\mathcal{S}$, т.е. $|NRec\,(C_i)| \leqslant w$ для всех $i \in \{1,\ldots,m\}$. По определению 14 справедливо $NRec\,(D) = \bigcup\limits_{i=1}^{m} NRec\,(C_i)$. В силу $m \leqslant w^{rank(D)}$ имеем следующее:

$$|NRec\,(D)| \leqslant \sum_{i=1}^{m} |NRec\,(C_i)| \leqslant m \cdot w \leqslant w^{rank(D)+1}.$$

Итак, мы получили, что (2.16) справедливо для каждого D, добавляемого в Q в конце итерации, что заканчивает доказательство индукционного перехода. □

Лемма 11 имеет два важных следствия, качественно и количественно характеризующих алгоритм CHCprodUCT.

Теорема 17. Алгоритм CHCprodUCT всегда завершается.

Доказательство. Заметим, что ни один дизъюнкт C не может попасть в Q после его удаления в строке 4: дизъюнкты могут быть повторно добавлены только в строке 16. При первом добавлении символ R в заголовке окажется в множестве V (строка 17), вследствие чего условный переход в строке 15 на следующих итерациях не будет выполнен.

$$
\begin{aligned}
P_1(\overline{x}_1) \wedge \cdots \wedge P_1(\overline{x}_w) &\to \bot \\
P_2(\overline{x}_1) \wedge \cdots \wedge P_2(\overline{x}_w) &\to P_1(\overline{v}_{P_1}) \\
&\cdots \\
P_n(\overline{x}_1) \wedge \cdots \wedge P_n(\overline{x}_w) &\to P_{n-1}(\overline{v}_{P_{n-1}}) \\
\top &\to P_n(\overline{v}_{P_n})
\end{aligned}
$$

Рис. 5. Система дизъюнктов, на которой достигается $M = w^n$.

Пусть n — количество символов в системе $\mathcal{S}$. Поскольку ранг любого символа ограничен сверху n, то по лемме 11 размер любого дизъюнкта в Q ограничен w^n. Количество символов в $\mathcal{R}'$ ограниченного сверху размера также ограничено сверху, поскольку ограничено сверху количество мультимножеств ограниченного размера, а G — биекция. По (2.2) количество правил для каждого символа в $\mathcal{R}'$ конечно. Следовательно, на протяжении всей работы алгоритма в Q может попадать лишь конечное множество дизъюнктов, причём каждый из них не может оказаться в Q повторно. Для завершения доказательства заметим, что каждая итерация удаляет из Q в точности один дизъюнкт (строка 4). Следовательно, алгоритм может сделать лишь конечное конечное число итераций перед тем, как Q опустеет. □

Теорема 18. Пусть n — количество неинтерпретированных символов в системе $\mathcal{S}$. Тогда $\max_{C \in \mathcal{S}'} \big(size\,(C) \big) \leqslant w^n$.

Доказательство. Заметим, что в строке 12 алгоритм CHCPRODUCT добавляет в систему $\mathcal{S}'$ дизъюнкт C из Q с заменой его нерекурсивных атомов, т.е. ранг C не меняется. По определению ранга, он не превосходит количества сильно связных компонент графа зависимостей системы $\mathcal{S}$, которое не превосходит числа вершин n. Следовательно, по лемме 11 для всех $C \in \mathcal{S}'$, выполнено $size\,(C) \leqslant w^n$. □

Оценка $size\,(C)$ в теореме 18 достигается. Рассмотрим систему на рис. 5. Для неё количество сильно связных компонент графа зависимостей равно количеству неинтерпретированных символов n. На первом шаге алгоритм CHCPRODUCT рассматривает дизъюнкт-запрос с $m = w$ и добавляет в Q пра-

вило

$$\underbrace{P_2(\overline{x}_1^1) \wedge \cdots \wedge P_2(\overline{x}_w^w)}_{w^2 \text{ раз}} \to P_1^w(\overline{v}_{P_1}^1, \ldots, \overline{v}_{P_1}^w),$$

где $P_1^w \equiv G(\{P_1 \mapsto w\})$.

Далее заметим, что все P_i и P_j нерекурсивны для всех i и j. Следовательно, на втором шаге будет $m = w^2$, на третьем — $m = w \cdot w^2$ и т.д. На n-й итерации алгоритм рассмотрит дизъюнкт

$$\underbrace{P_n(\overline{x}_1^1) \wedge \cdots \wedge P_n(\overline{x}_w^{w^{n-1}})}_{w^n \text{ раз}} \to P_{n-1}^{w^{n-1}}(\overline{v}_{P_{n-1}}^1, \ldots, \overline{v}_{P_{n-1}}^{w^{n-1}}).$$

На $n+1$-й итерации алгоритм рассмотрит дизъюнкт

$$C \equiv \top \to P_n^{w^n}(\overline{v}_{P_n}^1, \ldots, \overline{v}_{P_n}^{w^n}),$$

т.е. достигается $size\,(C) = w^n$.

Пусть $h \stackrel{\text{def}}{=} \max_{P \in \mathcal{R}} |rules(P)|$.

Теорема 19. Для всех символов R, входящих в систему $\mathcal{S}'$, верно $|rules(R)| \leqslant h^{(w^n)}$.

Доказательство. Пусть $R \in \mathcal{R}'$ — символ, входящий в систему $\mathcal{S}'$, и $r \equiv G^{-1}(R)$.

По теореме 18 имеем следующее:

$$size\,(R) = \sum_{P \in \mathcal{R}} r(P) \leqslant w^n.$$

В силу (2.2) справедливо, что

$$|rules(R)| = \prod_{P \in \mathcal{R}} |rules(P)|^{r(P)} \leqslant \prod_{P \in \mathcal{R}} h^{r(P)} = h^{\sum_{P \in \mathcal{R}} r(P)} \leqslant h^{(w^n)}.$$

□

Оценка $h^{(w^n)}$, указанная в теореме 19, достигается алгоритмом CHCPRODUCT. Рассмотрим, к примеру, систему на рис. 6. В конце первой итерации алгоритм добавляет h^w правил для символа P_1^w в Q. В результате их обработки добавляется $h^{(w^2)}$ правил для $P_2^{w^2}$. В конце концов, для символа P_n в $\mathcal{S}'$ добавляется $h^{(w^n)}$ правил.

$$\underbrace{P_1(\overline{x}_1) \wedge \cdots \wedge P_1(\overline{x}_w)}_{w \text{ раз}} \to \bot$$

$$h \text{ раз} \begin{cases} \underbrace{P_2(\overline{x}_1^1) \wedge \cdots \wedge P_2(\overline{x}_w^1)}_{w \text{ раз}} \to P_1(\overline{v}_{P_1}) \\ \cdots \\ \underbrace{P_2(\overline{x}_1^h) \wedge \cdots \wedge P_2(\overline{x}_w^h)}_{w \text{ раз}} \to P_1(\overline{v}_{P_1}) \end{cases}$$

$$\cdots$$

$$h \text{ раз} \begin{cases} \underbrace{P_n(\overline{x}_1^1) \wedge \cdots \wedge P_n(\overline{x}_w^1)}_{w \text{ раз}} \to P_{n-1}(\overline{v}_{P_{n-1}}) \\ \cdots \\ \underbrace{P_n(\overline{x}_1^h) \wedge \cdots \wedge P_n(\overline{x}_w^h)}_{w \text{ раз}} \to P_{n-1}(\overline{v}_{P_{n-1}}) \end{cases}$$

$$\cdots$$

$$h \text{ раз} \begin{cases} \varphi_1 \to P_n(\overline{v}_{P_n}) \\ \cdots \\ \varphi_h \to P_n(\overline{v}_{P_n}) \end{cases}$$

Рис. 6. Система дизъюнктов, на которой достигается $h^{(w^n)}$ правил.

Теорема 20. Пусть $m = \max_{C \in \mathcal{S}'} \big(size\,(C)\big)$ и $h > 1$. Тогда

$$|\mathcal{S}'| \leqslant |queries| + \frac{\left(\sum_{k=0}^{n-1}(-1)^{n-k-1}C_{m+k}^{m}(h-1)^k\right)\cdot h^{m+1} + (-1)^n}{(h-1)^n} - 1.$$

Доказательство. Очевидно, что $|\mathcal{S}'| \leqslant |queries| + \sum\limits_{\substack{P \in \mathcal{R}' \\ P \text{ входит в } \mathcal{S}'}} |rules(P)|$. Как показано в доказательстве теоремы 19, имеем

$$|rules(P)| \leqslant h^{size(P)}.$$

Количество символов $P \in \mathcal{R}'$ размера k не превосходит количества мультимножеств с элементами из $\mathcal{R}$ размера k. Последнее, в свою очередь, равно числу выборок с повторениями размера k из множества размера $|\mathcal{R}| = n$, т.е. C_{n+k-1}^{k} [131]. Следовательно, имеем, что

$$\sum_{\substack{P \in \mathcal{R}' \\ P \text{ входит в } \mathcal{S}'}} |rules(P)| \leqslant \sum_{k=1} C_{n+k-1}^{k} \cdot h^k.$$

Введем следующее обозначение:

$$S(n,m,h) \stackrel{\text{def}}{=} \sum_{k=0}^{m} C_{n+k-1}^{k} \cdot h^k = 1 + C_n^1 \cdot h + \ldots + C_{n+m-1}^{m} h^m.$$

Таким образом имеем, что $|\mathcal{S}'| \leqslant |queries| + S(n,m,h) - 1$.

Теперь осталось вычислить частичную сумму $S(n,m,h)$. Заметим, что

$$S(n,m,h)\cdot(h-1) = C_{n+m-1}^{m}\cdot h^{m+1} - \big(1 + (C_n^1 - C_{n-1}^0)\cdot h + \ldots + (C_{n+m-1}^{m} - C_{n+m-2}^{m-1})\cdot h^m\big).$$

Так как для всех натуральных n и k верно $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$, то имеем следующее:

$$S(n,m,h)\cdot(h-1) = C_{n+m-1}^{m}\cdot h^{m+1} - \big(1 + C_{n-1}^1 \cdot h + \ldots + C_{n+m-2}^{m}\cdot h^m\big),$$

$$S(n,m,h)\cdot(h-1) = C_{n+m-1}^{m}\cdot h^{m+1} - S(n-1,m,h),$$

$$S(n,m,h)\cdot(h-1)^2 = C_{n+m-1}^{m}\cdot(h-1)\cdot h^{m+1} - C_{n+m-2}^{m}\cdot h^{m+1} + S(n-2,m,h).$$

Повторив эту процедуру $n-1$ раз, получим следующее:

$$S(n,m,h)\cdot(h-1)^{n-1} = \left(\sum_{k=1}^{n-1}(-1)^{n-k-1}C_{m+k}^{m}\cdot(h-1)^{k-1}\cdot h^{m+1}\right) + (-1)^{n-1}\cdot S(1,m,h).$$

Очевидно, что выполнено следующее утверждение:

$$S(1,m,h) = \sum_{k=0}^{m} C_k^k \cdot h^k = 1 + h + \ldots + h^m = \frac{h^{m+1} - 1}{h - 1}.$$

Поэтому мы имеем, что истинно нижеследующее:

$$S(n,m,h) = \frac{\left(\sum_{k=0}^{n-1}(-1)^{n-k-1} C_{m+k}^m (h-1)^k\right) \cdot h^{m+1} + (-1)^n}{(h-1)^n}.$$

□

Теоремы 20 и 18 дают точную оценку количества дизъюнктов в $\mathcal{S}'$. Так как одна итерация алгоритма CHCprodUCT добавляет в $\mathcal{S}'$ в точности один дизъюнкт (строка 12), то оценка количества дизъюнктов в $\mathcal{S}'$ является также и оценкой количества итераций алгоритма.

Для упрощения оценки в теореме 20 заметим, что поскольку C_{m+k}^m строго убывают с убыванием k, и сумма в коэффициенте при h^{m+1} является знакопеременной, а также в силу теоремы 18 имеем $m \leqslant w^n$, то получаем, что выполнено следующее:

$$|\mathcal{S}'| \leqslant |queries| + \frac{C_{m+n-1}^m \cdot h^{m+1}}{h-1} \leqslant |queries| + \frac{C_{w^n+n-1}^{w^n} \cdot h^{w^n+1}}{h-1}.$$

При $w = 1$ система $\mathcal{S}$ линейна, и алгоритм CHCprodUCT не меняет её, т.е. $|\mathcal{S}| = |\mathcal{S}'|$. При $w \geqslant 2$ асимптотику $C_{w^n+n-1}^{w^n}$ можно оценить как $O\left(w^{(n^2)}\right)$.

Стоит оговориться, что теорема 20 верна при $h \geqslant 2$, поэтому следует рассмотреть случаи $h \in \{0,1\}$. При $h = 0$ в $\mathcal{S}'$ отсутствуют правила неинтерпретированных символов, т.е. $|\mathcal{S}'| = |queries|$.

При $h = 1$, в силу $\sum_{k=0}^{m} C_{n+k-1}^k = C_{n+m}^m$ [131], имеем:

$$|\mathcal{S}'| \leqslant |queries| + \sum_{k=1}^{m} C_{n+k-1}^k = |queries| + C_{n+m}^m - 1 \leqslant |queries| + C_{w^n+n}^{w^n}.$$

Во всех трёх случаях получаем следующую итоговую оценку сверху количества итераций алгоритма CHCprodUCT:

$$O\left(|queries| + w^{(n^2)} \cdot h^{(w^n)}\right).$$

2.8 Выбор накрытия рекурсивных атомов

Заметим, что в определении 14 допускается произвольное накрытие рекурсивных атомов перемножаемых дизъюнктов (см. (2.1)); конкретный выбор накрытия не был важен ни в одном доказательстве теорем данной главы. Свобода в выборе конкретного накрытия позволяет определять различные стратегии слияния рекурсивных атомов, делая алгоритм CHCprouct более гибким.

Прямолинейным способом выбора накрытия будет сопоставлять первые элементы множеств с первыми, вторые — со вторыми и т.д. до тех пор, пока некоторое множество не исчерпается; далее можно сопоставлять любой его элемент элементам оставшихся множеств. Как показывает практика, этот способ является лучшим по быстродействию и показывает хорошие результаты, т.к. на практике чаще всего в теле каждого дизъюнкта содержится не более одного рекурсивного атома.

Тем не менее, если в телах некоторых дизъюнктов имеется более одного рекурсивного атома, то выборы различных накрытий могут повлиять на существование сертификата выполнимости синхронизированной системы. Этот вопрос более подробно изучен в главе 4.

2.9 Обсуждение

Алгоритм CHCprouct: итоги. Итак, был представлен алгоритм, который преобразует нелинейную систему дизъюнктов Хорна к виду, где у каждого дизъюнкта в теле содержится не более одного нерекурсивного атома. Этот алгоритм завершается на любой входной системе $\mathcal{S}$ и возвращает систему $\mathcal{S}'$, эквивыполнимую исходной. При этом справедливо следующее:

- резолютивное опровержение $\mathcal{S}$ существует тогда и только тогда, когда существует резолютивное опровержение $\mathcal{S}'$;
- если у $\mathcal{S}$ существует сертификат выполнимости, то он существует и у $\mathcal{S}'$; обратное верно не всегда.

Таким образом, с помощью алгоритма CHCprouct можно частично решить проблему непредставимости символьных моделей, преобразовав нелинейную систему и далее воспользоваться существующими инструментами решения систем дизъюнктов Хорна (см. раздел 1.4).

Ещё одной особенностью алгоритма является то, что он «очищает» преобразованную систему от неиспользуемых неинтерпретированных символов и соответствующих правил, упрощая построение решения, как было показано в разделе 2.7.2.

Основным ограничением алгоритма является его экспоненциальная сложность. Как было показано, существуют системы, которые преобразуются алгоритмом с экспоненциальным увеличением количества правил. Это может повлечь ситуацию, когда некоторая система $\mathcal{S}$ относительно быстро решается существующими инструментами, но процесс её преобразования в $\mathcal{S}'$ вместе с последующим решением не укладывается в реалистичные временные ограничения. Такие случаи были зафиксированы на практике (см. главу 5). Решению данной проблемы посвящены две следующие главы диссертации.

Границы применимости. Оправданным является следующий вопрос: для какого класса систем алгоритм CHCprodUCT полезен? Другими словами, как можно описать класс систем дизъюнктов с непредставимыми моделями, для которых алгоритм CHCprodUCT будет создавать системы с представимыми моделями?

К сожалению, современное состояние данной предметной области не позволяет дать точного ответа на этот вопрос. Дело в том, что полноценное описание класса таких систем требует, как минимум, наличия подходов к описанию класса представимых неподвижных точек отношений переходов систем дизъюнктов в произвольных теориях (см. раздел 2.3). Однако на сегодняшний день такое описание существует лишь для интерпретаций с конечным носителем и арифметики Пеано [18], которая неразрешима, и потому на практике не применяется в системах автоматического решения дизъюнктов Хорна. Поэтому вместо точного ответа дадим неформальные пояснения.

Алгоритм CHCprodUCT полезен для систем, выражающих условия верификации программ, выполняющих цепочку обработки данных. Приведём ряд примеров.

Вернёмся к системе из примера 1:

$$
\begin{aligned}
T = leaf \wedge n = 0 &\to size(T, n) \\
T = node(v, L, R) \wedge n = 1 + n^L + n^R \wedge size(L, n^L) \wedge size(R, n^R) &\to size(T, n) \\
T = leaf \wedge s = 0 &\to sum(T, s) \\
T = node(v, L, R) \wedge s = v + s^L + s^R \wedge & \\
sum(L, s^L) \wedge sum(R, s^R) &\to sum(T, s) \\
T = leaf \wedge U = leaf &\to inc(T, U) \\
T = node(v, L, R) \wedge U = node(v+2, L', R') \wedge & \\
inc(L, L') \wedge inc(R, R') &\to inc(T, U) \\
s' \neq s + 2n \wedge size(T, n) \wedge sum(T, s) \wedge inc(T, T') \wedge sum(T', s') &\to \bot
\end{aligned}
$$

Эта система выражает условия верификации для программы, которая вычисляет число n — количество узлов в дереве T, а также строит по дереву T дерево T', для которого значение в каждом узле получается увеличением значения в соответствующем узле T на 2, и, наконец, считает суммы s и s' в узлах деревьев, проверяя, что $s' = s + 2n$. В сущности, эта программа производит три обхода дерева T и один обход дерева T', при этом каждое значение в T' связано с соответствующим значением в T.

Синхронизация позволяет переписать эту систему таким образом, что доступ к каждому узлу дерева T и соответствующего ему узла дерева T' происходит *одновременно*. Из-за большого размера синхронизированная система не будет приведена целиком, но приведём упрощённую систему, эквивыполнимую ей:

$$
\begin{aligned}
T = leaf \wedge T' = leaf \wedge s' = 0 \wedge s = 0 \wedge s' = 0 &\to s^3 i(T, T', n, s, s') \\
T = node(v, L, R) \wedge T' = node(v+2, L', R') \wedge & \\
\wedge n = 1 + n_L + n_R \wedge s = v + s_L + s_R \wedge s' = v + 2 + s'_L + s'_R \wedge & \\
\wedge s^3 i(L, L', n_L, s_L, s'_L) \wedge s^3 i(R, R', n_R, s_R, s'_R) &\to s^3 i(T, T', n, s, s') \\
s' \neq s + 2n \wedge s^3 i(T, T', n, s, s') &\to \bot
\end{aligned}
$$

Поскольку предикатный символ $s^3 i$ теперь применяется ко всем аргументам сразу (T, T', s, s', n), поэтому тот же самый язык ограничений оказывается достаточно выразительным для представления символьной модели. В данном случае у изначальной системы не существует сертификата выполнимости, но

для синхронизированной системы появляется простой сертификат выполнимости:

$$\mathcal{I} \stackrel{\text{def}}{=} \{s^{\mathcal{J}} i \mapsto s' = s + 2n\}.$$

Подобные цепочки преобразований данных часто встречаются в распределённых системах. В качестве примера можно привести модель map-reduce [132], а также функциональном программировании, где часто встречаются цепочки операций над индуктивными типами данных типа `fold`, `filter`, `map`, `reduce`, `iter`, `unfold` и т.д. Можно также упомянуть о сетевых приложениях (извлечение информации из множественных пакетов и последующая её обработка), драйверах устройств (цепочка обработки потока данных с устройств) и т.д. Для подобных программ алгоритм CHCprodуct является инструментом, позволяющим автоматически породить систему дизъюнктов, описывающую синхронный доступ всех промежуточных вычислений к данным. Это значительно повышает выразительность языка ограничений для представлений инвариантов таких вычислений, что на практике позволяет решателям успешно выводить сертификаты выполнимости для синхронизированных систем, в то время как для изначальных систем они не завершаются (см. главу 5).

Алгоритм CHCprodуct может быть эффективно использован в реляционной верификации, где сопоставляются два или больше похожих друг на друга вычисления. В этом случае CHCprodуct автоматически «выравнивает» поведения программ, позволяя решателю выводить логические соотношения промежуточных результатов этих вычислений.

Глава 3. Реляционные символьные интерпретации

В предыдущей главе описано синхронизирующее преобразование дизъюнктов. Как было показано, оно частично решает проблему непредставимости моделей в языке ограничений, но может порождает системы с экспоненциальным числом правил.

В данной главе вводятся понятие *реляционного сертификата выполнимости*, обобщающее понятие «обычных» сертификатов выполнимости. Ключевая идея состоит в том, чтобы сопоставлять формулы *мультимножествам* реляционных символов, а не каждому символу в отдельности. Это позволяет определять в языке ограничений *отношения* между переменными различных неинтерпретированных атомов вместо «резюмирования» семантики каждого символа в отдельности. Поэтому реляционные сертификаты являются более выразительным видом символьных решений нелинейных систем дизъюнктов Хорна по сравнению с “классическими” сертификатами выполнимости. Будет формально показано, что аналогично обычным сертификатам, реляционные сертификаты являются синтаксическим свидетельством выполнимости системы дизъюнктов, а также, что если у системы существует сертификат выполнимости, то существует и реляционный сертификат выполнимости.

3.1 Определение реляционных символьных интерпретаций

Телом мультимножества неинтерпретированных символов $r = \{P_1 \mapsto k_1, \ldots, P_n \mapsto k_n\} \in \mathbb{N}^{\mathcal{R}}$ назовём конъюнкцию тел символов из этого мультимножества с переименованием переменных:

$$body(r) \stackrel{\text{def}}{=} \underbrace{body(P_1) \stackrel{+}{\wedge} \ldots \stackrel{+}{\wedge} body(P_1)}_{k_1 \text{ раз}} \stackrel{+}{\wedge} \ldots \stackrel{+}{\wedge} \underbrace{body(P_n) \stackrel{+}{\wedge} \ldots \stackrel{+}{\wedge} body(P_n)}_{k_n \text{ раз}}.$$

Напомним, $\varphi \stackrel{+}{\wedge} \psi$ — конъюнкция φ и ψ, гарантирующая, что свободные переменные операндов различны:

$$\varphi(\overline{x}) \stackrel{+}{\wedge} \psi(\overline{y}) \stackrel{\text{def}}{=} (\varphi \wedge \psi)\,(\overline{x} \uplus \overline{y}).$$

Вспомним, что $\overline{v}_R$ означает кортеж переменных в заголовке каждого правила символа $R \in \mathcal{R}$. Расширим эту нотацию для мультимножества r:

$$\overline{v}_r \stackrel{\text{def}}{=} \underbrace{\overline{v}_{P_1} \uplus \ldots \uplus \overline{v}_{P_1}}_{k_1 \text{ раз}} \uplus \ldots \uplus \underbrace{\overline{v}_{P_n} \uplus \ldots \uplus \overline{v}_{P_n}}_{k_n \text{ раз}}.$$

Здесь $\overline{x} \uplus \overline{y}$ означает конкатенацию кортежей с переименованием переменных, гарантирующим, что все переменные в результирующем кортеже попарно различны. Например, $\langle x,y,z\rangle \uplus \langle x,z\rangle = \langle x_1,y,z_1,x_2,z_2\rangle$.

Вспомним также, что мы сопоставляем мультимножества кортежам; поэтому, если $\overline{R} = \langle R_1,\ldots,R_m\rangle$, то будем также использовать нотацию $body(\overline{R})$, $body(R_1,\ldots,R_m)$, $\overline{v}_{\overline{R}}$ и $\overline{v}_{R_1,\ldots,R_m}$.

Определение 24. Пусть $\mathcal{S}$ — система дизъюнктов над множеством реляционных символов $\mathcal{R}$. *Реляционной символьной интерпретацией* назовём частичное отображение $\mathcal{J} : \mathbb{N}^{\mathcal{R}} \to \mathcal{A}$, отображающее мультимножество r в ограничение над переменными $\overline{v}_r$.

Для $\mathcal{J}$ будем обозначать как $dom(\mathcal{J})$ её область определения. Без потери общности будем считать, что $\mathcal{R} \subseteq dom(\mathcal{J})$ (если $\{R \mapsto 1\} \notin dom(\mathcal{J})$, то отобразим $\{R \mapsto 1\}$ в $\top$).

Пусть $R_1,\ldots,R_m \in \mathcal{R}$, φ — ограничение, $\mathcal{J}$ — реляционная символьная интерпретация. Определим *реляционную подстановку* в тело мультимножества символов в два этапа. Вначале определим реляционную подстановку в формулы вида $\varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m)$ (заметим, что такой вид имеет тело любого дизъюнкта и любая конъюнкция тел дизъюнктов):

$$[\![\varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m)]\!]_{\mathcal{J}} \stackrel{\text{def}}{=} \tag{3.1}$$
$$\varphi \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k} \in \{R_1,\ldots,R_m\} \\ \langle R_{i_1},\ldots,R_{i_k}\rangle \in dom(\mathcal{J})}} \mathcal{J}(\langle R_{i_1},\ldots,R_{i_k}\rangle)[\langle \overline{x}_{i_1},\ldots,\overline{x}_{i_k}\rangle / \overline{v}_{R_{i_1},\ldots,R_{i_k}}].$$

Неформально, (3.1) перебирает все возможные варианты замены сразу нескольких неинтерпретированных атомов ограничением в теле дизъюнкта.

Далее, пусть $\{B_{i,j}\}_{\substack{i\in\{1,\ldots,k\} \\ j\in\{1,\ldots,\ell_i\}}}$ — семейство тел некоторых дизъюнктов $C_{i,j}$. Заметим, что реляционная подстановка в любую конъюнкцию $B_{i_1,j_1} \wedge \cdots \wedge B_{i_k,j_k}$

определена в (3.1). Положим

$$\left[\!\!\left[\bigwedge_{i=1}^{k}\bigvee_{j=1}^{\ell_i} B_{i,j}\right]\!\!\right]_{\mathcal{J}} \stackrel{\text{def}}{=} \bigvee_{\substack{1\leqslant j_1\leqslant \ell_1\\ \cdots\\ 1\leqslant j_k\leqslant \ell_k}} [\![B_{1,j_1}\wedge\cdots\wedge B_{k,j_k}]\!]_{\mathcal{J}}\,. \tag{3.2}$$

В (3.2) определена реляционная подстановка в тело любого мультимножества неинтерпретированных символов, которое, по определению, есть конъюнкция дизъюнкций тел дизъюнктов.

Пример 12. Рассмотрим следующую систему дизъюнктов:

$$\begin{aligned}
\varphi_1 &\to P(x_1,x_2)\\
\varphi_2\wedge P(x'_1,x'_2)\wedge P(x''_1,x''_2) &\to P(x_1,x_2)\\
\psi_1 &\to Q(y)\\
\psi_2\wedge g(y') &\to Q(y)\\
\varphi\wedge P(z_1,z_2)\wedge Q(z_3) &\to \bot
\end{aligned}$$

и следующую реляционную интерпретацию[1]:

$$\mathcal{J}=\{P\mapsto\top, Q\mapsto\eta_1(y), \langle P,Q\rangle\mapsto\eta_2(x_1,x_2,y)\}$$

Результат подстановки $\mathcal{J}$ в $body(P,Q)$ таков:

$$\begin{aligned}
[\![body(P,Q)]\!]_{\mathcal{J}} = \Big[\!\Big[&\Big(\varphi_1\vee\big(\varphi_2\wedge P(x'_1,x'_2)\wedge P(x''_1,x''_2)\big)\Big)\wedge\\
&\Big(\psi_1\vee\big(\psi_2\wedge Q(y')\big)\Big)\Big]\!\Big]_{\mathcal{J}} =\\
[\![\varphi_1\wedge\psi_1]\!]_{\mathcal{J}}\vee\,&[\![\varphi_1\wedge\psi_2\wedge Q(y')]\!]_{\mathcal{J}}\vee\\
&[\![\varphi_2\wedge\psi_1\wedge P(x'_1,x'_2)\wedge P(x''_1,x''_2)]\!]_{\mathcal{J}}\vee\\
&[\![\varphi_2\wedge\psi_2\wedge P(x'_1,x'_2)\wedge P(x''_1,x''_2)\wedge Q(y')]\!]_{\mathcal{J}} =\\
(\varphi_1\wedge\psi_1)\vee\,&(\varphi_1\wedge\psi_2\wedge\eta_1(y'))\vee(\varphi_2\wedge\psi_1)\vee\\
&(\varphi_2\wedge\psi_2\wedge\eta_1(y')\wedge\eta_2(x'_1,x'_2,y')\wedge\eta_2(x''_1,x''_2,y'))
\end{aligned}$$

Единичным будем называть мультимножество $\{R\mapsto 1\}$, состоящее из единственного элемента $R\in\mathcal{R}$. Покажем, что реляционные символьные интерпретации обобщают «классические» символьные интерпретации.

[1]Для простоты нотации мультимножества вида $\{P\mapsto 1\}$ записываются как P, а мультимножества вида $\{P\mapsto 1, Q\mapsto 1\}$ — как $\langle P,Q\rangle$

Теорема 21. Пусть $\mathcal{I} : \mathcal{R} \to \mathcal{A}$ — символьная интерпретация. Тогда частичное отображение $\mathcal{J} : \mathcal{R} \to \mathcal{A}$, такое, что для всех $r \in \mathbb{N}^{\mathcal{R}}$,

$$\mathcal{J}(r) \stackrel{\text{def}}{=} \begin{cases} \mathcal{I}(P), & \text{если } r = \{P \mapsto 1\} \text{ — единичное мультимножество,} \\ \texttt{не определено}, & \text{иначе,} \end{cases}$$

является реляционной символьной интерпретацией, причём для любого мультимножества $r \in \mathbb{N}^{\mathcal{R}}$,

$$[\![body(r)]\!]_{\mathcal{I}} \sim [\![body(r)]\!]_{\mathcal{J}}\,.$$

Доказательство. То, что $\mathcal{J}$ является реляционной символьной интерпретацией, очевидно из определения. Пусть $body(r) = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{\ell_i} B_{i,j}$. Пусть для некоторых $1 \leqslant j_1 \leqslant \ell_1, \ldots, 1 \leqslant j_k \leqslant \ell_k$

$$B_{1,j_1} \wedge \cdots \wedge B_{k,j_k} = \varphi \wedge R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m).$$

По (3.1),

$$[\![B_{1,j_1} \wedge \cdots \wedge B_{k,j_k}]\!]_{\mathcal{J}} = \varphi \wedge \bigwedge_{i=1}^{m} \mathcal{J}(\{R_i \mapsto 1\})(\overline{x}_i) =$$
$$= \varphi \wedge \mathcal{I}(R_1)[\overline{x_1}/\overline{v}_{R_1}] \cdots \wedge \mathcal{I}(R_m)[\overline{x_m}/\overline{v}_{R_m}] = [\![B_{1,j_1} \wedge \cdots \wedge B_{k,j_k}]\!]_{\mathcal{I}}\,.$$

Следовательно, по правилу дистрибутивности

$$A \wedge (B_1 \vee \cdots \vee B_\ell) \sim (A \wedge B_1) \vee \cdots \vee (A \wedge B_\ell),$$

имеем

$$[\![body(r)]\!]_{\mathcal{I}} = \left[\!\!\left[\bigwedge_{i=1}^{k} \bigvee_{j=1}^{\ell_i} B_{i,j} \right]\!\!\right]_{\mathcal{I}} \sim [\![\bigvee_{\substack{1 \leqslant j_1 \leqslant \ell_1 \\ \cdots \\ 1 \leqslant j_k \leqslant \ell_k}} B_{1,j_1} \wedge \cdots \wedge B_{k,j_k}]\!]_{\mathcal{I}} =$$
$$= \bigvee_{\substack{1 \leqslant j_1 \leqslant \ell_1 \\ \cdots \\ 1 \leqslant j_k \leqslant \ell_k}} [\![B_{1,j_1} \wedge \cdots \wedge B_{k,j_k}]\!]_{\mathcal{I}} = \bigvee_{\substack{1 \leqslant j_1 \leqslant \ell_1 \\ \cdots \\ 1 \leqslant j_k \leqslant \ell_k}} [\![B_{1,j_1} \wedge \cdots \wedge B_{k,j_k}]\!]_{\mathcal{J}} = [\![body(r)]\!]_{\mathcal{J}}\,.$$

□

3.2 Реляционные сертификаты выполнимости

Определение 25. *Реляционным сертификатом выполнимости* системы $\mathcal{S}$ назовём любую безопасную и индуктивную реляционную символьную интерпре-

тацию $\mathcal{J}$, т.е. для которой выполнено следующее:

$$\text{для всех } C \in \mathit{queries},\ \mathcal{T} \models \forall\big([\![\mathit{body}(C)]\!]_{\mathcal{J}} \to \bot\big) \qquad \text{(безопасность)}$$
$$\text{для всех } r \in \mathit{dom}(\mathcal{J}),\ \mathcal{T} \models \forall\big([\![\mathit{body}(r)]\!]_{\mathcal{J}} \to \mathcal{J}(r)\big) \qquad \text{(индуктивность)}$$

Кроме реляционных подстановок, главная разница между «классическими» и реляционными сертификатами выполнимости состоит в том, что последние должны быть индуктивны относительно тел *мультимножеств* символов. Другими словами, в отличие от обычных сертификатов, $\mathcal{J}$ должно быть индуктивно относительно $\mathit{body}(r)$ для *неединичных* мультимножеств $r \in \mathit{dom}(\mathcal{J})$.

Реляционный сертификат выполнимости является более выразительным видом решения систем дизъюнктов Хорна, чем «классический» сертификат выполнимости. В следующей теореме показано, что реляционные сертификаты не менее выразительны, чем «классические», строгость этого отношения показана на примерах в следующем разделе.

Теорема 22. Если у системы дизъюнктов существует сертификат выполнимости, то существует и реляционный сертификат выполнимости.

Доказательство. Пусть $\mathcal{I}$ — сертификат выполнимости системы $\mathcal{S}$. Определим реляционную символьную интерпретацию $\mathcal{J}$ следующим образом:

$$\mathcal{J}(r) \stackrel{\text{def}}{=} \begin{cases} \mathcal{I}(P), & \text{если } r = \{P \mapsto 1\} \text{ — единичное мультимножество,} \\ \texttt{не определено}, & \text{иначе.} \end{cases}$$

По определению сертификата выполнимости (см. определение 8 в главе 1) имеем:

$$\text{для всех } C \in \mathit{queries},\ \mathcal{T} \models \forall([\![\mathit{body}(C)]\!]_{\mathcal{I}} \to \bot)\,,$$
$$\text{для всех } P \in \mathcal{R},\ \mathcal{T} \models \forall([\![\mathit{body}(P)]\!]_{\mathcal{I}} \to \mathcal{I}(P))\,.$$

По теореме 21 для любого дизъюнкта $C \in \mathit{queries}$ справедливо следующее:

$$[\![\mathit{body}(C)]\!]_{\mathcal{J}} \sim [\![\mathit{body}(C)]\!]_{\mathcal{I}}\,.$$

Чтобы в этом убедиться заметим, что теорема 21 верна для произвольных систем дизъюнктов над любым множеством неинтерпретированных символов

$\mathcal{R}$. Для каждого запроса $C \in \mathit{queries}$ введём новый нульместный неинтерпретированный символ Q_C и заменим запрос $C \equiv \mathit{body}(C) \to \bot$ правилом $\mathit{body}(C) \to Q_C$, после чего применим теорему 21 к единичному мультимножеству $\{Q_C \mapsto 1\}$. Следовательно, для всех $C \in \mathit{queries}$ имеем $\forall(\llbracket \mathit{body}(C) \rrbracket_{\mathcal{I}} \to \bot) \sim \forall\big(\llbracket \mathit{body}(C) \rrbracket_{\mathcal{J}} \to \bot\big)$.

По этой же теореме, для любого мультимножества $r \in \mathbb{N}^{\mathcal{R}}$ имеем следующее:

$$\llbracket \mathit{body}(r) \rrbracket_{\mathcal{J}} \sim \llbracket \mathit{body}(r) \rrbracket_{\mathcal{I}}.$$

Для завершения доказательства осталось заметить, что по определению $\mathcal{J}$ для всех $r \in \mathit{dom}(\mathcal{J})$ и для некоторого $P \in \mathcal{R}$ имеем $r = \{P \mapsto 1\}$, т.е. $\mathcal{J}(r) \equiv \mathcal{I}(P)$, следовательно, справедливо следующее:

$$\forall(\llbracket \mathit{body}(P) \rrbracket_{\mathcal{I}} \to \mathcal{I}(P)) \sim \forall\big(\llbracket \mathit{body}(r) \rrbracket_{\mathcal{J}} \to \mathcal{J}(r)\big).$$

□

3.3 Примеры

Рассмотрим снова систему дизъюнктов из примера 7:

$$\begin{aligned} x{=}0 \wedge z{=}0 &\to \mathit{mul}(x,y,z) \\ x > 0 \wedge x'{=}x-1 \wedge z{=}z'{+}y \wedge \mathit{mul}(x',y,z') &\to \mathit{mul}(x,y,z) \\ x{=}x' \wedge y{=}y' \wedge z{\neq}z' \wedge \mathit{mul}(x,y,z) \wedge \mathit{mul}(x',y',z') &\to \bot \end{aligned}$$

Хотя у этой системы не существует «классического» сертификата выполнимости, представимого в языке линейной целочисленной арифметики, у неё существует следующий реляционный сертификат выполнимости:

$$\begin{aligned} \mathcal{J} = \{&\mathit{mul} \mapsto \top, \\ \langle \mathit{mul},\mathit{mul} \rangle \mapsto &(x^{\mathit{mul}_1}{=}x^{\mathit{mul}_2} \wedge y^{\mathit{mul}_1}{=}y^{\mathit{mul}_2} \to z^{\mathit{mul}_1}{=}z^{\mathit{mul}_2})\}. \end{aligned}$$

Убедиться в том, что $\mathcal{J}$ является реляционным сертификатом выполнимости, можно, проверив безопасность и индуктивность $\mathcal{J}$. Для облегчения про-

верки этого факта приведём результаты реляционной подстановки:

$$\begin{aligned}
&[\![x{=}x' \wedge y{=}y' \wedge z{\neq}z' \wedge mul(x,y,z) \wedge mul(x',y',z')]\!]_{\mathcal{J}} \equiv \\
&\qquad \equiv x{=}x' \wedge y{=}y' \wedge z{\neq}z' \wedge (x = x' \wedge y = y' \to z = z') \\
&[\![body(mul,mul)]\!]_{\mathcal{J}} \equiv (x_1{=}0 \wedge z_1{=}0 \wedge x_2{=}0 \wedge z_2{=}0) \vee \\
&\quad \vee (x_1{=}0 \wedge z_1{=}0 \wedge x_2 > 0 \wedge x_2'{=}x_2 - 1 \wedge z_2{=}z_2'{+}y_2) \vee \\
&\quad \vee (x_1 > 0 \wedge x_1'{=}x_1 - 1 \wedge z_1{=}z_1'{+}y_1 \wedge x_2{=}0 \wedge z_2{=}0) \vee \\
&\quad \vee (x_1 > 0 \wedge x_1'{=}x_1 - 1 \wedge z_1{=}z_1'{+}y_1 \wedge x_2 > 0 \wedge x_2'{=}x_2 - 1 \wedge z_2{=}z_2'{+}y_2 \wedge \\
&\qquad (x_1 = x_2 \wedge y_1 = y_2 \to z_1 = z_2)).
\end{aligned}$$

Таким образом очевидно, что $\mathcal{J}$ безопасна, т.е. выполнено следующее:

$$\mathcal{N} \models \forall [\![x{=}x' \wedge y{=}y' \wedge z{\neq}z' \wedge mul(x,y,z) \wedge mul(x',y',z')]\!]_{\mathcal{J}}.$$

Формула $\mathcal{J}$ также является индуктивной, поскольку справедливо следующее:

$$\mathcal{N} \models \forall\big([\![body(mul,mul)]\!]_{\mathcal{J}} \to (x_1{=}x_2 \wedge y_1{=}y_2 \to z_1{=}z_2)\big).$$

Обратимся к системе из примера 1:

$$\begin{aligned}
T{=}leaf \wedge n{=}0 &\to size(T,n) \\
T{=}node(v,L,R){\wedge}n{=}1{+}n^L{+}n^R{\wedge}size(L,n^L){\wedge}size(R,n^R) &\to size(T,n) \\
T{=}leaf{\wedge}s{=}0 &\to sum(T,s) \\
T{=}node(v,L,R){\wedge}s{=}v{+}s^L{+}s^R{\wedge}& \\
sum(L,s^L){\wedge}sum(R,s^R) &\to sum(T,s) \\
T{=}leaf{\wedge}U{=}leaf &\to inc(T,U) \\
T{=}node(v,L,R){\wedge}U{=}node(v{+}2,L',R'){\wedge}& \\
inc(L,L'){\wedge}inc(R,R') &\to inc(T,U) \\
s'{\neq}s{+}2n \wedge size(T,n){\wedge}sum(T,s){\wedge}inc(T,T') \wedge sum(T',s') &\to \bot
\end{aligned}$$

Она также имеет реляционный сертификат выполнимости, который выглядит следующим образом:

$$\begin{aligned}
\mathcal{J} = \{&sum \mapsto \top, \quad inc \mapsto \top, \\
&\langle size,sum,sum,inc\rangle \mapsto \big(T^{size}{=}T^{sum_1}{=}T^{inc} \wedge \\
&\qquad T^{sum_2}{=}U^{inc_2} \to s^{sum_2}{=}s^{sum_1}{+}2n^{size}\big)\}.
\end{aligned}$$

Можно заметить, что реляционные сертификаты сертификаты выполнимости имеют определённое сходство с сертификатами выполнимости синхронизаций этих систем (см., например, пример 10 в разделе 2.1 и раздел 2.9). Как будет показано в разделе 3.5, это сходство не случайное.

3.4 Обсуждение

Отличие реляционных сертификатов от обычных проиллюстрировано на рис. 7. Рассмотрим для примера нелинейную систему дизъюнктов $\mathcal{S}$ над $\mathcal{R} = \{P,Q\}$ с единственным запросом, который выглядит следующим образом:

$$C \equiv \varphi(\overline{x},\overline{y}) \wedge P(\overline{x}) \wedge Q(\overline{y}) \to \bot.$$

Без потери общности можно считать, что все переменные в кортежах $\overline{x}$ и $\overline{y}$ попарно различны, иначе введём новые переменные и приравняем их к старым в φ. Представленный выше запрос можно переписать следующим образом:

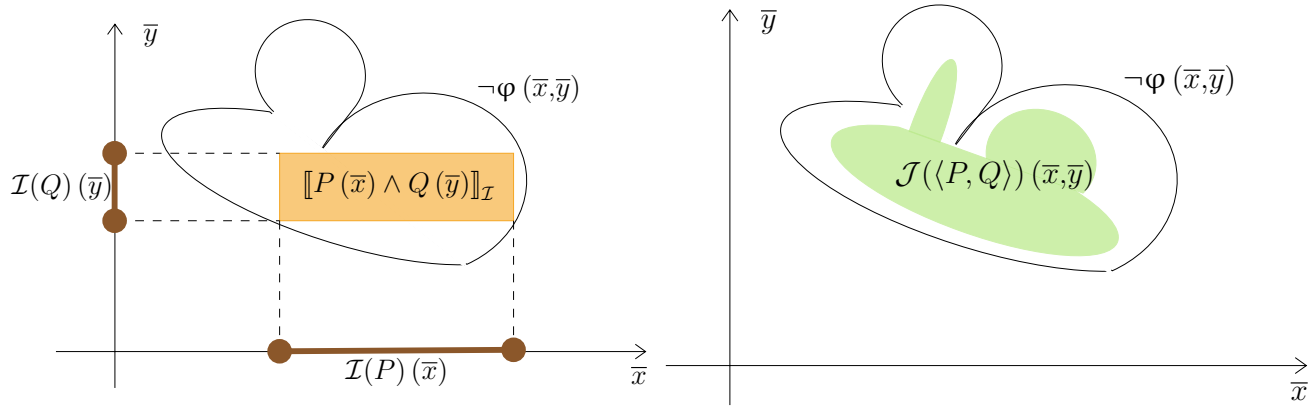
$$P(\overline{x}) \wedge Q(\overline{y}) \to \neg\varphi(\overline{x},\overline{y}).$$

Пусть у системы $\mathcal{S}$ существует сертификат выполнимости $\mathcal{I}$. Тогда для любой модели $\mathcal{M}$ теории $\mathcal{T}$ верно $\mathcal{M} \models \forall[\![C]\!]_{\mathcal{I}}$, другими словами, $\mathcal{M}\left([\![P(\overline{x}) \wedge Q(\overline{y})]\!]_{\mathcal{I}}\right) \subseteq \mathcal{M}(\neg\varphi)$. Но поскольку все переменные в кортежах $\overline{x}$ и $\overline{y}$ попарно различны, то справедливо следующее:

$$\mathcal{M}\left([\![P(\overline{x}) \wedge Q(\overline{y})]\!]_{\mathcal{I}}\right) = \mathcal{M}\left(\mathcal{I}(P)\right) \times \mathcal{M}\left(\mathcal{I}(Q)\right).$$

Таким образом, у такой нелинейной системы $\mathcal{S}$ может существовать сертификат выполнимости только в том случае, если для любой модели $\mathcal{M}$ теории $\mathcal{T}$ у $\mathcal{M}(\varphi)$ существует подмножество, являющееся декартовым произведением двух множеств X и Y, которые должны быть представимы в теории $\mathcal{T}$ и выполнять остальные правила системы (см. рис. 7а).

Напротив, понятие реляционная символьной интерпретации $\mathcal{J}$ позволяет интерпретировать $P(\overline{x}) \wedge Q(\overline{y})$ одной формулой, представляющей какое угодно отношение над $\overline{x}$ и $\overline{y}$ (см. рис. 7б).



а) Традиционные сертификаты

б) Реляционные сертификаты

Рис. 7. Реляционные и классические сертификаты выполнимости для системы $\mathcal{S}$.

В качестве ещё одного примера рассмотрим следующую систему:

$$\begin{aligned} x = 0 &\to P(x) \\ x' = x - 1 \wedge P(x') &\to P(x) \\ y = 0 &\to Q(y) \\ y' = y - 1 \wedge Q(y') &\to Q(y) \\ x \neq y \wedge P(x) \wedge Q(y) &\to \bot \end{aligned}$$

Любой «классический» сертификат выполнимости этой системы должен интерпретировать P такой формулой $\psi(x)$, что $\mathcal{N}(\psi)^2 \subseteq \{(x,x) \mid x \in \mathbb{Z}\}$, и при этом подстановка ψ вместо P выполняет оба правила P. Очевидно, что любая такая формула ψ будет равносильна формуле $x = c$ для некоторой константы $c \in \mathbb{N}$. Вместе с тем у системы существует следующий реляционный сертификат выполнимости:

$$\mathcal{J} = \{P \mapsto \top, Q \mapsto \top, \langle P,Q\rangle \mapsto x = y\}.$$

При этом он интерпретирует $P(x) \wedge Q(y)$ формулой $x = y$. Очевидно, в этом случае реляционные сертификаты дают большую гибкость, поскольку любой «классической» сертификат $\mathcal{I}$, сопоставляющий P формулу $x = c$, соответствует следующему более сильному реляционному сертификату:

$$\mathcal{J} = \{P \mapsto \top, Q \mapsto \top, \langle P,Q\rangle \mapsto x = y = c\}.$$

Итак, традиционные сертификаты недостаточно выразительны для нелинейных систем, т.к. представляют только *декартовы произведения индуктив-*

ных множеств. Реляционные сертификаты позволяют описывать *произвольные* индуктивные усиления реляционного свойства безопасности.

3.5 Корректность

Итак, было показано, что понятие реляционных сертификатов выполнимости обобщает понятие «классических» сертификатов выполнимости. Но действительно ли наличие реляционного сертификата выполнимости свидетельствует о выполнимости системы? В случае с «классическими» сертификатами этот факт был очевиден, поскольку можно было явно предъявить интерпретации неинтерпретированных символов (см. утверждение 1). В случае с реляционными сертификатами ситуация не столь проста, т.к. явное предъявление интерпретаций неинтерпретированных символов в любой модели теории $\mathcal{T}$ крайне затруднительно.

Цель данного раздела — показать, что наличие реляционного сертификата выполнимости действительно гарантирует выполнимость системы. Это будет сделано посредством применения уже доказанных в главе 2 результатов, конкретнее, через семантику неподвижной точки (см. разделы 2.4.3 и 2.5).

Теорема 3.5.1. *Если у системы дизъюнктов $\mathcal{S}$ существует реляционный сертификат выполнимости, то $\mathcal{S}$ выполнима.*

Доказательство. Пусть $\mathcal{J}$ — реляционный сертификат выполнимости. Возьмём произвольную модель $\mathcal{M}$ теории $\mathcal{T}$. Докажем теорему посредством построения вспомогательной системы $\mathcal{S}'$ и её «классического» сертификата выполнимости $\mathcal{I}$. Вспомним про множество символов $\mathcal{R}'$ и биекцию $G : \mathbb{N}^{\mathcal{R}} \to \mathcal{R}'$ из раздела 2.1. Рассмотрим частичное отображение $\mathcal{J}'$, сопоставляющее $r \in \mathbb{N}^{\mathcal{R}}$ атомарную формулу $R(\overline{v}_r)$, где $R = G(r)$.

Далее заметим, что определения (3.1) и (3.2) реляционной подстановки имеет смысл также и для $\mathcal{J}'$, т.е. для всех $r \in \mathbb{N}^{\mathcal{R}}$ осмысленна запись $[\![body(r)]\!]_{\mathcal{J}'}$. Более того, $[\![body(r)]\!]_{\mathcal{J}'}$ будет являться дизъюнкцией формул вида $\varphi \wedge R_1(\overline{y}_1) \wedge \cdots \wedge R_m(\overline{y}_m)$, т.е. осмысленно говорить о том, что $[\![body(r)]\!]_{\mathcal{J}'}$ само по себе является телом некоторого символа некоторой системы дизъюнктов над $\mathcal{R}'$.

Определим следующую систему дизъюнктов $\mathcal{S}'$. Для каждого мультимножества $r \in dom(\mathcal{J})$ и $R \stackrel{\text{def}}{=} G(r)$ положим $body(R) \stackrel{\text{def}}{=} [\![body(r)]\!]_{\mathcal{J}'}$; заголовком

каждого правила положим атомарную формулу $R(\overline{v}_r)$. Каждому запросу $C \in \mathcal{S}$ сопоставим запрос $C' \in \mathcal{S}'$ такой, что $body(C') = [\![body(C)]\!]_{\mathcal{J}'}$.

Например, для системы $\mathcal{S}$ и реляционной символьной интерпретации $\mathcal{J}$ в примере 12, система $\mathcal{S}'$ будет выглядеть следующим образом:

$$\begin{aligned}
\varphi_1 &\to P(x_1, x_2)\\
\varphi_2 \wedge P(x_1', x_2') \wedge P(x_1'', x_2'') &\to P(x_1, x_2)\\
\psi_1 &\to Q(y)\\
\psi_2 \wedge Q(y') &\to Q(y)\\
\varphi_1 \wedge \psi_1 &\to PQ(x_1,x_2,y)\\
\varphi_1 \wedge \psi_2 \wedge Q(y') &\to PQ(x_1,x_2,y)\\
\varphi_2 \wedge \psi_1 \wedge P(x_1', x_2') \wedge P(x_1'', x_2'') &\to PQ(x_1,x_2,y)\\
\varphi_2 \wedge \psi_2 \wedge P(x_1{}', x_2{}') \wedge P(x_1{}'', x_2{}'') \wedge Q(y') \wedge & \\
PQ(x_1{}',x_2{}',y) \wedge PQ(x_1{}'',x_2{}'',y) &\to PQ(x_1,x_2,y)\\
\varphi \wedge P(z_1,z_2) \wedge Q(z_3) \wedge PQ(z_1,z_2,z_3) &\to \bot
\end{aligned}$$

Определим символьную интерпретацию $\mathcal{I} : \mathcal{R}' \to \mathcal{A}$ так:

$$\mathcal{I}(R) \stackrel{\text{def}}{=} \begin{cases} \mathcal{J}(G^{-1}(r)), & \text{если } G^{-1}(R) \in dom(\mathcal{J}) \\ \top, & \text{иначе.} \end{cases}$$

Заметим, что по построению $\mathcal{J}'$ и $\mathcal{I}$ для всех $r \in dom(\mathcal{J})$ выполняется $\mathcal{J}(r) \equiv [\![\mathcal{J}'(r)]\!]_{\mathcal{I}}$. Следовательно, для любого дизъюнкта $C \in \mathcal{S}$ имеем $[\![C]\!]_{\mathcal{J}} \equiv [\![[\![C]\!]_{\mathcal{J}'}]\!]_{\mathcal{I}}$. В частности, для любого запроса $C \in \mathcal{S}$ и соответствующего ему $C' \in \mathcal{S}'$ верно $[\![C]\!]_{\mathcal{J}} \equiv [\![C']\!]_{\mathcal{I}}$. Из этого следует, что для любого $r \in dom(\mathcal{J})$ выполняется следующее:

$$[\![body(r)]\!]_{\mathcal{J}} \equiv [\![[\![body(r)]\!]_{\mathcal{J}'}]\!]_{\mathcal{I}} \equiv [\![body(R)]\!]_{\mathcal{I}}\,, \text{ где } R = G(r).$$

В силу безопасности и индуктивности $\mathcal{J}$ получаем, что символьная интерпретация $\mathcal{I}$ также безопасна и индуктивна в смысле определения 8 в разделе 1.2.2, т.е. $\mathcal{I}$ является сертификатом выполнимости $\mathcal{S}'$.

Пусть $\mathcal{R} = \{P_1, \ldots, P_n\}$, и пусть также $X_1 \subseteq |\mathcal{M}|^{ar(P_1)}, \ldots, X_n \subseteq |\mathcal{M}|^{ar(P_n)}$. Обогатим интерпретацию $\mathcal{M}$ сигнатуры Σ до интерпретации $\overline{\mathcal{M}}$ сигнатуры $\mathcal{M} \cup$

$\mathcal{R}'$, добавив следующие интерпретации символов из $\mathcal{R}'$:

$$\overline{\mathcal{M}}(R) \stackrel{\text{def}}{=} X_1^{r(P_1)} \times \ldots X_n^{r(P_n)}, \text{ где } r = G^{-1}(R). \tag{3.3}$$

Пусть $C_1, \ldots, C_k$ — какие угодно дизъюнкты с попарно различными переменными. Пусть $body(C_1) \wedge \cdots \wedge body(C_k) \equiv \varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m)$.

Вспомним про произведение атомарных формул $\prod_{i=1}^{k} R_i(\overline{x}_i)$, определённое в разделе 2.1. Заметим, что если кортеж $\langle R_1, \ldots, R_k \rangle$ соответствует мультимножеству $r \in \mathbb{N}^{\mathcal{R}}$, то $\mathcal{J}'(r) \equiv \prod_{i=1}^{k} R_i(\overline{v}_{R_i}^{(k)})$, а значит, справедливо следующее:

$$[\![body(C_1) \wedge \cdots \wedge body(C_k)]\!]_{\mathcal{J}'} \equiv \varphi \ \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k} \in \{R_1,\ldots,R_m\} \\ \langle R_{i_1},\ldots,R_{i_k} \rangle \in dom(\mathcal{J})}} \mathcal{J}'(\langle R_{i_1}, \ldots, R_{i_k} \rangle)[\langle \overline{x}_{i_1}, \ldots, \overline{x}_{i_k} \rangle / \overline{v}_{R_{i_1},\ldots,R_{i_k}}] \equiv$$

$$\equiv \varphi \ \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k} \in \{R_1,\ldots,R_m\} \\ \langle R_{i_1},\ldots,R_{i_k} \rangle \in dom(\mathcal{J})}} \prod_{j=1}^{k} R_{i_j}(\overline{x}_{i_j}).$$

По лемме 5 имеем следующее:

$$\overline{\mathcal{M}}\left([\![body(C_1) \wedge \cdots \wedge body(C_k)]\!]_{\mathcal{J}'}\right) = \overline{\mathcal{M}}\left(\varphi \ \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k} \in \{R_1,\ldots,R_m\} \\ \langle R_{i_1},\ldots,R_{i_k} \rangle \in dom(\mathcal{J})}} \bigwedge_{j=1}^{k} R_{i_j}(\overline{x}_{i_j})\right) =$$

$$= \overline{\mathcal{M}}\left(\varphi \wedge R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m)\right) = \overline{\mathcal{M}}\left(body(C_1) \wedge \cdots \wedge body(C_k)\right). \tag{3.4}$$

По лемме 6 справедливо следующее:

$$\overline{\mathcal{M}}\left([\![body(C_1) \wedge \cdots \wedge body(C_k)]\!]_{\mathcal{J}'}\right) = \overline{\mathcal{M}}\left(body(C_1 \times \cdots \times C_k)\right).$$

Далее вспомним, что для любого $R \in \mathcal{R}'$ запись $body(R)$ определена в контексте этого доказательства: $body(R) = [\![body(G^{-1}(R))]\!]_{\mathcal{J}'}$. Она также определена и в смысле (2.2) (см. раздел 2.1):

$$body(R) = \bigvee_{\langle C_1,\ldots,C_k \rangle \in rules^{(1)}(R_1) \times \ldots \times rules^{(k)}(R_k)} body(C_1 \times \cdots \times C_k), \text{ где } \langle R_1, \ldots, R_k \rangle \text{ соответствует } G^{-1}(R).$$

Для того, чтобы устранить двузначность, временно изменим нотацию: запись $body^{rel}(R)$ будет означать $body(R)$ в смысле этого доказательства, а $body^{prod}(R)$ будет означать $body(R)$ в смысле (2.2).

Пусть $r \in \mathbb{N}^{\mathcal{R}}$, $R = G(r)$ и кортеж $\langle R_1, \ldots, R_k \rangle$ соответствует мультимножеству r. Тогда по (3.2) и определению тела мультимножества символов имеем:

$$\overline{\mathcal{M}}\left(body^{rel}(R)\right) \stackrel{\text{def}}{=} \overline{\mathcal{M}}\left(\bigvee_{\substack{C_1 \in rules^{(1)}(R_1) \\ \ldots \\ C_k \in rules^{(k)}(R_k)}} [\![body(C_1) \wedge \cdots \wedge body(C_k)]\!]_{\mathcal{J}'}\right) =$$

$$= \overline{\mathcal{M}}\left(\bigvee_{\substack{C_1 \in rules^{(1)}(R_1) \\ \ldots \\ C_k \in rules^{(k)}(R_k)}} body(C_1 \times \cdots \times C_k)\right) = \overline{\mathcal{M}}\left(body^{prod}(R)\right).$$

Это означает, что несмотря на синтаксические различия в двух версиях определения $body(R)$, семантически эти определения эквивалентны. В частности, теорема 13 показывает, что для любого неинтерпретированного символа R системы $\mathcal{S}'$ верно

$$[\![R]\!]_{\mathcal{M}} = [\![R_1]\!]_{\mathcal{M}} \times \cdots \times [\![R_k]\!]_{\mathcal{M}}.$$

Итак, по имеющемуся реляционному сертификату $\mathcal{J}$ нам удалось построить систему $\mathcal{S}'$ вместе с «классическим» сертификатом выполнимости $\mathcal{I}$ такую, что семантика каждого символа этой системы есть произведение семантик символов изначальной системы $\mathcal{S}$.

Пусть теперь C — произвольный запрос системы $\mathcal{S}$. По построению $\mathcal{S}'$ для него имеется запрос $C' \in \mathcal{S}'$ такой, что $body(C') = [\![body(C)]\!]_{\mathcal{J}'}$. Из (3.4) следует, что для любого обогащения $\overline{\mathcal{M}}$ вида (3.3) выполнено следующее:

$$\overline{\mathcal{M}}\left(body(C')\right) = \overline{\mathcal{M}}\left([\![body(C)]\!]_{\mathcal{J}'}\right) = \overline{\mathcal{M}}\left(body(C)\right).$$

В частности, рассмотрим следующее обогащение $\mathcal{M}^{sem}$, верное для всех $R \in \mathcal{R}'$:

$$\mathcal{M}^{sem}(R) \stackrel{\text{def}}{=} [\![R]\!]_{\mathcal{M}} = \left([\![P_1]\!]_{\mathcal{M}}\right)^{r(P_1)} \times \ldots \left([\![P_n]\!]_{\mathcal{M}}\right)^{r(P_n)}, \text{ где } r = G^{-1}(R).$$

Тогда для $\mathcal{M}^{sem}$ выполнено следующее:

$$\mathcal{M}^{sem}\left(body(C')\right) = \mathcal{M}^{sem}\left(body(C)\right).$$

Заметим, что теорему 8 можно переформулировать в терминах структуры $\mathcal{M}^{sem}$ следующим образом: система дизъюнктов $\mathcal{S}$ выполнима тогда и только

тогда, когда для всякой модели $\mathcal{M}$ теории $\mathcal{T}$ и всякого запроса C справедливо $\mathcal{M}^{sem}(body(C)) = \varnothing$.

Поскольку $\mathcal{I}$ является сертификатом выполнимости для $\mathcal{S}'$, то система $\mathcal{S}'$ выполнима, а это значит, что к ней, а также к интерпретации $\mathcal{M}$ и к запросу C' можно применить теорему 8, т.е. справедливо следующее:

$$\mathcal{M}^{sem}(body(C')) = \varnothing.$$

Мы получили, что $\mathcal{M}^{sem}(body(C)) = \mathcal{M}^{sem}(body(C')) = \varnothing$, значит, по теореме 8, система $\mathcal{S}$ выполнима. □

Заметим, что в данном доказательстве формально отражена связь между главами 2 и 3: по любой реляционной символьной интерпретации $\mathcal{J}$ можно определить такое отображение $\mathcal{J}'$, что реляционная подстановка $\mathcal{J}'$ в любую систему $\mathcal{S}$ порождает систему $\mathcal{S}'$ с семантикой некоторого произведения правил системы $\mathcal{S}$. Причём если $\mathcal{J}$ — реляционный сертификат выполнимости $\mathcal{S}$, то для новой системы $\mathcal{S}'$ существует также и «классический» сертификат выполнимости $\mathcal{I}$.

В частности, это означает, что, с точки зрения проблемы непредставимости моделей, понятие реляционного сертификата выполнимости не даёт никакого «выигрыша» по сравнению с синхронизацией дизъюнктов, т.е. оба метода решают проблему непредставимости моделей вплоть до одной и той же степени: если у некоторой синхронизации системы $\mathcal{S}$ существует сертификат выполнимости, то у изначальной системы $\mathcal{S}$ будет существовать реляционный сертификат выполнимости. И наоборот, если у системы $\mathcal{S}$ существует реляционный сертификат выполнимости, то можно построить некоторую её синхронизацию $\mathcal{S}'$ с классическим сертификатом выполнимости.

Напоследок сформулируем вспомогательное следствие выкладок в доказательстве предыдущей теоремы, которое будет использовано в главе 4.

Лемма 12. Пусть $\mathcal{M}$ — интерпретация, $\mathcal{S}'$ — система дизъюнктов Хорна с ограничениями, $b > 0$ — натуральное число, $\mathcal{J}$ — такая реляционная символьная интерпретация, что для всех $\langle R_1, \ldots, R_m\rangle \in dom(\mathcal{J})$ справедливо следующее:

$$[\![R_1]\!]^{b-1}_{\mathcal{M}} \times \cdots \times [\![R_m]\!]^{b-1}_{\mathcal{M}} \subseteq \mathcal{M}\left(\mathcal{J}\left(\langle R_1, \ldots, R_m\rangle\right)\right).$$

Тогда для всех $\langle R_1, \ldots, R_m\rangle \in dom(\mathcal{J})$ выполнено:

$$[\![R_1]\!]^{b}_{\mathcal{M}} \times \cdots \times [\![R_m]\!]^{b}_{\mathcal{M}} \subseteq \mathcal{M}\left(\exists\overline{\ell}_{R_1,\ldots,R_m}.\,[\![body(R_1, \ldots, R_m)]\!]_{\mathcal{J}}\right).$$

Доказательство. Обогатим интерпретацию $\mathcal{M}$ сигнатуры Σ до интерпретации $\overline{\mathcal{M}}$ сигнатуры $\mathcal{M} \cup \mathcal{R}'$, добавив следующие интерпретации символов из $\mathcal{R}'$:

$$\overline{\mathcal{M}}(R) \stackrel{\text{def}}{=} \left(\llbracket P_1 \rrbracket_{\mathcal{M}}^{b-1}\right)^{r(P_1)} \times \ldots \left(\llbracket P_n \rrbracket_{\mathcal{M}}^{b-1}\right)^{r(P_n)}, \text{ где } r = G^{-1}(R).$$

Определим систему $\mathcal{S}'$ и «классическую» символьную интерпретацию $\mathcal{I} : \mathcal{R}' \to \mathcal{A}$ тем же образом, что в доказательстве предыдущей теоремы. Для этой системы было показано, что для всех $R \in \mathcal{R}'$ выполнено следующее:

$$\overline{\mathcal{M}}\left(\exists \bar{\ell}_R.body^{rel}(R)\right) = \overline{\mathcal{M}}\left(\exists \bar{\ell}_R.body^{prod}(R)\right).$$

Но в разделе 2.5 продемонстрировано, что $\overline{\mathcal{M}}\left(\exists \bar{\ell}_R.body^{prod}(R)\right) = \llbracket R \rrbracket_{\mathcal{M}}^{b} = \llbracket R_1 \rrbracket_{\mathcal{M}}^{b} \times \cdots \times \llbracket R_m \rrbracket_{\mathcal{M}}^{b}$, где $\langle R_1, \ldots, R_m \rangle = G^{-1}(R)$.

Определим другое обогащение $\overline{\mathcal{M}}_{\mathcal{J}}$ структуры $\mathcal{M}$, добавив следующие интерпретации символов $G(r)$ для всех $r \in dom(\mathcal{J})$:

$$\overline{\mathcal{M}}_{\mathcal{J}}(G(r)) \stackrel{\text{def}}{=} \mathcal{M}\left(\mathcal{J}\left(r\right)\right).$$

По условию леммы, для всех $r \in dom(\mathcal{J})$, если $R = G(r)$, то $\overline{\mathcal{M}}(R) \subseteq \overline{\mathcal{M}}_{\mathcal{J}}(R)$. Поскольку неинтерпретированные символы входят в тела символов лишь позитивно, то для всех $\langle R_1, \ldots, R_m \rangle \in dom(\mathcal{J})$ и $R = G(\langle R_1, \ldots, R_m \rangle)$ имеем:

$$\overline{\mathcal{M}}\left(\exists \bar{\ell}_R.body^{rel}(R)\right) \subseteq \overline{\mathcal{M}}_{\mathcal{J}}\left(\exists \bar{\ell}_R.body^{rel}(R)\right) = \mathcal{M}\left(\exists \bar{\ell}_{R_1,\ldots,R_m}.\, \llbracket body(R_1, \ldots, R_m) \rrbracket_{\mathcal{J}}\right).$$

□

3.6 Быстрое вычисление подстановки

Итак, выразительная сила реляционных сертификатов выполнимости совпадает с сертификатами выполнимости синхронизированных систем. Тем не менее понятие реляционного сертификата выполнимости является полезным с вычислительной точки зрения.

Как следует из (3.2), «наивная» реализация $\left[\!\left[\bigwedge_{i=1}^{k} \bigvee_{j=1}^{\ell_i} B_{i,j} \right]\!\right]_{\mathcal{J}}$ требует вычисления $\ell_1 \cdot \ldots \cdot \ell_k$ подстановок, а это произведение экспоненциально по k. То же самое верно и для синтаксической синхронизации дизъюнктов (см. (2.2)). В этом разделе демонстрируется альтернативный метод, позволяющий избежать явного перечисления всех комбинаций тел правил.

Ключевая идея заключается в построении *эквивыполнимой* формулы вместо «наивного» вычисления по правилам (3.1) и (3.2).

Пусть $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)$ — все реляционные атомы в формуле Φ. Для каждого атома $R_i(\overline{x}_i)$ заведём пропозициональный атом (т.е. нульместный предикатный символ) a_i. Пусть Φ' — формула, полученная заменой всех вхождений $R_i(\overline{x}_i)$ на a_i для всех i в Φ. Тогда очевидно, что формула $[\![\Phi]\!]_{\mathcal{J}}$ эквивыполнима следующей формуле:

$$\Phi' \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k} \in \{R_1,\ldots,R_m\} \\ \langle R_{i_1},\ldots,R_{i_k}\rangle \in dom(\mathcal{J})}} \big(a_{i_1} \wedge \ldots \wedge a_{i_k} \leftrightarrow \mathcal{J}(\langle R_{i_1},\ldots,R_{i_k}\rangle)[\langle \overline{x}_{i_1},\ldots,\overline{x}_{i_k}\rangle / \overline{v}_{R_{i_1},\ldots,R_{i_k}}]\big).$$

Очевидно, что размер этой формулы уже не экспоненциален по k.

Следует отметить, что описанная процедура похожа на преобразование Цейтина [133], где также строится эквивыполнимая формула с помощью введения новых пропозициональных атомов. Преобразование Цейтина используется для приведения формулы в конъюнктивную нормальную форму (КНФ). Также, как и в данном случае, «наивное» приведение формул в КНФ с использованием правила дистрибутивности порождает формулу экспоненциального размера, в то время как введение новых пропозициональных атомов, «помечающих» подформулы, позволяет избежать этого.

Пример 13. В примере 12 следующая формула эквивыполнима с $[\![body(P,Q)]\!]_{\mathcal{J}}$:

$$\big(\varphi_1 \vee (\varphi_2 \wedge a_1 \wedge a_2)\big) \wedge \big(\psi_1 \vee (\psi_2 \wedge a_3)\big) \wedge \big(a_3 \to \eta_1(y')\big) \wedge$$
$$\wedge \big(a_1 \wedge a_3 \to \eta_2(x'_1,x'_2,y')\big) \wedge \big(a_2 \wedge a_3 \to \eta_2(x''_1,x''_2,y')\big)$$

Данный метод используется в программной реализации алгоритма RelRecMc (см. главу 4). Он позволяет сохранить модульность процесса вывода реляционных лемм и значительно повышает скорость вычисления реляционной подстановки по сравнению с наивной реализацией.

Глава 4. Направляемый свойством вывод реляционных инвариантов

В данной главе описывается направляемый свойством алгоритм RelRecMc автоматического вывода реляционных инвариантов. Алгоритм RelRecMc обобщает алгоритм RecMC, подробно описанный в [6].

4.1 Используемые понятия

Для построения сертификатов выполнимости алгоритм RelRecMc использует две логические процедуры — *интерполяцию Крейга* и *проекцию на основе моделей*. Кратко опишем эти процедуры.

Интерполяция Крейга. Пусть A и B — формулы некоторого языка первого порядка, и формула $A \wedge B$ невыполнима. В этом случае будем называть A и B *несовместными. Интерполянтом Крейга* формул A и B называется формула I, удовлетворяющая трём условиям: 1) $A \vDash I$, 2) $B \vDash \neg I$ и 3) I состоит только из общих для A и B переменных, предикатных и функциональных символов.

Интерполяционная теорема Крейга утверждает, что каким бы ни был язык первого порядка, для любой пары несовместных формул существует Крейговский интерполянт [134]. Единственность интерполянта Крейга не гарантируется.

Говорят, что теория $\mathcal{T}$ *допускает бескванторную интерполяцию*, если для любых бескванторных формул φ и ψ таких, что $\varphi \wedge \psi$ невыполнимо в теории θ, существует бескванторная формула θ такая, что 1) $\mathcal{T} \models \forall(\varphi \rightarrow \theta)$, 2) $\mathcal{T} \models \forall(\psi \rightarrow \neg\theta)$ и 3) в θ входят только общие для φ и ψ переменные.

Будем говорить, что теория $\mathcal{T}$ *допускает устранение кванторов*, если для любой формулы $\varphi(\overline{x})$ существует бескванторная формула $\psi(\overline{x})$ такая, что $\mathcal{T} \models \forall\overline{x}.(\varphi \leftrightarrow \psi)$. К примеру, устранение кванторов допускает линейная целочисленная [135] и рациональная арифметика [53], вещественная арифметика [54], теория алгебраических типов данных [50] и фрагмент теории массивов [47]. Очевидно, что если теория допускает устранение кванторов, то она допускает бескванторные интерполянты.

Далее будем считать, что теория $\mathcal{T}$ в языке ограничений $\mathcal{A}$ допускает устранение кванторов. Таким образом, можно считать, что для любых двух

несовместных в $\mathcal{T}$ ограничений φ и ψ существует ограничение θ, интерполирующее φ и ψ. Этот факт будем обозначать следующим образом: $\theta \in \mathrm{ITP}(\varphi, \psi)$.

Проекция на основе модели. Если теория $\mathcal{T}$ допускает устранение кванторов, то в ней можно определить оператор *проекции на основе модели* (model-based projection, MBP) [6, 136, 137].

В этом разделе будет удобным полагать, что интерпретация также включает в себя оценку свободных переменных формулы, т.е. для формулы $\varphi(x_1, \ldots, x_n)$ имеет смысл запись $\mathcal{M} \models \varphi$.

Рассмотрим формулу $\exists \overline{x}.\tau$, где τ — бесквантворная формула. По данной формуле и её модели M оператор $\mathrm{MBP}\,(\tau, \overline{x}, M)$ возвращает бесквантворную конъюнкцию литералов τ' со следующими свойствами:

- $M \models \tau'$;
- $\mathcal{T} \models \forall(\tau' \to \exists \overline{x}.\tau)$;
- для каждой формулы τ существует конечное множество интерпретаций $M_1, \ldots, M_k$ таких, что $\mathcal{T} \models \forall(\exists \overline{x}.\tau \Leftrightarrow \bigvee_{i=1}^{k} \mathrm{MBP}\,(\tau, \overline{x}, M_i))$.

Последнее условие можно переформулировать следующим образом: если зафиксировать формулу τ и множество переменных $\overline{x}$, то образ отображения $\{M \mapsto \mathrm{MBP}\,(\tau, \overline{x}, M)\}$ конечен.

Неформально, последовательность проекций на основе моделей *лениво* устраняют квантор из $\exists \overline{x}.\tau$. Обычно устранение квантора происходит путём предъявления конечного числа формул $\tau_1, \ldots, \tau_m$ таких, что $\mathcal{T} \models \forall(\exists x.\tau \leftrightarrow \bigvee_i \tau_i)$. Работа оператора MBP может рассматриваться как выбор очередной τ_i, выполняющейся в заданной интерпретации M. При включении процедуры проекции на основе моделей в алгоритм, который итеративно перебирает различные модели некоторого семейства формул (такими, например, являются все алгоритмы семейства CEGAR, см. раздел 1.4), вместо того, чтобы сразу порождать все $\tau_1, \ldots, \tau_m$, количество которых может экспоненциально зависеть от размера формулы, часто бывает достаточным рассматривать в один момент лишь одну τ_i. «Ленивость» в устранении кванторов позволит сохранять размер символьных представлений достаточно небольшим и рассматривать только релевантные сценарии поведения системы.

Нормальная форма системы дизъюнктов. Для удобства будем предполагать, что алгоритм RelRecMc принимает на вход только системы, находящиеся в нормальной форме.

Определение 26. Будем говорить, что система дизъюнктов Хорна с ограничениями находится в *нормальной форме*, если выполнены условия, перечисленные ниже.

1. Для всех пар различных P,P' неинтерпретированных символов системы все переменные в $\overline{v}_P$ и $\overline{v}_{P'}$ попарно различны, а также попарно различны все переменные, входящие в $body(P)$ и $body(P')$.
2. Для каждого дизъюнкта C, верно, что переменные всех пар различных неинтерпретированных атомов в теле C попарно различны.
3. В системе есть единственный запрос вида $P_0 \to \bot$, где P_0 — нульместный неинтерпретированный символ.

Любую систему дизъюнктов Хорна можно привести в нормальную форму. Действительно, для обеспечения требования 1 достаточно переименовать переменные. Чтобы удовлетворить требованию 2, следует заменить переменные $\overline{x}_{old}$ каждой атомарной формулы на новые переменные $\overline{x}_{new}$ и добавить равенства вида $\overline{x}_{old} = \overline{x}_{new}$ в соответствующее ограничение дизъюнкта. Наконец, для обеспечения требования 3 можно добавить в $\mathcal{R}$ новый нульместный неинтерпретированный символ и заменить каждый запрос в $\mathcal{S}$, который по определению запроса имеет вид $\varphi \wedge R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m) \to \bot$, следующим правилом:

$$\varphi \wedge R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m) \to P_0.$$

Кроме того, в систему следует также добавить запрос $P_0 \to \bot$. Очевидно, что полученная система $\mathcal{S}'$ будет эквивыполнима исходной системе $\mathcal{S}$: любое дерево вывода T с корнем (C,φ) высоты b системы $\mathcal{S}$ соответствует дереву вывода высоты $b+1$ системы $\mathcal{S}'$ с корнем $(P_0 \to \bot, \varphi)$, к которому «подвешено» T.

4.2 Структуры данных алгоритма

Алгоритм RelRecMc использует две структуры данных ρ и σ, в которых он хранит *достижимые ветви* системы и *леммы* о системе соответственно[1]. Первая структура отображает $P \in \mathcal{R}$ и натуральное число b на множество

[1]Обозначения σ и ρ заимствованы из [6].

формул над $\overline{v}_P$, вторая отображает мультимножество $\overline{P} \in \mathbb{N}^{\mathcal{R}}$ и натуральное число b на множество формул над $\overline{v}_{\overline{P}}$.

Алгоритм использует ρ для построения деревьев гиперрезолютивного опровержения и σ для построения реляционных доказательств. При этом в ρ размещаются формулы, соответствующие различным узлам деревьев вывода системы. Важным инвариантом ρ является тот факт, что каждый элемент $\rho(P, b)$ аппроксимирует семантику P глубины b снизу, т.е. для каждой формулы $\delta \in \rho(P, b)$ для всех моделей $\mathcal{M}$ теории $\mathcal{T}$ выполняется $\mathcal{M}(\delta) \subseteq [\![P]\!]^b_{\mathcal{M}}$.

В структуре σ размещаются факты, *резюмирующие* систему, которые называются *леммами*. Формулы в $\sigma(r,b)$ аппроксимируют сверху семантику r глубины b (которая, как показано в разделе 2.5, равна произведению семантик элементов r). Структура σ используется для построения реляционного сертификата выполнимости.

Таким образом, структуры ρ и σ определяют для системы символьную интерпретацию U^b_ρ и реляционную символьную интерпретацию O^b_σ соответственно[2]:

$$U^b_\rho(P) \stackrel{\text{def}}{=} \bigvee \{\delta \in \rho(P,c) \mid c \leqslant b\},$$
$$O^b_\sigma(\overline{P}) \stackrel{\text{def}}{=} \bigwedge \left\{\delta \in \sigma(\overline{R},c) \mid \overline{R} \in dom(\sigma), \overline{R} \subseteq \overline{P}, c \geqslant b\right\}.$$

Выбор обозначений U^b_ρ и O^b_σ объясняется тем, что они аппроксимируют семантику системы глубины b снизу (underapproximate) и сверху (overapproximate) соответственно.

Будем сокращать $[\![\pi]\!]_{U^b_\rho}$ и $[\![\pi]\!]_{O^b_\sigma}$ до $[\![\pi]\!]^b_\rho$ и $[\![\pi]\!]^b_\sigma$ соответственно. Для удобства положим значения интерпретаций на уровне -1 $U^{-1}_\rho(P) \stackrel{\text{def}}{=} \bot$ и $O^{-1}_\sigma(r) \stackrel{\text{def}}{=} \bot$ для всех $P \in \mathcal{R}$ и $r \in \mathbb{N}^{\mathcal{R}}$.

4.3 Описание алгоритма RelRecMc

В листинге 3 представлен псевдокод процедуры RelRecMc, которая итеративно добавляет новые ветви в ρ и усиливает леммы σ до тех пор, пока либо ρ не засвидетельствует контрпример (строка 4), либо леммы в σ не станут индуктивными (строка 13).

Итерация номер b RelRecMc проверяет на наличие гиперрезолютивного опровержения высоты b. Если нарушения не достижимы (строки 6–12), то

[2]Конъюнкция пустого множества — $\top$, дизъюнкция — $\bot$.

Листинг 3: Псевдокод RelRecMc

Вход : Система дизъюнктов $\mathcal{S}$

Выход : $\langle$ВЫПОЛНИМА, реляционный сертификат$\rangle$/НЕВЫПОЛНИМА

```
1  b := 0; ρ := ∅; σ := ∅;
2  пока true
3      ⟨res, ρ, σ⟩ := RelBndSafety(S, b, ρ, σ);
4      если res = ДОСТИЖИМО то вернуть НЕВЫПОЛНИМА;
5      иначе
6          inductive := true;
7          для всех c ∈ [0..b]
8              для всех ⟨r, c⟩ ∈ dom(σ)
9                  для всех δ ∈ σ(r,c)
10                     если [[body(r)]]^c_σ ∧ ¬δ невыполнимо в T то
11                         σ := σ ∪ {⟨r, c+1⟩ ↦ δ};
12                     иначе inductive := false;
13         если inductive то вернуть ⟨ВЫПОЛНИМА, O^b_σ⟩;
14         b := b + 1;
```

неформально говоря, σ содержит леммы, доказывающие, что все гиперрезолютивные выводы системы высоты b не являются опровержениями.

Затем RelRecMc распространяет все индуктивные леммы из σ на уровень $b+1$ и итерируется, если их не достаточно для построения сертификата выполнимости системы в целом.

Алгоритм работает лишь с системами в нормальной форме.

4.4 Процедура RelBndSafety

Процедура RelBndSafety, псевдокод которой приведён в листинге 4, проверяет наличие гиперрезолютивных опровержений высоты, ограниченной сверху B.

Введём следующее обозначение:

$$[\![r]\!]^b_{\mathcal{M}} \stackrel{\text{def}}{=} ([\![P_1]\!]^b_{\mathcal{M}})^{r(P_1)} \times \cdots \times ([\![P_n]\!]^b_{\mathcal{M}})^{r(P_n)},$$

Листинг 4: Псевдокод RelBndSafety

Вход : Система дизъюнктов $\mathcal{S}$, уровень B, хранилища ρ, σ

Выход : $\langle$ДОСТИЖИМО/НЕДОСТИЖИМО$, \rho, \sigma\rangle$

Q := $\{\langle P_0, \top, B\rangle\}$;
пока $Q \neq \varnothing$
  выбрать $\langle r, \pi, b\rangle$ из Q с наименьшим b;
  если $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ *невыполнимо в* $\mathcal{T}$
    пусть $\psi \in \mathrm{ITP}([\![body(r)]\!]^{b-1}_{\sigma}, \pi)$;
    σ := $\sigma \cup \{\langle r, b\rangle \mapsto \psi\}$;
    Q := $Q \setminus \{\langle r', \eta, c\rangle \mid r \subseteq r', c \leqslant b, \psi \wedge \eta$ невыполнимо в $\mathcal{T}\}$;
  иначе
    A := $\{[\![R(\overline{x})]\!]^{b-1}_{\rho} \mid R(\overline{x})$ — атом в $body(r)\}$;
    пусть $sat_\rho \subseteq A$ — максимальное по включению, такое, что $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi \wedge sat_\rho$ выполнимо в $\mathcal{T}$;
    пусть $M \models [\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi \wedge sat_\rho$;
    ψ := π; $atoms$:= $\varnothing$;
    для всех P **в мультимножестве** r
      пусть $C \in rules(P)$ такой, что $M \models [\![body(C)]\!]^{b-1}_{\sigma}$;
      $\eta \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m)$:= $body(C)$;
      sat^P_ρ := $\{[\![R_1(\overline{x}_1)]\!]^{b-1}_{\rho}, \ldots, [\![R_m(\overline{x}_m)]\!]^{b-1}_{\rho}\} \cap sat_\rho$;
      ψ_P := MBP$\big(\eta \wedge sat^P_\rho, \overline{\ell}_P, M\big)$;
      ψ := $\psi \wedge \psi_P$;
      $atoms_P$:= $\{R_i(\overline{x}_i) \mid [\![R_i(\overline{x}_i)]\!]^{b-1}_{\rho} \notin sat_\rho\}$;
      $atoms$:= $atoms \cup atoms_P$;
      если $atoms_P = \varnothing$
        ρ := $\rho \cup \{\langle P, b\rangle \mapsto \psi_P\}$;
      иначе ψ_P := $\bot$;
    Q := $Q \setminus \{\langle r', \eta, c\rangle \mid r' \subseteq r, c \geqslant b, (\bigwedge^{+}_{P \text{ в } r'} \psi_P) \wedge \eta$ выполнимо в $\mathcal{T}\}$;
    $Groups$:= Partition$(atoms, \pi, \psi)$;
    для всех $\{R_{i_1}(\overline{x}_{i_1}), \ldots, R_{i_k}(\overline{x}_{i_k})\} \in Groups$
      $syms$:= $\langle R_{i_1}, \ldots, R_{i_k}\rangle$;
      $vars$:= $\overline{x}_{i_1} \ldots \overline{x}_{i_k}$;
      ψ' := $\psi \wedge \Big[\!\Big[\bigwedge_{\substack{R_j(\overline{x}_j) \in atoms \\ j \notin \{i_1, \ldots, i_k\}}} R_j(\overline{x}_j) \Big]\!\Big]^{b-1}_{\sigma}$;
      ψ' := MBP$\big(\psi', \overline{v}_r \cup \overline{\ell}_r \setminus vars, M\big)$;
      Q := $Q \cup \{\langle syms, \psi'[\overline{v}_{syms}/vars], b-1\rangle\}$;
если $[\![P_0]\!]^{B}_{\rho}$ *выполнимо в* $\mathcal{T}$ **вернуть** $\langle$ДОСТИЖИМО$, \rho, \sigma\rangle$;
иначе вернуть $\langle$НЕДОСТИЖИМО$, \rho, \sigma\rangle$;

где $r \in \mathbb{N}^{\mathcal{R}}$ — это мультимножество неинтерпретированных символов. Это обозначение согласуется с результатами раздела (2.5).

Процедура формулирует и решает серию *вопросов* $\langle r, \pi, b\rangle$, где $r \in \mathbb{N}^{\mathcal{R}}$, π — ограничение, $b \in \mathbb{N}$. Ответ на $\langle r, \pi, b\rangle$ состоит в проверке пустоты $[\![r]\!]^b_{\mathcal{M}} \cap \mathcal{M}(\pi)$.

Вопросы помещаются в *очередь вопросов* Q, которая в начале содержит только тройку $\langle P_0, \top, B\rangle$. Каждая итерация начинается с выбора вопроса с наименьшим b (строка 3), на который может быть дан ответ — положительный (строка 7) или отрицательный (строка 24), либо может породить дочерние вопросы (строка 31). Когда на все вопросы дан ответ, RelBndSafety возвращает результат `(НЕ-)ДОСТИЖИМО` (строка 33 или 32 соответственно).

Вывод лемм. Если леммы в σ достаточно сильны для доказательства недостижимости π (строка 4), RelBndSafety выводит новую лемму вычислением Крейговского интерполянта формул $[\![body(r)]\!]^{b-1}_{\sigma}$ и π. В результате новая лемма аппроксимирует сверху семантику P глубины b, доказывая отсутствие деревьев вывода глубины b с корнями, совместными с π. Важно, что новая лемма сформулирована только над переменными $\overline{v}_r$, выражая *отношения* между аргументами символов в r. Каждый вопрос $\langle r',\eta,c\rangle \in Q$, такой, что обновлённая σ доказывает невыполнимость η, считается отвеченным положительно и удаляется из очереди (в частности, $\langle r,\pi,b\rangle$).

Вывод ветвей. Если леммы недостаточно сильны для доказательства недостижимости π, то существует модель $M \models \pi$, получаемая процедурой RelBndSafety на строке 11. Эта модель соответствует *контрпримеру к индуктивности* в терминах IC3 [15]. Дальнейшие действия процедуры состоят либо в добавлении в ρ ветвей некоторого дерева вывода, корень которого выполняется в M, либо в добавлении новых лемм в σ, блокирующих M.

Вначале определяется максимально совместное с $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ множество ветвей sat_ρ уровня $b-1$ в теле r (строка 10). Эта операция эффективно выполняется MAX-SMT-решателями [138, 139]. Максимальная совместность означает, что в sat_ρ нельзя добавить ни одной формулы из A так, чтобы $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi \wedge sat_\rho$ осталось выполнимым в $\mathcal{T}$.

Для каждого P в r алгоритм находит дизъюнкт C такой, что $M \models [\![body(C)]\!]^{b-1}_{\sigma}$. Такой дизъюнкт гарантированно существует, т.к. $M \models [\![body(P)]\!]^{b-1}_{\sigma}$. Множество атомов в теле C делятся на атомы ветви которых не выполняются в M ($atoms_P$ в строке 19), и противоположные (sat^P_ρ в строке 16).

Нетрудно видеть, что условие $atoms_P = \varnothing$ в строке 21 выполняется тогда и только тогда, когда $M \models [\![body(C)]\!]^{b-1}_{\rho}$. Другими словами, результат замены каждого атома $R_i(\overline{x}_i)$ в теле C формулой $U^{b-1}_{\rho}(R_i)$ с переименованием переменных выполняется в M. Если в формулы $\rho(R_i,b-1)$ соответствуют корням деревьев вывода высоты не более $b-1$, то такая замена соответствует гиперрезольвенте ветвей в $\rho(R_i,b-1)$ с дизъюнктом C, т.е. $[\![body(C)]\!]^{b-1}_{\rho}$ соответствует корню некоторого дерева вывода P высоты не более b, который к тому же выполняется в M. Таким образом, алгоритм выводит новую ветвь в системе и добавляет её в ρ (строка 22). Неформально говоря, вместо исследования всех правил r процедура исследует за одну итерацию только одну ветвь ψ_P, выбираемую в направлении к свойству π. Для получения символьного представления ветви ψ_P алгоритм использует проекцию на основе модели (строка 17).

Каждый вопрос $\langle r', \eta, c\rangle \in Q$, в котором η совместно с обновлённым U^c_{ρ} в теории $\mathcal{T}$, считается отвеченным отрицательно и удаляется из Q (строка 24). Так как старая и новая интерпретации U^c_{ρ} отличаются лишь добавленными ветвями ψ_P, достаточно проверить лишь совместность с этими новыми ветвями.

Нетрудно видеть, что $\langle r, \pi, b\rangle$ удаляется из Q тогда и только тогда, когда $atoms_P = \varnothing$ для всех P в r, что равносильно выполнимости $[\![body(r)]\!]^{b-1}_{\rho}$ в M.

Генерация вопросов. Если ни $[\![body(r)]\!]^{b-1}_{\rho} \wedge \pi$ выполнимо, ни $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ невыполнимо, формулы в ρ на уровне $b-1$ слишком сильны для того, чтобы быть совместными с π, а леммы на уровне $b-1$ слишком слабы несовместности с π. В этом случае процедура порождает дочерние вопросы, ответы на которых изменят ρ и σ таким образом, что либо $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ перестанет выполняться в M, либо $[\![body(r)]\!]^{b-1}_{\rho} \wedge \pi$ начнёт выполняться в M.

К моменту попадания в строку 25 множество $atoms$ содержит атомы, ветви которых слишком сильны для того, чтобы доказать совместность с π, а леммы слишком слабы для доказательства несовместности с π. Последующие действия процедуры направлены на создание вопросов на уровне $b-1$ для различных мультимножеств неинтерпретированных символов в $atoms$.

Процедура RelBndSafety параметризована оракулом Partition (строка 25), который группирует входное множество атомов на множество подмножеств этих атомов $Groups$, где $\bigcup Groups = atoms$. В частности, если $atoms = \varnothing$, то $Groups = \varnothing$, и процедура уходит на следующую итерацию.

Для каждого подмножества формируется мультимножество неинтерпретированных символов $syms$ и множество кортежей переменных $vars$. Например, множество атомов $\{f(\overline{x}_1), f(\overline{x}_2), g(\overline{y})\}$ может быть поделено на мультимножества $syms \in \{\{f \mapsto 2\}, \{g \mapsto 1\}\}$ символов и кортежи переменных $vars \in \{\overline{x}_1\overline{x}_2, \overline{y}\}$. В результате $syms$ перебирает все мультимножества символов, которые будут исследованы во время ответов на дочерние вопросы.

Процедура RelBndSafety работоспособна при различных реализациях оракула Partition, но в данной главе предполагается, что Partition удовлетворяет следующим свойствам. Пусть $atoms = rec \uplus nrec$, где rec — множество атомов в $atoms$, рекурсивных хотя бы с одним из символов в r, а $nrec$ — оставшиеся элементы $atoms$. Пусть $Groups = \textsc{Partition}(atoms, \pi, \psi)$. Тогда $nrec \in Groups$, и для любого $g \in Groups \setminus \{nrec\}$ верно $g \subseteq rec$ и $|g| \leqslant |r|$. Другими словами, Partition разделяет множества атомов на рекурсивные и нерекурсивные, и если количество рекурсивных атомов превосходит мощность мультимножества r, то оно дополнительно разделяется на подмножества меньшего размера.

Ограничение на количество рекурсивных атомов в подмножествах необходимо для гарантии завершаемости RelBndSafety: это останавливает неограниченный рост размеров мультимножеств вопросов.

Разбиение рекурсивных атомов можно делать любым способом, однако в реализации, представленной в главе 5, Partition выбирает разбиение рекурсивных атомов, сохраняющее индуктивность максимального фрагмента свойства. Раздел 4.5 демонстрирует его работу на примере 1.

Для каждого мультимножества в $syms$ генерируется дочерний вопрос. Мультимножеству сопоставляется вопрос на предыдущем уровне $b - 1$, формула которого является конъюнкцией родительского свойства безопасности π (строка 12), ограничений и ветвей всех дизъюнктов, выполняющихся в M (строка 18) и лемм оставшихся мультимножеств в разбиении (строка 29). Неформально говоря, RelBndSafety усиливает свойство π информацией о достижимых ветвях и дочерних леммах, связанных с M. Далее, процедура проектирует полученное свойство, устраняя все переменные, кроме $vars$, используя проекцию на основе модели (строка 30), переименовывает переменные и помещает новый вопрос в очередь (строка 31). Дочерние вопросы формулируются над переменными $\overline{v}_{syms}$, для этого переменные переименовываются (строка 31). Аналогично

выводу ветвей, использование проекции на основе моделей поддерживает размер формул в вопросах небольшим.

4.5 Пример

В этом разделе показаны несколько итераций алгоритма для задачи в примере 1. Вначале система приводится в нормальную форму:

$$
\begin{aligned}
T_1 = \mathit{leaf} \wedge n = 0 &\rightarrow \mathit{size}(T_1, n)\\
T_1 = \mathit{node}(v_1, L_1, R_1) \wedge n = 1 + n^L + n^R \wedge & \\
\mathit{size}(L_1, n^L) \wedge \mathit{size}(R_1, n^R) &\rightarrow \mathit{size}(T_1, n)\\
T_2 = \mathit{leaf} \wedge s_2 = 0 &\rightarrow \mathit{sum}(T_2, s_2)\\
T_2 = \mathit{node}(v_2, L_2, R_2) \wedge s_2 = v_2 + s_2^L + s_2^R \wedge & \\
\mathit{sum}(L_2, s_2^L) \wedge \mathit{sum}(R_2, s_2^R) &\rightarrow \mathit{sum}(T_2, s_2)\\
T_4 = \mathit{leaf} \wedge U = \mathit{leaf} &\rightarrow \mathit{inc}(T_4, U)\\
T_4 = \mathit{node}(v_4, L_4, R_4) \wedge U = \mathit{node}(v_4 + 2, L', R') \wedge & \\
\mathit{inc}(L_4, L') \wedge \mathit{inc}(R_4, R') &\rightarrow \mathit{inc}(T_4, U)\\
s_0' \neq s_0 + 2n_0 \wedge A = T_0 \wedge B = T_0 \wedge C = T_0 \wedge D = E \wedge \mathit{size}(A, n_0) \wedge & \\
\mathit{sum}(B, s_0) \wedge \mathit{inc}(C, D) \wedge \mathit{sum}(E, s_0') &\rightarrow P_0\\
P_0 &\rightarrow \bot
\end{aligned}
$$

Для того, чтобы пропустить часть «рутинных» действий и сделать демонстрацию более интересной, будет считаться, что в ветвях системы на уровне 0 сразу зафиксированы ограниченные факты символов, т.е. что алгоритм RelRecMc запускается при

$$
\begin{aligned}
\rho(\mathit{size}, 0) &\equiv \{T_1 = \mathit{leaf} \wedge n = 0\},\\
\rho(\mathit{sum}, 0) &\equiv \{T_2 = \mathit{leaf} \wedge s_2 = 0\},\\
\rho(\mathit{inc}, 0) &\equiv \{T_4 = \mathit{leaf} \wedge U = \mathit{leaf}\}.
\end{aligned}
$$

Уровень 0. Алгоритм начинает с вызова процедуры RelBndSafety для уровня $B = 0$, который помещает вопрос $\langle P_0, \top, 0\rangle$ в Q. И $[\![\mathit{body}(P_0)]\!]_\rho^{-1}$, и $[\![\mathit{body}(P_0)]\!]_\sigma^{-1}$ состоят из единственной формулы $\bot$, поэтому процедура попадает на строку 6, где добавляется $\mathrm{ITP}(\bot, \top) = \bot$ в $\sigma(P_0, 0)$. RelBndSafety завершается с ре-

зультатом НЕДОСТИЖИМО, но т.к. добавленная лемма не индуктивна, RelRecMc продолжает свою работу на уровне 1.

Уровень 1. RelBndSafety начинает с $Q = \{\langle P_0, \top, 1\rangle\}$. Здесь $[\![body(P_0)]\!]^0_\rho \equiv \bot$, $[\![body(P_0)]\!]^0_\sigma \equiv \varphi_0 \stackrel{\text{def}}{=} s'_0 \neq s_0 + 2n_0 \wedge A = T_0 \wedge B = T_0 \wedge C = T_0 \wedge D = E$. Т.к. $\varphi_0 \wedge \top$ выполнима, процедура получает $M = \{A,B,C,D,E,T_0 \mapsto leaf; n_0 \mapsto 1; s_0 \mapsto 0; s'_0 \mapsto 1\}$ и переходит на строку 11.

Затем выбирается единственное возможное правило P_0 с телом $\varphi_0 \wedge size(A, n_0) \wedge sum(B, s_0) \wedge inc(C, D) \wedge sum(E, s'_0)$. Далее, $M \not\models [\![size(A, n_0)]\!]^0_\rho \equiv A = leaf \wedge n_0 = 0$, $M \models [\![sum(B, s_0)]\!]^0_\rho$, $M \models [\![inc(C, D)]\!]^0_\rho$, $M \not\models [\![sum(E, s'_0)]\!]^0_\rho$, откуда $atoms = \{size(A, n_0), sum(E, s'_0)\}$ и $\psi \equiv \varphi_0 \wedge B = leaf \wedge s_0 = 0 \wedge C = leaf \wedge D = leaf$. Т.к. и inc, и sum не рекурсивны с P_0, Partition возвращает $\text{Partition}(atoms) = \{atoms\}$.

Обозначим $\alpha_1 = \{size \mapsto 1, sum \mapsto 1\}$. Для получения вопроса для α_1 процедура строит проекцию ψ, устраняя все переменные, кроме A, n_0, E и s'_0, получая $\psi' \equiv \text{MBP}(\psi, \{T_0,B,C,D,s_0\}, M) \equiv A = leaf \wedge E = leaf \wedge s'_0 \neq 2n_0$, которая после переименования переменных становится $\psi_1 \stackrel{\text{def}}{=} T_1 = leaf \wedge T_2 = leaf \wedge s_2 \neq 2n$. Таким образом, Q становится $\{\langle P_0, \top, 1\rangle, \langle \alpha_1, \psi_1, 0\rangle\}$, и процедура итерируется.

На второй итерации процедура RelBndSafety выбирает из очереди $\langle \alpha_1, \psi_1, 0\rangle$ как вопрос с наименьшим уровнем. Пусть $\beta_1 \stackrel{\text{def}}{=} [\![body(\alpha_1)]\!]^{-1}_\rho \equiv [\![body(\alpha_1)]\!]^{-1}_\sigma \equiv T_1 = leaf \wedge n = 0 \wedge T_2 = leaf \wedge s_2 = 0$. Т.к. $\beta_1 \wedge \psi_1$ не выполнима, алгоритм выводит новую лемму $\delta_1 \stackrel{\text{def}}{=} \text{ITP}(\beta_1, \psi_1) \equiv T_1 = T_2 = leaf \wedge n = s_2 = 0$. Таким образом, $\sigma(\alpha_1, 0) = \{\delta_1\}$, вопрос на уровне 0 отвечен и убирается из Q.

На третьей итерации алгоритм снова выбирает $\langle P_0, \top, 1\rangle$. На этот раз $[\![body(P_0)]\!]^0_\sigma \equiv \varphi_0 \wedge \delta_1(A,n,B,s_0) \wedge \delta_1(A,n,E,s'_0)$. Т.к. теперь $[\![body(P_0)]\!]^0_\sigma$ не выполнима, в $\sigma(P_0, 1)$ добавляется $\text{ITP}([\![body(P_0)]\!]^0_\sigma, \top) \equiv \bot$. Новые леммы σ не индуктивны, поэтому RelRecMc переходит на уровень 2.

Уровень 2. Вопрос $\langle P_0, \top, 2\rangle$ выбирается из Q. На этот раз $M = \{A,B,C,T_0 \mapsto node(0, leaf, leaf); D,E \mapsto leaf; n_0, s_0, s'_0 \mapsto 1\}$. Т.к. $M \not\models [\![size(A, n_0)]\!]^1_\rho$, $M \not\models [\![sum(B, s_0)]\!]^1_\rho$, $M \not\models [\![inc(C, D)]\!]^1_\rho$, $M \not\models [\![sum(E, s'_0)]\!]^1_\rho$, получаем $atoms = \{size(A, n_0), sum(B, s_0), inc(C, D), sum(E, s'_0)\}$. Т.к. ни один из символов не рекурсивен с P_0, получаем $Groups = \{atoms\}$. Для получения дочернего свойства безопасности алгоритм устраняет T_0 проекцией и уходит на следующую

итерацию с $\{\langle P_0, \top, 2\rangle, \langle \alpha_2, \psi_2, 1\rangle\}$, где $\alpha_2 \stackrel{\text{def}}{=} \{size \mapsto 1, sum \mapsto 2, inc \mapsto 1\}$ и $\psi_2 \stackrel{\text{def}}{=} T_1 = T_2 = T_4 \wedge U = T_3 \wedge s_3 \neq s_2 + 2n$.

На следующей итерации $\langle \alpha_2, \psi_2, 1\rangle$ выбирается из Q. RelRecMc применяет подход, описанный в разделе 3.6 для эффективного вычисления подстановки σ.

Введем следующие обозначения.

$$\begin{aligned}
\beta_{size} \stackrel{\text{def}}{=} & (T_1 = leaf \wedge n = 0) \vee \\
& (T_1 = node(v_1, L_1, R_1) \wedge n = 1 + n^L + n^R \wedge a_L \wedge a_R) \\
\beta_{sum_1} \stackrel{\text{def}}{=} & (T_2 = leaf \wedge s_2 = 0) \vee \\
& (T_2 = node(v_2, L_2, R_2) \wedge s_2 = v_2 + s_2^L + s_2^R \wedge b_L \wedge b_R) \\
\beta_{sum_2} \stackrel{\text{def}}{=} & (T_3 = leaf \wedge s_3 = 0) \vee \\
& (T_3 = node(v_3, L_3, R_3) \wedge s_3 = v_3 + s_3^L + s_2^R \wedge c_L \wedge c_R) \\
\beta_{inc} \stackrel{\text{def}}{=} & (T_4 = leaf \wedge U = leaf) \vee \\
& (T_4 = node(v_4, L_4, R_4) \wedge U = node(v_4 + 2, L', R') \wedge d_L \wedge d_R)
\end{aligned}$$

Здесь a_L, a_R, b_L, b_R, c_L, c_R, d_L и d_R — пропозициональные абстракции неинтерпретированных атомов. Тогда справедливо следующее:

$$\begin{aligned}
[\![body(\alpha_2)]\!]^0_\sigma \equiv & \beta_{size} \wedge \beta_{sum_1} \wedge \beta_{sum_2} \wedge \beta_{inc} \wedge \\
& (a_L \wedge b_L \Rightarrow \delta_1(L_1, n_L, L_2, s_2^L)) \wedge \\
& (a_L \wedge b_R \Rightarrow \delta_1(L_1, n_L, R_2, s_2^R)) \wedge \\
& (a_L \wedge c_L \Rightarrow \delta_1(L_1, n_L, L_3, s_3^L)) \ldots
\end{aligned}$$

Если вместо этого прямолинейно перевести формулу $[\![body(\alpha_2)]\!]^0_\sigma$ в ДНФ и заменить всевозможные комбинации неинтерпретированных атомов формулой δ_1, мы получим формулу в 2^4 раз большего размера.

$[\![body(\alpha_2)]\!]^0_\sigma \wedge \psi_2$ выполнима с $M = \{T_k \mapsto node(1, leaf, leaf); T_3, U \mapsto node(3, leaf, l$ для $k \in \{1,2,4\}$. Таким образом, на строке 14 алгоритм выбирает по второму (рекурсивному) правилу для каждого отношения в $\mathcal{R}$.

Предположим теперь, что M не выполняет ни одну из ветвей в ρ ни одного атома в телах правил. Тогда получаем $atoms = \{size(L_1, n^L), size(R_1, n^R), sum(L_2, s_2^L), sum(R_2, s_2^R), sum(L_3, s_3^L), sum(R_3, s_3^R), inc(L_4, L'), inc(R_4, R')\}$ где все атомы рекурсивны, и $\psi \equiv \psi_2 \wedge T_1 = node(v_1, L_1, R_1) \wedge n = 1 + n^L + n^R \wedge \ldots$.

Если Partition вернёт $\{atoms\}$, то далее будет получен вопрос для мультимножества размера 8, что в результате породит вопрос для 16 отношений и т.д.; в результате, RelBndSafety разойдётся. Чтобы контролировать размер

мультимножеств, PARTITION группирует $atoms$ на множества размера, не превосходящего $|\alpha_2| = 4$. Однако существует $C_8^4 = 70$ вариантов уже для разбиения 8 отношений на два множества размера 4. Для выбора наилучшего разбиения, PARTITION применяет следующую эвристику.

Так как формула свойства безопасности в каждом вопросе в Q (кроме корневого) есть результат проекции, она представляет собой конъюнкцию литералов. Для каждого подмножество $atoms$, PARTITION определяет *максимальное индуктивное подмножество* литералов.

В данном случае множество литералов в ψ_2 является $\{T_1 = T_2, T_1 = T_4, U = T_3, s_3 \neq s_2 + 2n\}$. Например, литерал $T_1 \wedge T_2$ индуктивен относительно $size(R_1, n^R)$ и $sum(R_2, s_2^R)$. Чтобы убедиться в этом, переименуем $T_1 = T_2$ в $R_1 = R_2$ (т.к. T_1, R_1 и T_2, R_2 — первые аргументы соответствующих атомов) и заметим, что имеет место $\psi \Rightarrow R_1 = R_2$. В нашем случае, вся формула ψ_2 индуктивна относительно множеств $\{size(L_1, n^L), sum(L_2, s_2^L), sum(L_3, s_3^L), inc(L_4, L')\}$ и $\{size(R_1, n^R), sum(R_2, s_2^R), sum(R_3, s_3^R), inc(R_4, R')\}$, поэтому PARTITION возвращает $Groups = \{\{1,3,5,7\},\{2,4,6,8\}\}$. Для обоих множеств в Q добавляются вопросы α_2 уровня 0.

На следующих итерациях, алгоритм выводит лемму $T_1 = T_2 = T_4 \wedge U = T_4 \Rightarrow s_3 = s_2 + 2n$ и завершает построение реляционного сертификата выполнимости.

4.6 Свойства алгоритма

В этом разделе доказаны важные свойства RELRECMC и RELBNDSAFETY. Общая структура доказательств утверждений адаптирована из [6].

4.6.1 Корректность алгоритма RELRECMC

Покажем, что если алгоритм RELRECMC завершается, то он возвращает правильный ответ.

Вначале продемонстрируем важный инвариант алгоритма: U_ρ^b и O_σ^b аппроксимируют ограниченные семантики системы снизу и сверху соответственно.

Лемма 13. В процессе работы алгоритма RelRecMc для любой модели $\mathcal{M}$ теории $\mathcal{T}$ всегда выполены следующие утверждения:

$$\mathcal{M}(U^b_\rho(R)) \subseteq [\![R]\!]^b_{\mathcal{M}} \qquad \text{для всех } b \geqslant 0 \text{ и } R \in \mathcal{R}, \tag{4.1}$$

$$[\![R_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![R_m]\!]^b_{\mathcal{M}} \subseteq \mathcal{M}(O^b_\sigma(r)) \quad \text{для всех } b \geqslant 0 \text{ и } \langle R_1, \ldots, R_m\rangle \in dom(\sigma). \tag{4.2}$$

Доказательство. Пусть $\mathcal{M} \models \mathcal{T}$. Докажем утверждение индукцией по количеству итераций алгоритма.

Перед началом работы алгоритма $\rho(R,b) = \sigma(R,b) = \varnothing$, поэтому $U^b_\rho(R) = \bot$, $O^b_\sigma(R) = \top$, и утверждения леммы тривиально выполняются.

Обновления σ. Структура σ обновляется во время проверки индуктивности в RelRecMc (см. строку 11 листинга 3) и во время вывода новой леммы в RelBndSafety (см. строку 6 листинга 4). Рассмотрим подробно лишь обновление σ в строке 6; случай со строкой 11 аналогичен.

В строке 6 процедура RelBndSafety помещает в σ интерполянт $\psi \in \mathrm{ITP}([\![body(r)]\!]^{b-1}_\sigma, \pi)$, т.е. по определению интерполянта (см. раздел 4.1) $\mathcal{M} \models \forall([\![body(r)]\!]^{b-1}_\sigma \to \psi)$. Заметим, что для любого вопроса $\langle r, \pi, b\rangle \in Q$ верно следующее утверждение: все свободные переменные π являются переменными $\overline{v}_r$. Действительно, это верно для первого вопроса (строка 1), а также это верно для каждого добавляемого далее в Q вопроса (строка 31). Следовательно, среди свободных переменных π нет переменных из $\overline{\ell}_r$, а значит $\mathcal{M} \models \forall(\exists\overline{\ell}_r.\, [\![body(r)]\!]^{b-1}_\sigma \to \psi)$, и следовательно, справедливо следующее:

$$\mathcal{M}(\exists\overline{\ell}_r.\, [\![body(r)]\!]^{b-1}_\sigma) \subseteq \mathcal{M}(\psi). \tag{4.3}$$

Пусть кортеж $\langle R_1, \ldots, R_m\rangle$ соответствует мультимножеству r. Рассмотрим два случая: $b = 0$ и $b > 0$. Если $b = 0$, то образы $O^{b-1}_\sigma(s) = \bot$ для всех $s \in \mathbb{N}^{\mathcal{R}}$, значит, верно следующее:

$$[\![\exists\overline{\ell}_r.body(r)]\!]^{b-1}_\sigma \sim \left[\!\!\left[\exists\overline{\ell}_{R_1}.body(R_1) \overset{+}{\wedge} \ldots \overset{+}{\wedge} \exists\overline{\ell}_{R_m}.body(R_m)\right]\!\!\right]^{b-1}_\sigma \sim$$

$$\sim \Big(\bigvee_{\substack{C \in rules(R_1)\\ C \text{ — ограниченный факт}}} \exists\overline{\ell}_{R_1}.body(C)\Big) \overset{+}{\wedge} \ldots \overset{+}{\wedge} \Big(\bigvee_{\substack{C \in rules(R_m)\\ C \text{ — ограниченный факт}}} \exists\overline{\ell}_{R_m}.body(C)\Big).$$

Отсюда следует истинность утверждения ниже:

$$\mathcal{M}\left(\llbracket body(r)\rrbracket_{\sigma}^{b-1}\right) = \Big(\bigcup_{\substack{C\in rules(R_1)\\ C\text{ — ограниченный факт}}} \mathcal{M}\big(\exists\overline{\ell}_{R_1}.body(C)\big)\Big) \times \cdots \times \Big(\bigcup_{\substack{C\in rules(R_m)\\ C\text{ — ограниченный факт}}} \mathcal{M}\big(\exists\overline{\ell}_{R_m}.body(C)\big)\Big).$$

Но ограниченные факты — единственные возможные дизъюнкты в корнях деревьев вывода высоты 0, т.е. по теореме 9,

$$\mathcal{M}\left(\llbracket body(r)\rrbracket_{\sigma}^{b-1}\right) = \llbracket R_1\rrbracket_{\mathcal{M}}^{0} \times \cdots \times \llbracket R_m\rrbracket_{\mathcal{M}}^{0}.$$

Наконец, по (4.3) имеем:

$$\llbracket R_1\rrbracket_{\mathcal{M}}^{0} \times \cdots \times \llbracket R_m\rrbracket_{\mathcal{M}}^{0} \subseteq \mathcal{M}(\psi).$$

Пусть теперь $b-1 \geqslant 0$. По индукционному предположению, (4.2) выполняется для σ перед обновлением в строке 6. В частности, для всех $\langle R'_1 \ldots, R'_{m'}\rangle \in dom(\sigma)$ имеем $\llbracket R'_1\rrbracket_{\mathcal{M}}^{b-1} \times \cdots \times \llbracket R'_{m'}\rrbracket_{\mathcal{M}}^{b-1} \subseteq \mathcal{M}\left(O_{\sigma}^{b-1}(\langle R'_1 \ldots, R'_{m'}\rangle)\right)$.

Далее, по лемме 12 справедливо следующее:

$$\llbracket R_1\rrbracket_{\mathcal{M}}^{b} \times \cdots \times \llbracket R_m\rrbracket_{\mathcal{M}}^{b} \subseteq \mathcal{M}(\exists\overline{\ell}_r.\, \llbracket body(r)\rrbracket_{O_{\sigma}^{b-1}}) = \mathcal{M}(\exists\overline{\ell}_r.\, \llbracket body(r)\rrbracket_{\sigma}^{b-1}) \overset{(4.3)}{\subseteq} \mathcal{M}(\psi).$$

Итак, и при $b = 0$, и при $b > 0$ имеем:

$$\llbracket R_1\rrbracket_{\mathcal{M}}^{b} \times \cdots \times \llbracket R_m\rrbracket_{\mathcal{M}}^{b} \subseteq \mathcal{M}(\psi). \tag{4.4}$$

В строке 6 структура σ обновляется так: $\sigma' = \sigma \cup \{\langle r,b\rangle \mapsto \psi\}$. После этого обновления реляционные символьные интерпретации $O_{\sigma'}^{b}$ отличаются от O_{σ}^{b} лишь в точках $\langle r',b'\rangle$, таких, что $r \subseteq r'$ и $0 \leqslant b' \leqslant b$, причём $O_{\sigma'}^{b'}(r') = O_{\sigma}^{b'}(r') \wedge \psi$.

Возьмём произвольный r', такой, что $r \subseteq r'$ и $b' \in [0..b]$. Пусть $m' = |r'|$, и пусть кортеж $\langle R_1, \ldots, R_m, R_{m+1}, \ldots, R_{m'}\rangle$ соответствует r' (напомним, кортеж $\langle R_1, \ldots, R_m\rangle$ соответствует r). В ψ не входят свободные переменные из $\overline{v}_{r'} \setminus \overline{v}_r$, т.е. $\mathcal{M}\left(O_{\sigma'}^{b'}(r')\right) = \mathcal{M}\left(O_{\sigma}^{b'}(r')\right) \cap \left(\mathcal{M}(\psi) \times |\mathcal{M}|^{ar(R_{m+1})} \times \cdots \times |\mathcal{M}|^{ar(R_{m'})}\right)$. По индукционному предположению имеем:

$$\llbracket R_1\rrbracket_{\mathcal{M}}^{b'} \times \cdots \times \llbracket R_{m'}\rrbracket_{\mathcal{M}}^{b'} \subseteq \mathcal{M}(O_{\sigma}^{b'}(r')).$$

В силу (4.4) выполнено следующее:

$$\llbracket R_1\rrbracket_{\mathcal{M}}^{b'} \times \cdots \times \llbracket R_m\rrbracket_{\mathcal{M}}^{b'} \times \llbracket R_{m+1}\rrbracket_{\mathcal{M}}^{b'} \times \cdots \times \llbracket R_{m'}\rrbracket_{\mathcal{M}}^{b'} \subseteq$$
$$\subseteq \mathcal{M}(\psi) \times |\mathcal{M}|^{ar(R_{m+1})} \times \cdots \times |\mathcal{M}|^{ar(R_{m'})}.$$

Таким образом, имеем $\llbracket R_1\rrbracket_{\mathcal{M}}^{b} \times \cdots \times \llbracket R_{m'}\rrbracket_{\mathcal{M}}^{b} \subseteq O_{\sigma'}^{b'}(r')$, что и требовалось доказать.

Обновления ρ. Обновление структуры ρ производится лишь в строке 22 процедуры RelBndSafety.

В строке 22 производится следующее действие: $atoms_P = \varnothing$ (условие в строке 21), следовательно, имеем, $\eta \wedge sat^P_\rho \equiv [\![body(C)]\!]^{b-1}_\rho$ т.е. в структуру ρ помещается результат проекции $\psi_P = \text{MBP}\left([\![body(C)]\!]^{b-1}_\rho, \bar{\ell}_P, M\right)$, где $C \in rules(P)$. По определению проекции на основе модели (см. раздел 4.1), $\mathcal{M} \models \forall(\psi_P \to \exists\bar{\ell}_P.\, [\![body(C)]\!]^{b-1}_\rho)$. Так как $body(P) = \bigvee_{C\in rules(P)} body(C)$, то имеем $\mathcal{M} \models \forall(\exists\bar{\ell}_P.\, [\![body(C)]\!]^{b-1}_\sigma \to \exists\bar{\ell}_P.\, [\![body(P)]\!]^{b-1}_\sigma)$. Таким образом, выполнено следующее:

$$\mathcal{M}(\psi_P) \subseteq \mathcal{M}(\exists\bar{\ell}_P.\, [\![body(P)]\!]^{b-1}_\sigma). \tag{4.5}$$

При $b = 0$, по определению, $U^{-1}_\rho(P) = \bot$ для всех b и $P \in \mathcal{R}$, следовательно, как показано в доказательстве для σ, выполнено следующее утверждение:

$$[\![P]\!]^0_\mathcal{M} = \mathcal{M}(\exists\bar{\ell}_P.\, [\![body(P)]\!]^{-1}_\rho) \supseteq \mathcal{M}(\psi_P).$$

Если $b > 0$, то по индукционному предположению (4.1) выполняется для ρ перед обновлением в строке 22. В частности, для всех $P' \in \mathcal{R}$ имеем $\mathcal{M}(U^{b-1}_\rho(P')) \subseteq [\![P']\!]^{b-1}_\mathcal{M}$.

Пусть T — отношение переходов P.

$$[\![P]\!]^b_\mathcal{M} = T(\left\langle [\![P_1]\!]^{b-1}_\mathcal{M}, \ldots, [\![P_n]\!]^{b-1}_\mathcal{M} \right\rangle) \overset{\text{монот. } T}{\supseteq} T(\langle \mathcal{M}(U^{b-1}_\rho(P_1)), \ldots, \mathcal{M}(U^{b-1}_\rho(P_n)) \rangle) \overset{\text{лем. 4}}{=}$$
$$= \mathcal{M}(\exists\bar{\ell}_P.\, [\![body(P)]\!]_{U^{b-1}_\rho}) = \mathcal{M}(\exists\bar{\ell}_P.\, [\![body(P)]\!]^{b-1}_\rho) \overset{(4.5)}{\supseteq} \mathcal{M}(\psi_P).$$

Таким образом, ρ обновляется так: $\rho' = \rho \cup \{\langle P,b\rangle \mapsto \psi_P\}$. При этом $\mathcal{M}(\psi_P) \subseteq [\![P]\!]^b_\mathcal{M}$. Следовательно, по определению U имеем $U^{b'}_{\rho'}(P') = U^{b'}_\rho(P')$ для всех $P' \neq P$ и для $P' = P$ и $b' < b$. Для $b' \geqslant b$ имеем $U^{b'}_{\rho'}(P) = U^{b'}_\rho(P) \vee \psi_P$. Но по индукционному предположению $\mathcal{M}\left(U^{b'}_\rho(P)\right) \subseteq [\![P]\!]^{b'}_\mathcal{M}$, а потому из $\mathcal{M}(\psi_P) \subseteq [\![P]\!]^b_\mathcal{M}$ имеем:

$$[\![P]\!]^{b'}_\mathcal{M} \supseteq \mathcal{M}\left(U^{b'}_\rho(P)\right) \cup \mathcal{M}(\psi_P) = \mathcal{M}\left(U^{b'}_{\rho'}(P)\right).$$

□

Теперь сформулируем свойства для структур σ и ρ, истинные сразу после ответов на вопросы.

Лемма 14. Любой вопрос удаляется процедурой RelBndSafetyиз Q либо в строке 7, либо в строке 24 . При этом после удаления вопроса $\langle r, \eta, b\rangle$ из Q в строке 7 формула $O^b_\sigma(r) \wedge \eta$ является невыполнимой в $\mathcal{T}$, а если удаление происходит в строке 24, то формула $\overset{+}{\bigwedge}_{P \text{ в } r} U^b_\rho(P) \wedge \eta$ выполнима в $\mathcal{T}$.

Доказательство. Оба утверждения очевидным образом следуют из условия удаления вопросов и того факта, что ψ_P и ψ добавлены к моменту удаления в ρ и σ соответственно. □

Теперь можно доказать корректность алгоритма RelRecMc. Для начала покажем, что процедура RelBndSafety всегда возвращает верный ответ.

Теорема 23. Для системы $\mathcal{S}$ процедура RelBndSafety возвращает на уровне B значение `НЕДОСТИЖИМО` тогда и только тогда, когда у $\mathcal{S}$ не существует гиперрезолютивных опровержений высоты не более чем B.

Доказательство. По лемме 13 для любой модели $\mathcal{M}$ теории $\mathcal{T}$ верно следующее утверждение:

$$\mathcal{M}\left(\llbracket P_0 \rrbracket^B_\rho\right) \subseteq \llbracket P_0 \rrbracket^B_{\mathcal{M}} \subseteq \mathcal{M}\left(\llbracket P_0 \rrbracket^B_\sigma\right).$$

Далее, поскольку RelBndSafety работает лишь с системами в нормальной форме, у которых имеется единственный запрос в виде дизъюнкта $P_0 \to \bot$, по теореме 9 у $\mathcal{S}$ существует резолютивное опровержение глубины не более, чем B, тогда и только тогда, когда $\llbracket P_0 \rrbracket^B_{\mathcal{M}} \neq \varnothing$ для некоторой модели $\mathcal{M}$ теории $\mathcal{T}$.

Процедура возвращает значение `ДОСТИЖИМО`, только если $\llbracket P_0 \rrbracket^B_\rho$ выполнима в $\mathcal{T}$ (см. строку 32 листинга 4). При этом Q пуста, т.е. на все вопросы получен ответ, в частности, на первый вопрос $\langle P_0, \top, B\rangle$. При этом из-за того, что вопросы обрабатываются в порядке возрастания уровня (строка 3), а уровни новых вопросов, добавляемых в очередь лишь убывают (строка 31), вопрос $\langle P_0, \top, B\rangle$ будет отвечен последним.

По лемме 14 алгоритм возвращает значение `ДОСТИЖИМО` тогда и только тогда, когда $\llbracket P_0 \rrbracket^B_\rho$ выполнимо в $\mathcal{T}$, и `НЕДОСТИЖИМО` тогда и только тогда, когда $\llbracket P_0 \rrbracket^B_\sigma$ невыполнимо в $\mathcal{T}$. В первом случае существует модель $\mathcal{M}$ теории $\mathcal{T}$, в которой $\mathcal{M}(\llbracket P_0 \rrbracket^B_\rho) \neq \varnothing$, а значит, $\llbracket P_0 \rrbracket^B_{\mathcal{M}} \neq \varnothing$. Во втором случае для всех моделей $\mathcal{M}$ теории $\mathcal{T}$ имеем $\llbracket P_0 \rrbracket^B_\sigma = \varnothing$, а значит $\llbracket P_0 \rrbracket^B_{\mathcal{M}} = \varnothing$. □

Теорема 24. Алгоритм RelRecMc корректен, т.е. если он останавливается для системы $\mathcal{S}$, то она выполнима тогда и только тогда, когда алгоритм вернул значение ВЫПОЛНИМА. Более того, в случае ответа ВЫПОЛНИМА O^b_σ представляет собой реляционный сертификат выполнимости системы.

Доказательство. Алгоритм RelRecMc возвращает значение НЕВЫПОЛНИМА тогда и только тогда, когда процедура RelBndSafety (строка 3 листинга 3) возвращает ДОСТИЖИМО (строка 4). По предыдущей теореме, результат ДОСТИЖИМО может быть получен тогда и только тогда, когда у системы существует гиперрезолютивное опровержение глубины b. Но алгоритм итеративно увеличивает b (строка 14) от нуля (строка 1). Следовательно, если входная система невыполнима, то по полноте исчисления гиперрезолюций у неё существует гиперрезолютивное опровержение, а потому рано или поздно алгоритм вернёт НЕВЫПОЛНИМА. Осталось показать, что в случае результата ВЫПОЛНИМА O^b_σ является реляционным сертификатом выполнимости.

Заметим, что ВЫПОЛНИМА возвращается в случае, если для всех $\langle r,b\rangle \in dom(\sigma)$ и для всех $\delta \in \sigma(r,b)$ верно $\mathcal{T} \models \forall([\![body(r)]\!]^b_\sigma \to \delta)$, а значит и $\mathcal{T} \models \forall([\![body(r)]\!]_{O^b_\sigma} \to O^b_\sigma(r)$, т.е. реляционная символьная интерпретация O^b_σ индуктивна (в смысле определения 25). Более того, единственным запросом системы является $P_0 \to \bot$, и как было показано в доказательстве предыдущей теоремы, RelBndSafety возвращает НЕДОСТИЖИМО, только если $O^b_\sigma(M) \sim \bot$. Значит, $\mathcal{T} \models O^b_\sigma(M) \to \bot$, т.е. O^b_σ безопасна. □

4.6.2 Завершаемость RelBndSafety и RelRecMc

Целью данного раздела является доказательство завершаемости процедуры RelBndSafety.

Сначала сформулируем ещё один важный инвариант для данной процедуры. Неформально говоря, лемма 14 показывает, что после ответа на вопрос $\langle r,\pi,b\rangle$ в $\rho(r,b)$ и $\sigma(r,b)$ достаточно информации для ответа на этот вопрос. Теперь покажем, что имеет место и обратное утверждение, т.е. что в любой момент работы RelBndSafety в ρ и σ недостаточно информации для ответа на открытые вопросы.

Сделаем важную оговорку: в условии лемм данного раздела будем неявно предполагать, что в момент вызова RelBndSafety$(\mathcal{S}, B, \rho, \sigma)$ истинно следу-

ющее утверждение:

$$\sigma(P_0, B) = \varnothing. \tag{4.6}$$

Это условие может нарушаться после переноса лемм в RelRecMc в строке 11. Однако как было показано в доказательстве теоремы 24, в $\sigma(P_0, b)$ может содержать разве что лемму $\bot$. При этом т.к. уровень B строго возрастает в итерациях RelRecMc, необходимым и достаточным условием $\sigma(P_0, B) = \varnothing$ является $[\![body(M)]\!]_\sigma^{B-1} \wedge \neg\delta$ невыполнимо в $\mathcal{T}$ (т.е. условие в строке 10 листинга 3). Нетрудно видеть, что это же является и условием в строке 4 RelBndSafety, т.е. если RelBndSafety ответит на первый вопрос $\langle M,\top,B\rangle$ на первой же итерации в строке 7 и завершится.

Таким образом, если условие (4.6) нарушается, то RelBndSafety сделает ровно одну итерацию. Так как текущей целью является завершаемость RelBndSafety, в посылках дальнейших лемм можно неявно предполагать (4.6).

Лемма 15. Если $\sigma(P_0, B) = \varnothing$, то на любом шаге процедуры RelBndSafety$(\mathcal{S}, B, \sigma, \rho)$ для всех вопросов $\langle r,\pi,b\rangle \in Q$ верно следующее:

- $O_\sigma^b(r) \wedge \eta$ выполнима в $\mathcal{T}$;
- $\overset{+}{\bigwedge}_{P \text{ в } r} U_\rho^b(P) \wedge \eta$ невыполнима в $\mathcal{T}$.

Доказательство. Докажем утверждение индукцией по числу итераций процедуры RelBndSafety.

По условию (4.6) $O_\sigma^B(P_0) = \top$ и $U_\rho^B(P_0) = \bot$, т.е. утверждение леммы выполняется для первого вопроса $\langle P_0,\top,B\rangle$.

Теперь предположим, что в начале некоторой итерации выполняется утверждение леммы. Докажем, что оно выполняется и в конце итерации.

При обновлении σ на уровне b в строках 6–7 листинга 4, по лемме 14, все вопросы, не удовлетворяющие утверждению на уровнях $c \leqslant b$, удаляются из Q (строка 7), т.е. утверждение леммы выполняется для всех вопросов на уровне $c \leqslant b$. При этом по определению O_σ^c не меняется на уровнях $c > b$, а потому по индукционному предположению утверждение леммы верно для оставшихся вопросов в Q.

Аналогично, по лемме 14, после обновления ρ на уровне b (строка 22) все вопросы, не удовлетворяющие утверждению леммы на уровнях $c \geqslant b$, удаляются из Q (строка 7), т.е. утверждение леммы выполняется для всех вопросов на

уровне $c \leqslant b$; для остальных уровней $c < b$ U^c_ρ не меняется, т.е. утверждение леммы сохраняется и при обновлении ρ.

Осталось показать, что утверждение теоремы выполняется при добавлении новых вопросов в строке 31.

Пусть $C_1, \ldots, C_{|r|}$ — дизъюнкты, получаемые на различных итерациях в строке 14. Пусть η_i — ограничение дизъюнкта C_i, $\eta \stackrel{\text{def}}{=} \eta_1 \stackrel{+}{\wedge} \ldots \stackrel{+}{\wedge} \eta_{|r|}$, и $body(C_1) \wedge \cdots \wedge body(C_{|r|}) \equiv \eta \wedge R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m)$.

Поскольку $M \models [\![body(r)]\!]^{b-1}_\sigma \wedge \pi$ (см. строку 11), а также в силу выбора дизъюнктов, имеем следующее:

$$M \models \eta \wedge [\![R_1(\overline{x}_1) \wedge \cdots \wedge R_m(\overline{x}_m)]\!]^{b-1}_\sigma \wedge \pi. \tag{4.7}$$

Далее, если процедура попадает в строку 29, то уже сформированы непустое множество $atoms$ и формула ψ. Пусть $atoms_\rho \stackrel{\text{def}}{=} \{R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)\} \setminus atoms$. При этом $a \in atoms_\rho$ тогда и только тогда, когда $M \models [\![a]\!]^{b-1}_\rho$ (см. строку 19). Легко увидеть, что справедливо следующее утверждение:

$$\psi \equiv \pi \wedge \eta \wedge \bigwedge_{a \in atoms_\rho} [\![a]\!]^{b-1}_\rho .$$

Поскольку каждый элемент этой конъюнкции выполняется в M, то имеем $M \models \psi$.

Сделаем небольшое отсутпление в доказательстве для того, чтобы показать одно важное свойство реляционной подстановки. Пусть A, B — некоторые множества атомов, причём $B \subseteq A$. По определению реляционной подстановки очевидно, что для какой бы то ни было реляционной символьной интерпретации $\mathcal{J}$ верно $\left[\!\left[\bigwedge_{a \in A} a\right]\!\right]_{\mathcal{J}} \vdash \left[\!\left[\bigwedge_{a \in B} a\right]\!\right]_{\mathcal{J}}$, т.к. конъюнкция (3.1) для подстановки $\left[\!\left[\bigwedge_{a \in B} a\right]\!\right]_{\mathcal{J}}$ берётся по подмножеству элементов в конъюнкции для подстановки $\left[\!\left[\bigwedge_{a \in A} a\right]\!\right]_{\mathcal{J}}$.

Вернёмся к доказательству леммы. Заметим, что поскольку $Groups$ содержит лишь подмножества $atoms$, то выполнено следующее: $\{R_{i_1}(\overline{x}_{i_1}), \ldots, R_{i_k}(\overline{x}_{i_k})\} \subseteq atoms$ для всех $\{i_1, \ldots, i_k\} \in Groups$. Поэтому из (4.7) можно заключить, что для всех $\{i_1, \ldots, i_k\} \in Groups$ справедливо следующее:

$$M \models \left[\!\left[\bigwedge_{j \in \{i_1, \ldots, i_k\}} R_j(\overline{x}_j) \right]\!\right]^{b-1}_\sigma .$$

Другими словами, имеем $M \models O^{b-1}_{\sigma}(syms)[vars/\overline{v}_{syms}]$, и при этом выполнено следующее:

$$M \models \Big[\!\Big[\bigwedge_{\substack{R_j(\overline{x}_j)\in atoms \\ j\notin\{i_1,\ldots,i_k\}}} R_j(\overline{x}_j) \Big]\!\Big]^{b-1}_{\sigma}.$$

.

Поэтому ψ' в строке 29 выполняется в M. Так как по определению проекция на основе моделей сохраняет выполнимость в M, ψ' в строке 30 также выполняется в M. Поэтому $O^{b-1}_{\sigma}(syms) \wedge \psi'[\overline{v}_{syms}/vars]$ выполнимо в $\mathcal{T}$ (например, в M с переименованием $vars$ в $\overline{v}_{syms}$). Таким образом, первое утверждение леммы доказано.

Второе утверждение леммы следует из максимальности sat_{ρ}. Действительно, заметим, что по построению $[\![R_{i_j}(\overline{x}_{i_j})]\!]^{b-1}_{\rho} \notin sat_{\rho}$ для всех $j \in \{1,\ldots,k\}$. Но если $[\![R_{i_1}(\overline{x}_{i_1})]\!]^{b-1}_{\rho} \wedge \cdots \wedge [\![R_{i_k}(\overline{x}_{i_k})]\!]^{b-1}_{\rho} \wedge \psi'$ выполнимо, то добавление формул $[\![R_{i_1}(\overline{x}_{i_1})]\!]^{b-1}_{\rho}, \ldots, [\![R_{i_k}(\overline{x}_{i_k})]\!]^{b-1}_{\rho}$ в sat_{ρ} оставляет его совместным с $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \eta$ в теории $\mathcal{T}$, что противоречит максимальности sat_{ρ}. □

Теперь покажем, что RelBndSafety может вывести лишь конечное число ветвей и породить лишь конечное число вопросов.

Лемма 16. Процедура RelBndSafety(B, ρ, σ) может вывести лишь конечное число ветвей, т.е. ρ может обновиться лишь конечное число раз.

Доказательство. Покажем индукцией по уровню $b \geqslant -1$, что на каждом уровне алгоритм может вывести лишь конечное число ветвей.

Заметим, что т.к. $O^{-1}_{\sigma}(r) = U^{-1}_{\rho}(P) = \bot$ для всех $r \in \mathbb{N}^{\mathcal{R}}$ и $P \in \mathcal{R}$, то для всех $r \in \mathbb{N}^{\mathcal{R}}$ верно $[\![body(r)]\!]^{-1}_{\sigma} = [\![r]\!]^{-1}_{\rho}$ формулы. Поэтому оба условия в строках 21 и 4 не могут не выполниться на уровне $b = 0$ одновременно. Следовательно, алгоритм не может попасть в строку 31 на уровне $b = 0$. Но т.к. новые ветви на уровне b выводятся только в результате ответов на вопросы уровня b (строка 22), ρ никогда обновляется на уровне -1, т.е. база индукции доказана.

Далее предположим, что на уровне $b-1$ алгоритм может вывести лишь конечное число ветвей. Каждая выведенная ветвь на уровне b получается проекцией формулы $\eta \wedge sat^{P}_{\rho} \equiv [\![body(C)]\!]^{b-1}_{\rho}$ для некоторого дизъюнкта $C \in \mathcal{S}$ (см. строку 17). Но так как в $\mathcal{S}$ конечное число дизъюнктов, и возможно конечное число различных U^{b-1}_{ρ}, существует лишь конечное число различных формул

$[\![body(C)]\!]_{\rho}^{b-1}$. Но по определению проекции на основе моделей образ отображения $\left\{M \mapsto \text{MBP}\left([\![body(C)]\!]_{\rho}^{b-1}, \overline{\ell}_r, M\right)\right\}$ конечен, поэтому возможно лишь конечное число ветвей на уровне b.

Но по леммам 14 и 15 RelBndSafety не может вывести одну и ту же ветвь дважды: каждая ветвь может быть выведена только в результате ответа на некоторый вопрос, для которого в ρ недостаточное количество ветвей, т.е. после вывода ветви все вопросы, в результате ответа на которые эту ветвь возможно вывести повторно, не буду быть помещены в Q по лемме 15. □

Размером вопроса $\langle r, \varphi, b\rangle$ назовём мощность мультимножества r. Покажем, что размеры всех любого вопроса в RelBndSafety ограничен сверху.

Лемма 17. Для любой системы дизъюнктов существует такое $s \in \mathbb{N}$, что размер любого вопроса в Q не превышает s.

Доказательство. Размер первого вопроса $\langle P_0, \top, B\rangle$ равен 1. Размер каждого нового вопроса, добавляемого в Q, равен мощности $syms$ (см. строку 31). Каждое множество $syms$ является элементом множества $Groups$, возвращаемого процедурой Partition. Из требований, предъявляемых к Partition (см. раздел 4.4), имеем следующее:

- либо $syms$ является множеством символов, каждый из которых не рекурсивен ни с каким символом в r,
- либо $syms$ является множеством символов, каждый из которых рекурсивен с некоторым символом в r, и при этом мощность $syms$ не превышает мощности r.

Более того, символы в $syms$ входят в тело соответствующих символов из r, т.е. между символами в r и в $syms$ существуют рёбра в графе зависимостей.

Из данных двух наблюдений можно заметить, что порядок, в котором вопросы помещаются в Q в точности соответствует порядку обхода символов алгоритмом CHCproduct (см. листинг 1 в главе 2). Более формально, для любого вопроса $\langle syms, \cdot, \cdot\rangle$ найдётся символ $R \in \mathcal{R}'$ системы, возвращаемой алгоритмом CHCproduct($\mathcal{S}$), такой, что $|syms| \leqslant size(R)$. Но в лемме 11 показано, что размер всех символов в системе ограничен сверху.[3] □

[3] Ясно, что данное рассуждение не формально и не является строгим доказательством утверждения леммы. Для получения строгого доказательства достаточно адаптировать доказательства соответствующих лемм из раздела 2.7.3. Эта рутинная работа здесь опускается.

Лемма 18. Процедура RelBndSafety(B, ρ, σ) может породить лишь конечное число вопросов, т.е. Q может обновиться лишь конечное число раз.

Доказательство. Докажем утверждение в три этапа; первый и второй этапы будут показывать утверждение леммы для модификации RelBndSafety. На каждом этапе будем доказывать, что на каждом уровне $B, B-1, \ldots, 0$ может быть задано лишь конечное число вопросов.

На каждом этапе будем пользоваться индукцией по уровню b с базой $b = B$ и переходом $b \to b-1$. База индукции общая для всех этапов: добавление вопроса происходит со строгим уменьшением уровня (см. строку 31), поэтому на уровне B в Q будет добавлен единственный вопрос $\langle P_0, \top, B\rangle$. Далее, заметим что для любого вопроса $\langle r, \pi, b\rangle$, добавляемого в Q, верно $0 \leqslant b \leqslant B$, и $|r| \leqslant s$ для некоторого s (по предыдущей лемме). Таким образом, на любом уровне возможно лишь конечное множество комбинаций (r, b). Поэтому несмотря на то, что вопрос является тройкой $\langle r, \pi, b\rangle$, на каждом этапе достаточно показать лишь конечность множества вторых элементов π.

Этап 1. На первом этапе представим, что алгоритм не обновляет ни σ, ни ρ, т.е. удалим строки 6 и 22 из листинга 4. Покажем, что процедура может поместить в Q лишь конечное число *различных* вопросов (факт повторного добавления в Q на данном этапе не имеет значения).

По индукционному предположению, на уровне b в Q может быть добавлено лишь конечное число различных вопросов. Но каждый из вопросов на уровне может породить лишь конечное число вопросов на уровне $b-1$ (см. строку 31). Действительно, множество правил каждого символа в $\mathcal{S}$ конечно, мощность r ограничена некоторым s, U_ρ^{b-1} и O_σ^{b-1} не меняются, поэтому при попадании в строку 25 для каждого $\langle r, \pi, b\rangle \in Q$ возможно лишь конечное число возможных $(atoms, \psi)$. Но т.к. множество всех подмножеств конечного множества конечно, Partition может вернуть лишь конечное множество различных результатов. Но по конечности образа оператора MBP (см. раздел 4.1) в строке 30 возможно лишь конечное множество возможных результатов проекции, а значит и множество возможных ψ' при попадании в строку 31 конечно.

Этап 2. На втором этапе разрешим обновление σ, т.е. «вернём» строку 6. Пусть вопрос $\langle syms, \pi', b-1\rangle$, где $\pi' \equiv \psi'[\overline{v}_{syms}/vars]$, порождён вопросом

$\langle r, \pi, b\rangle$ в строке 31. Достаточно показать, что при фиксированных множествах $atoms$ и $Groups$ процедура может добавить в Q лишь конечное количество вопросов для $syms$.

Если вопрос $\langle syms, \pi', b-1\rangle$ отвечен положительно (строка 7), то по лемме 14, после обновления σ в строке 6 формула $O_\sigma^{b-1}(syms) \wedge \pi'$ становится невыполнимой в $\mathcal{T}$. Предположим, что процедура снова попадает в строку 30 с теми же $atoms$, $Groups$ и ψ' и с некоторой новой интерпретацией M'. По построению для нового вопроса вида $\langle syms, \pi'', b-1\rangle$ гарантировано, что $M' \models O_\sigma^{b-1}(syms) \wedge \pi''$, где $\pi'' \equiv \text{MBP}\left(\psi', \overline{v}_r \cup \overline{\ell}_r \setminus vars, M\right)[\overline{v}_{syms}/vars]$. Но т.к. $O_\sigma^{b-1}(syms) \wedge \pi'$ невыполнимо в $\mathcal{T}$, ни одна модель π' не выполняет $O_\sigma^{b-1}(syms) \wedge \pi''$.

Следовательно, M' не принадлежит прообразу π' отображения $\{M \mapsto \text{MBP}\left(\psi', \overline{v}_r \cup \overline{\ell}_r \setminus vars, M\right)\}$. Дальнейший ответ на вопрос $\langle syms, \pi'', b-1\rangle$ исключит модели из прообраза π'' (в частности, M') и т.д. По конечности образа MBP отображение $\{M \mapsto \text{MBP}\left(\psi', \overline{v}_r \cup \overline{\ell}_r \setminus vars, M\right)\}$ разбивает класс моделей ψ' в теории $\mathcal{T}$ на конечное число таких прообразов. Следовательно, рано или поздно этот процесс завершится.

Это означает, что при фиксированном вопросе $\langle r, \pi, b\rangle$, множествам $atoms$ и $Groups$, спустя конечное число положительных ответов на вопросы в Q, все вопросы к $syms$ на уровне $b-1$ будут исчерпаны. Так как добавление лемм в σ для других мультимножеств символов в $Groups$ может лишь усиливать ψ', положительные ответы для других вопросов в Q не меняют вышесказанного.

Этап 3. Наконец, рассмотрим всю процедуру RelBndSafety. На первых двух этапах было показано, что при фиксированном U_ρ^b в Q может быть добавлено лишь конечное число вопросов уровня b. Но по лемме 16 множество различных вариантов U_ρ^b конечно для каждого уровня b. □

Наконец, покажем, что каждая итерация RelBndSafety либо отвечает на некоторый вопрос, либо добавляет в Q новый вопрос.

Лемма 19. Каждая итерация процедуры RelBndSafety либо удаляет, как минимум, один вопрос из Q, либо добавляет, как минимум, один вопрос в Q.

Доказательство. На каждой итерации алгоритм выбирает вопрос $\langle r,\pi,b\rangle$ из Q. Для этого запроса либо $[\![body(r)]\!]_\sigma^{b-1} \wedge \pi$ невыполнимо в $\mathcal{T}$, либо $[\![body(r)]\!]_\sigma^{b-1} \wedge \pi$ выполнимо в $\mathcal{T}$.

Случай 1: $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ невыполнимо в $\mathcal{T}$. Так как $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ невыполнимо в $\mathcal{T}$, алгоритм попадает в строку 7. При этом по определению интерполянта (см. раздел 4.1) гарантируется, что $\psi \wedge \eta$ невыполнимо в $\mathcal{T}$. Таким образом, из Q удалится как минимум вопрос $\langle r,\pi,b\rangle$.

Случай 2: $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ выполнимо в $\mathcal{T}$. Так как $[\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ выполнимо в $\mathcal{T}$, алгоритм попадает в цикл в строке 13. Из $M \models [\![body(r)]\!]^{b-1}_{\sigma} \wedge \pi$ следует $M \models [\![body(P)]\!]^{b-1}_{\sigma}$ для всех P в r, откуда следует существование такого дизъюнкта $C \in rules(P)$, т.е. алгоритм успешно сможет выбрать дизъюнкт C на каждой итерации в строке 14.

Пусть $C_1, \ldots, C_{|r|}$ — все такие дизъюнкты, полученные при итерировании P по r. Снова, возможны два случая.

Случай 2.1: $M \models [\![body(C_i)]\!]^{b-1}_{\rho} \wedge \pi$ для всех $i \in \{1, \ldots, |r|\}$. При этом ветви всех атомов выполняются в M, т.е. множество $atoms$ окажется пустым (см. строку 19). Следовательно $M \models \psi_P$ для всех P в r. Также напомним, что $M \models \pi$. Таким образом, $M \models \bigwedge^{+}_{P \text{ в } r} \psi_P \wedge \pi$, т.е. из Q удалится как минимум вопрос $\langle r,\pi,b\rangle$.

Случай 2.2: $M \not\models [\![body(C_i)]\!]^{b-1}_{\rho} \wedge \pi$ для некоторого $i \in \{1, \ldots, |r|\}$. Поскольку $M \models [\![body(C_i)]\!]^{b-1}_{\sigma} \wedge \pi$ и $M \not\models [\![body(C_i)]\!]^{b-1}_{\rho} \wedge \pi$, в тело C входит как минимум один неинтерпретированный атом (иначе $[\![body(C_i)]\!]^{b-1}_{\sigma} \equiv [\![body(C_i)]\!]^{b-1}_{\rho}$). Следовательно, в теле C_i должен быть минимум один атом $R(\overline{x})$, для которого $M \not\models [\![R(\overline{x})]\!]^{b-1}_{\rho}$ (иначе получаем противоречие с условием случая 2.2). Следовательно, при попадании в строку 25 множество $atoms$ не будет пустым. Но т.к. из требований к Partition $\cup Groups = atoms$, множество $Groups$ не будет пустым, и в строке 31 в Q добавится минимум один вопрос. □

И RelRecMc, и RelBndSafety проверяют на выполнимость формулы языка ограничений (например, в строке 10 листинга 3 или в строке 21 листинга 4). Некоторые теории могут быть неразрешимы, т.е. не существует процедуры, которая всегда останавливается и определяет выполнимость формулы в данной теории. Очевидно, RelRecMc и RelBndSafety в общем случае не будут завершаться для неразрешимых теорий. Однако такая причина их незавершаемости не представляет большого интереса. Для того, чтобы исследовать

завершаемость алгоритмов вне зависимости от разрешимости теории $\mathcal{T}$, будем считать, что RelRecMc и RelBndSafety могут консультироваться с *оракулом* выполнимости в $\mathcal{T}$.

Назовём оракулом выполнимости в $\mathcal{T}$ процедуру, которая для любой формулы языка ограничений φ за один шаг проверяет выполнимость φ в $\mathcal{T}$. Также подразумевается, что такой оракул способен выдавать модели формул, Крейговские интерполянты и проектировать формулы на основе моделей; на практике таким оракулом является SMT-решатель разрешимых теорий (см. раздел 1.1.4).

Теорема 25. Процедура RelBndSafety полна относительно оракула выполнимости в $\mathcal{M}$, т.е. при наличии оракула выполнимости в $\mathcal{M}$ для любой системы $\mathcal{S}$ и уровня $b \in \mathbb{N}$ RelBndSafety останавливается и возвращает либо значение `ДОСТИЖИМО`, либо `НЕДОСТИЖИМО`.

Доказательство. По лемме 19 каждая итерация процедуры либо добавляет в Q вопрос, либо удаляет из Q вопрос. По лемме 18 в Q может быть совершено лишь конечное число добавлений вопросов, поэтому после некоторого числа итераций очередь Q опустеет, и процедура завершится (см. строку 2). □

Теперь, когда доказана корректность и относительная полнота RelBndSafety, можно охарактеризовать RelRecMc. Сначала покажем, что процедура RelRecMc является ко-разрешающей процедурой проблемы выполнимости дизъюнктов.

Теорема 26. При наличии оракула выполнимости в $\mathcal{M}$ для любой невыполнимой системы $\mathcal{S}$ RelRecMc останавливается и возвращает значение `НЕВЫПОЛНИМА`.

Доказательство. По полноте исчисления гиперрезолюций (см. утверждение 3) у системы $\mathcal{S}$ существует резолютивное опровержение. Пусть B — наименьшая высота резолютивного опровержения. Тогда по предыдущей теореме вызовы RelBndSafety на уровнях $0,1,\ldots,B$ завершатся, причём по теореме 23 результат `НЕДОСТИЖИМО` будет возвращён лишь на уровне B. □

Так как проблема выполнимости дизъюнктов Хорна с ограничениями неразрешима, на выполнимых системах RelRecMc в общем случае не завершается.

4.6.3 Дополнительные свойства

Процедура RelRecMc ведёт себя аналогично «классическому» алгоритму проверки достижимости, направляемой свойством, на линейных системах дизъюнктов и *обобщает* её поведение на нелинейных системах: если Partition группирует неинтерпретированные атомы в теле дизъюнкта на группы размера 1, то алгоритм строит «классические» сертификаты выполнимости.

В некоторых случаях, когда «классический» PDR не может вывести сертификат выполнимости нелинейных систем из-за непредставимости моделей в языке ограничений, RelRecMc находит реляционный сертификат выполнимости. Напротив, если «классический» PDR успешно доказывает или опровергает выполнимость системы, RelRecMc также справляется (с точностью до «угадывания» нужных интерполянтов в строке 5 листинга 4). Так как «классический» PDR здесь не описан, это утверждение остаётся без доказательства.

Глава 5. Реализация и эксперименты

В данной главе описана программная реализация алгоритмов CHCprodуct и RelRecMc , а также результаты экспериментов.

5.1 Реализация

И алгоритм CHCprodуct (см. раздел 2.7), и алгоритм RelRecMc (см. главу 4) были реализованы в SMT-решателе Z3 [11] на языке C++.

SMT-решатель Z3. На сегодняшний день Z3 является одиним из самых популярных SMT-решателей. Его архитектура приведена на рис. 8. Z3 состоит из трёх больших модулей: ядро SMT-решателя, подсистема μZ и программный интерфейс приложения.

Z3 имеет API для различных языков программирования, среди которых C++, C#, Java, Python, OCaml. Кроме того, в Z3 встроен синтаксический анализатор формата SMT-LIB 2.0 [140].

Самая сложная часть Z3 — ядро SMT-решателя. Изначально входная формула упрощается соответствующей компонентой. Проверка упрощённой формулы на выполнимость состоит в итеративном взаимодействии решателей теорий с SAT-решателем.

SAT-решатель в Z3 использует большинство современных методов для уменьшения пространства поиска, включая индексирование просмотренных литералов [141], нехронологический возврат и обучение дизъюнктам [37] и т.д.

Комбинирование теорий осуществляется алгоритмом, специально разработанным авторами решателя [142]. Z3 поддерживает множество теорий, включая линейную целочисленную арифметику, вещественную арифметику, теорию неинтерпретированных функций с равенством, теорию массивов и битовых векторов, алгебраические типы данных.

Обработка кванторов осуществляется соответствующей компонентой. Работа с кванторами в Z3 осуществляется на основе проекции на основе моделей (см. раздел 4.1) и E-сопоставления [143].

Подсистема μZ. Решением систем дизъюнктов Хорна в Z3 занимается подсистема, которая называется μZ. μZ является инструментарием, в котором реализованы все основные примитивы решения систем дизъюнктов Хорна: син-

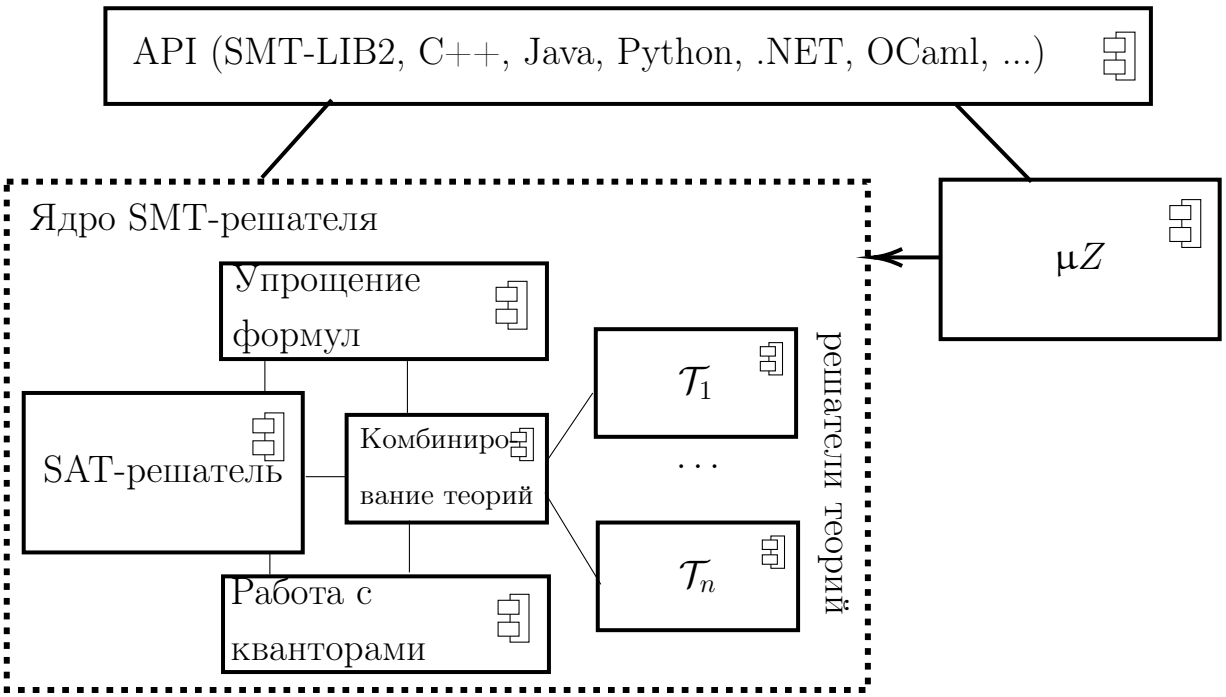


Рис. 8. Архитектура SMT-решателя Z3

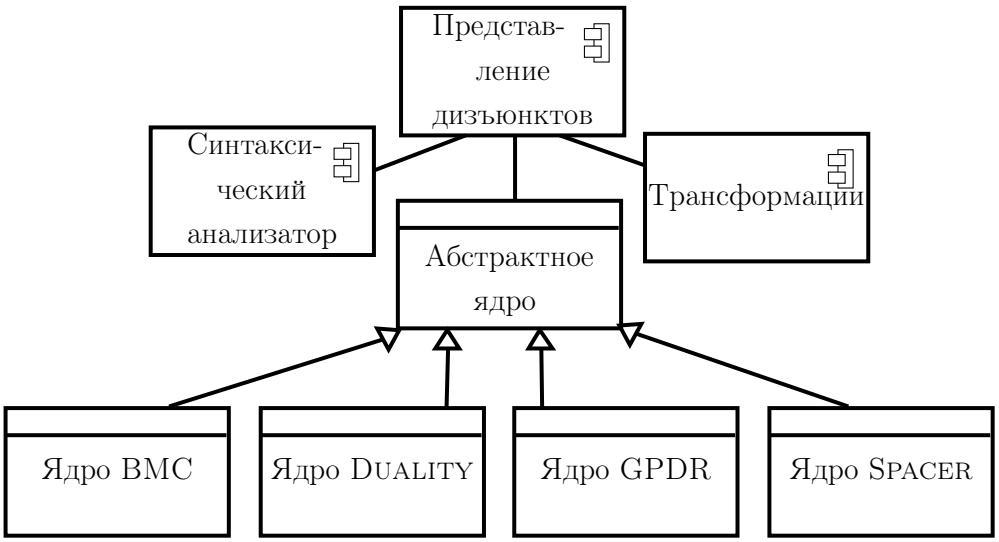


Рис. 9. Архитектура подсистемы μZ

таксический анализатор входного формата, внутреннее представление систем дизъюнктов, тактики переписывания дизъюнктов и абстрактная реализация ядра решателя дизъюнктов Хорна. Архитектура μZ представлена на рис. 9.

Компонента "Синтаксический анализатор"содержит набор расширений синтаксического анализатора SMT-LIB 2.0-формата. В ней добавляется обработка синтаксических конструкций `declare-rel`, `declare-var`, `rule` и `query`[1].

[1]Подробнее об этих конструкциях см. `https://rise4fun.com/Z3/tutorial/fixedpoints` (дата обращения: 04.04.2020).

В компоненте "Представление дизъюнктов"описано внутренне представление систем дизъюнктов и набор вспомогательных процедур работы с ними. Представление ограничений заимствуется из ядра SMT-решателя Z3.

Компонента "Трансформации"содержит набор тактик синтаксических преобразований систем дизъюнктов. Среди прочих, там реализованы преобразование вопросов-ответов [69], добавление инвариантов Kappa [144,145], распространение равенств, магические множества, нормализация чисел для дизъюнктов над вещественной арифметикой, построение срезов [146] и т.д.

Непосредственно решением систем дизъюнктов в μZ занимаются различные ядра, в которых реализованы те или иные подходы. Каждое такое ядро наследует абстрактное ядро. В μZ реализовано несколько подходов к решению дизъюнктов Хорна. Среди них — BMC [3], Duality [147], GPDR [125] и Spacer [6].

Алгоритм CHCproduct. Данный алгоритм реализован как тактика переписывания систем дизъюнктов в инструментарии μZ (компонента «Трансформации» на рис. 9).

Тактика принимает систему дизъюнктов Хорна во внутреннем представлении μZ. Далее на этой системе запускается алгоритм CHCproduct. После использовании тактики CHCproduct ядро решателя дизъюнктов будет запускаться не на входной системе, а на синхронизированной системе.

Общий объём реализации составил 510 строк кода. Реализация CHCproduct была интегрирована в основную ветку Z3[2].

Алгоритм RelRecMc. Данный алгоритм был реализован в ядре Spacer (см. рис. 9)[3].

Ядро принимает систему дизъюнктов Хорна во внутреннем представлении μZ. Далее на ней запускается алгоритм RelRecMc с незначительными оптимизациями. Если алгоритм останавливается, то возвращается либо реляционный сертификат выполнимости системы, либо дерево её гиперрезолютивного опровержения.

[2] `https://github.com/Z3Prover/z3/blob/master/src/muz/transforms/dl_mk_synchronize.cpp` (дата обращения: 04.04.2020).

[3] Реализация доступна по ссылке `https://github.com/dvvrd/z3` (дата обращения: 04.04.2020).

Было переиспользовано множество примитивов Spacer, включая извлечение интерполянтов Крейга и проекцию на основе моделей, управление очередью вопросов, извлечение ветвей, общая канва алгоритма.

Кратко опишем главные изменения в кодовой базе ядра Spacer.

– *Построение ответа.* Было реализовано представление, построение и вывод реляционных сертификатов выполнимости.
– *Представление лемм и вопросов.* Представление лемм и вопросов изменено, т.к. они привязываются к мультимножеству неинтерпретированных символов, а не к одному.
– *Взаимодействие с ядром SMT-решателя.* В Spacer реализованы сложные механизмы взаимодействия с SMT-решателем. Взаимодействие строится таким образом, что количество SMT-запросов сокращается до минимума, более эффективно переиспользуется информация с различных уровней, а ответы SMT-решателя дают больше информации для ускорения сходимости алгоритма. Привязка лемм и вопросов к мультимножествам символов потребовала технически сложных изменений в этом взаимодействии.
– *Управление дизъюнктами.* Существенно была переработана работа с дизъюнктами и управление переменными. Эти правки коснулись практически всей кодовой базы ядра Spacer: практически все подсистемы были спроектированы в предположении, что операции выполняются на одном дизъюнкте. В RelRecMc операции выполняются на мультимножестве дизъюнктов. При этом было решено множество технических трудностей, связанных с управлением переименованием совпадающих переменных в телах дизъюнктов.

Общий объём изменений в исходном коде Spacer составил 3000 строк кода[4].

5.2 Эксперименты

Было проведено сравнение полученных реализаций с решателями Spacer и HoIce [9] на двух наборах тестов[5]. Эксперименты были проведены на компью-

[4]На момент написания диссертации реализация RelRecMc планируется к интеграции в основную ветку Z3.

[5]Тесты доступны по ссылке `https://github.com/dvvrd/spacer-benchmarks` (дата обращения: 04.04.2020).

Кол-во тестов	HoIce	Spacer	RelRecMc
840	808	788	807

Табл. 5. Количество решённых тестов из тестового набора [9]

тере под управлением ОС Arch Linux с процессором Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz.

Эксперименты были проведены на двух тестовых наборах.

Тестовый набор [9]. Первый набор тестов содержит 840 проблем из [9], сгенерированных верификатором MoCHi [148] как условия верификации различных функциональных программ высших порядков на языке OCaml. Среди этих тестов были как линейные, так и нелинейные системы. Таймаут для этих тестов был 60 секунд.

Данный тестовый набор примечателен тем, что решатель HoIce, использующий обучение с учителем, показывает лучшие результаты, чем Spacer за счёт того, что HoIce оптимизирован специально для решения нелинейных систем дизъюнктов Хорна, порождённых из функциональных программ высшего порядка.

Количество решённых тестовых проблем представлены в таблице 5.

Spacer решил 788 из 840 проблем с 50 таймаутами и 2 ошибками времени исполнения. RelRecMc решил 807 проблем с 34 таймаутами. Накладные расходы по времени в сравнении со Spacer незначительны (менее 0.1 секунды на 87% проблем).

RelRecMc решил большинство проблем, решённых Spacer. Тем не менее, 10 были решены Spacer, но не RelRecMc; это объясняется неудачным подбором Крейговских интерполянтов. Все линейные системы, решённые Spacer, решены и RelRecMc.

HoIce решил 808 проблем с 26 таймаутами и 6 ошибками времени исполнения, но и Spacer, и RelRecMc затратил меньше времени, чем HoIce на решённых проблемах.

Таким образом, реляционное обобщение PDR показывает лучшие результаты в сравнении с обычным PDR на нелинейных системах, позволяя конкурировать с подходами, основанными на обучении с учителем и оптимизированными специально для решения систем определённого вида.

Кол-во тестов	HoIce	Spacer	CHCProduct	RelRecMc
37	11	11	24	32

Табл. 6. Количество решённых тестов из реляционного тестового набора

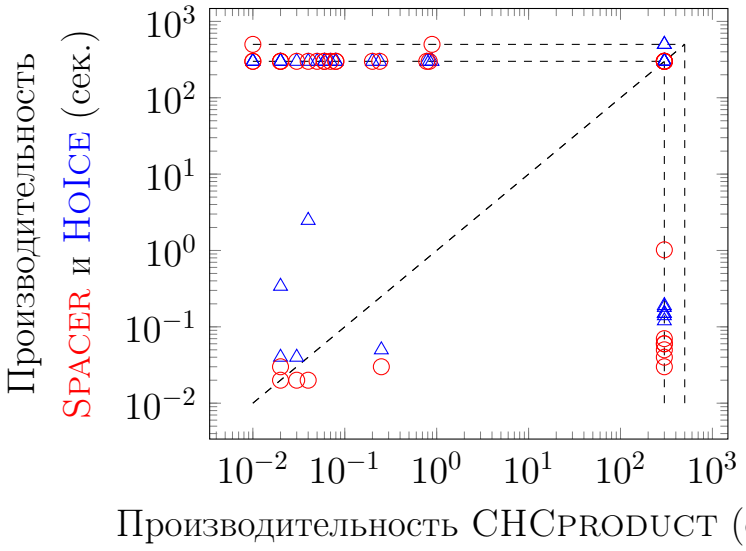


Рис. 10. Сравнение CHCProduct с другими решателями.

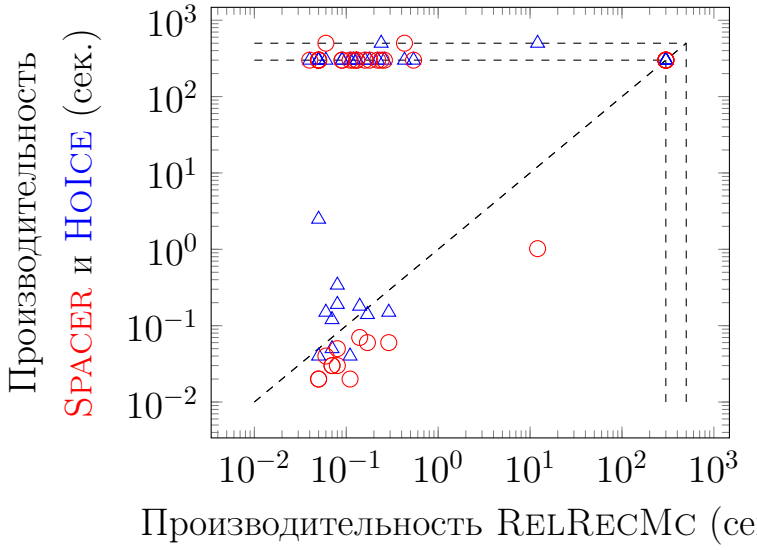


Рис. 11. Сравнение RelRecMc с другими решателями.

Реляционный тестовый набор. Второй набор тестов содержит 37 проблем реляционной верификации из [149]. Каждая система в данном тестовом наборе является нелинейной.

В данном сравнении участвуют четыре алгоритма: RelRecMc, Spacer, HoIce и синтаксическое преобразование CHCProduct с последующей применением Spacer.

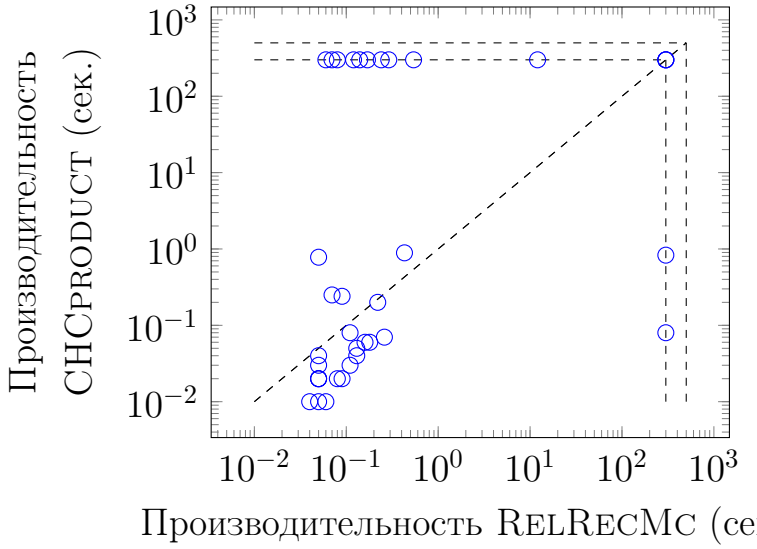


Рис. 12. Сравнение RELRECMC с CHCPRODUCT.

Количество решённых тестовых проблем представлены в таблице 6. SPACER и HOICE решили только 11 из 37 задач с 5-минутным таймаутом. CHCPRODUCT решил 24 задачи, а RELRECMC решил 32 задачи из 37.

Схематичное сравнение времени работы решателей показано на рис. 10, 11 и 12. Каждая точка на каждом графике представляет пару времён исполнения теста (сек. × сек.) решателем на оси x и другим решателем, на оси y. Если на оси y сразу два решателя, то они различаются разными цветами и формами маркеров на графике: решателю SPACER соответствует круглый маркер, HOICE — треугольный. Превышения временного лимита указаны внутренней пунктирной линией; на внешних пунктирных линиях лежат тесты, на которых решатели завершились с ошибкой времени исполнения.

Важно заметить, что RELRECMC решил некоторые задачи, с которыми не справился SPACER после применения CHCPRODUCT. Для невыполнимых систем большого размера, например, `point-location*`, CHCPRODUCT порождает экспоненциальное количество правил, расходуя всё отведённое время, в то время как другие решатели справляются с поиском гиперрезолютивного опровержения за секунды.

Следовательно, и CHCPRODUCT, и RELRECMC более эффективны, чем стандартные решатели дизъюнктов Хорна на проблемах реляционной верификации, но при этом RELRECMC показывает лучшие результаты, чем CHCPRODUCT за счёт того, что строит решение модульно, без явного перемножения дизъюнктов.

Заключение

Основные результаты работы заключаются в следующем.

1. Предложено синхронизирующее преобразование систем дизъюнктов Хорна с ограничениями первого порядка. Доказана корректность синхронизирующего преобразования. В частности, показано, что система выполнима тогда и только тогда, когда выполнима преобразованная, а также что если у системы существует символьная модель, то она существует и у преобразованной.
2. Предложен алгоритм CHCprodct, реализующий синхронизирующее преобразование дизъюнктов Хорна с ограничениями, доказана его корректность, завершаемость, проанализирована сложность.
3. Введено понятие реляционного сертификата выполнимости как решения нелинейных систем дизъюнктов.
4. Предложен алгоритм RelRecMc для автоматического, построения реляционных сертификатов выполнимости для систем дизъюнктов Хорна с ограничениями. Реализация обобщает подход PDR, не меняя его поведения на линейных системах, но позволяя эффективно выводить реляционные сертификаты выполнимости для нелинейных систем вместо «классических» символьных моделей.
5. Алгоритмы CHCprodct и RelRecMc реализованы в известном SMT-решателе Z3. Выполнено экспериментальное исследование данных алгоритмов на различных тестовых наборах, включающих условия верификации свойств безопасности программного обеспечения и проблем реляционной верификации.

В рамках **рекомендации по применению результатов работы** в индустрии и научных исследованиях указывается, что разработанный алгоритм применим для автоматизации рассуждений о системах дизъюнктов над произвольными теориями первого порядка, а также что его реализация выполнена в широко используемом SMT-решателе Z3. Это позволяет модульно реализовать инструменты автоматического доказательства корректности программного кода на любых языках, вывода уточняющих типов произведений функциональных программ, автоматического аннотирования императивных программ в декартовой логике Хоара, а также проверки эквивалентности программ и провер-

ки свойств невмешательства, являющихся важными в области компиляторов и компьютерной безопасности.

Также были определены **перспективы дальнейшей разработки тематики**, основным из которых является разработка подхода к автоматическому определению нетривиальных стратегий синхронизации отношений нелинейных систем дизъюнктов Хорна. Это позволит достичь максимальной гибкости в представлении сертификатов корректности программ за счёт рассмотрения эквивалентных трансформаций изначальной системы. Это особо важно в контексте автоматического доказательства свойств программ, манипулирующих структурами данных с произвольным доступом (например, массивами), позволяя выводить бескванторные инварианты для программ со сложной логикой обращения к различным областям памяти.

Список сокращений и условных обозначений

		Стр.
CLP	constraint logic programming, парадигма программирования, в которой программа представляется в виде множества дизъюнктов Хорна с ограничениями	12
SMT	satisfiability modulo theories, задача определения выполнимости формулы языка первого порядка в некоторой теории	15
SAT	satisfiability problem, задача определения выполнимости формулы языка высказываний	16
CDCL	conflict driven clause learning, алгоритм решения задачи **SAT**	16
CHC	constrained Horn clause, дизъюнкт Хорна с ограничением	19
WLP	weakest liberal precondition, слабейшее либеральное предусловие	30
SyGuS	syntax-guided synthesis, задача синтеза программы по синтаксической и семантической спецификации	38
CEGAR	counterexample-guided abstraction refinement, подход к верификации программ, основанный на итеративном уточнении абстракции программы	38
ICE	learning using Examples, Counter-examples, and Implications, подход к выводу индуктивных инвариантов методом обучения с учителем	38
PDR	Property Directed Reachability, подход к инкрементальному выводу индуктивных инвариантов программ или доказательства их отсутствия	39
IC3	Incremental Construction of Inductive Clauses for Indubitable Correctness, то же, что **PDR**	39
MBP	model-based projection, техника итеративного устранения кванторов в формулах	116

Список литературы

1. Barrett, Clark W. Satisfiability Modulo Theories / Clark W. Barrett, Cesare Tinelli // Handbook of Model Checking. — 2018. — P. 305–343. — URL: https://doi.org/10.1007/978-3-319-10575-8_11.

2. Baldoni, Roberto. A Survey of Symbolic Execution Techniques / Roberto Baldoni, Emilio Coppa, Daniele Cono D'elia, Camil Demetrescu, Irene Finocchi // ACM Computing Surveys (CSUR). — 2018. — Vol. 51, no. 3. — P. 1–39.

3. Biere, A. Bounded Model Checking / A. Biere, A. Cimatti, E.M. Clarke, O. Strichman, Y. Zhu // Advances in Computers. — 2003. — Vol. 58. — P. 118–149.

4. Godefroid, Patrice. Compositional Dynamic Test Generation / Patrice Godefroid // POPL. — ACM, 2007. — P. 47–54.

5. Jaffar, Joxan. Constraint Logic Programming / Joxan Jaffar, J-L Lassez // Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages / ACM. — 1987. — P. 111–119.

6. Komuravelli, Anvesh. SMT-Based Model Checking for Recursive Programs / Anvesh Komuravelli, Arie Gurfinkel, Sagar Chaki // LNCS, CAV. — Vol. 8559. — 2014. — P. 17–34.

7. Hojjat, Hossein. The ELDARICA Horn Solver / Hossein Hojjat, Philipp Rümmer // 2018 Formal Methods in Computer Aided Design (FMCAD) / IEEE. — 2018. — P. 1–7.

8. Grebenshchikov, Sergey. HSF(C): A Software Verifier Based on Horn Clauses / Sergey Grebenshchikov, Ashutosh Gupta, Nuno P. Lopes, Corneliu Popeea, Andrey Rybalchenko // Tools and Algorithms for the Construction and Analysis of Systems / Ed. by Cormac Flanagan, Barbara König. — Springer Berlin Heidelberg, 2012. — P. 549–551.

9. Champion, Adrien. HoIce: An ICE-Based Non-linear Horn Clause Solver / Adrien Champion, Naoki Kobayashi, Ryosuke Sato // Asian Symposium on Programming Languages and Systems / Springer. — 2018. — P. 146–156.

10. Fedyukovich, Grigory. Sampling Invariants from Frequency Distributions / Grigory Fedyukovich, Samuel Kaufman, Rastislav Bodík // FMCAD. — IEEE, 2017. — P. 100–107.

11. de Moura, Leonardo Mendonça. Z3: An Efficient SMT Solver / Leonardo Mendonça de Moura, Nikolaj Bjørner // LNCS, TACAS. — Vol. 4963. — Springer, 2008. — P. 337–340.

12. Gurfinkel, Arie. The SeaHorn Verification Framework / Arie Gurfinkel, Temesghen Kahsai, Anvesh Komuravelli, Jorge A. Navas // LNCS, CAV. — Vol. 9206. — Springer, 2015. — P. 343–361.

13. Kahsai, Temesghen. JayHorn: A Framework for Verifying Java programs / Temesghen Kahsai, Philipp Rümmer, Huascar Sanchez, Martin Schäf // LNCS, CAV, Part I. — Vol. 9779. — Springer, 2016. — P. 352–358.

14. Beyer, Dirk. Automatic Verification of C and Java Programs: SV-COMP 2019 / Dirk Beyer // International Conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 2019. — P. 133–155.

15. Bradley, Aaron R. SAT-Based Model Checking without Unrolling / Aaron R. Bradley // LNCS, VMCAI. — Vol. 6538. — Springer, 2011. — P. 70–87.

16. Eén, Niklas. Efficient Implementation of Property Directed Reachability / Niklas Eén, Alan Mishchenko, Robert K. Brayton // FMCAD. — IEEE, 2011. — P. 125–134.

17. Haase, Christoph. A Survival Guide to Presburger Arithmetic / Christoph Haase // ACM SIGLOG News. — 2018. — Vol. 5, no. 3. — P. 67–82.

18. Apt, Krzysztof R. Ten Years of Hoare's Logic: A Survey - Part I / Krzysztof R Apt // ACM Transactions on Programming Languages and Systems (TOPLAS). — 1981. — Vol. 3, no. 4. — P. 431–483.

19. Матиясевич, Юрий Владимирович. Диофантовость перечислимых множеств / Юрий Владимирович Матиясевич // Доклады Академии наук / Российская академия наук. — Vol. 191. — 1970. — P. 279–282.

20. Kafle, Bishoksan. Solving Non-Linear Horn Clauses Using a Linear Horn Clause Solver / Bishoksan Kafle, John P. Gallagher, Pierre Ganty // EPTCS, HCVS. — Vol. 219. — 2016. — P. 33–48.

21. Kafle, Bishoksan. Tree Dimension in Verification of Constrained Horn Clauses / Bishoksan Kafle, John P Gallagher, Pierre Ganty // Theory and Practice of Logic Programming. — 2018. — Vol. 18, no. 2. — P. 224–251.

22. Garg, Pranav. ICE: A Robust Framework for Learning Invariants / Pranav Garg, Christof Löding, P. Madhusudan, Daniel Neider // LNCS, CAV. — Vol. 8559. — Springer, 2014. — P. 69–87.

23. De Angelis, Emanuele. Relational Verification Through Horn Clause Transformation / Emanuele De Angelis, Fabio Fioravanti, Alberto Pettorossi, Maurizio Proietti // LNCS, SAS. — Vol. 9837. — 2016. — P. 147–169.

24. De Angelis, Emanuele. Predicate Pairing for Program Verification / Emanuele De Angelis, Fabio Fioravanti, Alberto Pettorossi, Maurizio Proietti // Theory and Practice of Logic Programming. — 2018. — Vol. 18, no. 2. — P. 126–166.

25. Clarkson, Michael R. Hyperproperties / Michael R. Clarkson, Fred B. Schneider // J. Comput. Secur. — 2010. — Vol. 18, no. 6. — P. 1157–1210.

26. Barthe, Gilles. Relational Verification Using Product Programs / Gilles Barthe, Juan Manuel Crespo, César Kunz // LNCS, FM. — Vol. 6664. — Springer, 2011. — P. 200–214.

27. Shemer, Ron. Property Directed Self Composition / Ron Shemer, Arie Gurfinkel, Sharon Shoham, Yakir Vizel // CAV, Part I. — Vol. 11561. — Springer, 2019. — P. 161–179.

28. Захаров, В. А. Проблема эквивалентности программ: модели, алгоритмы, сложность / В. А. Захаров. — Аргамак-Медиа Москва, 2016. — 304 с.

29. Щербина, ОА. Удовлетворение ограничений и программирование в ограничениях / ОА Щербина // Интеллектуальные системы. — 2011. — Vol. 15, no. 1-4. — P. 53–170.

30. Кейслер, Г. Теория моделей: Пер. с англ / Г. Кейслер, Ч. Ч. Чэн, С. С Гончаров, С. Д. Денисова. — Мир, 1977.

31. Верещагин, Николай Константинович. Языки и исчисления / Николай Константинович Верещагин, Александр Шень. — Издательство МЦНМО, 2012.

32. Герасимов, Александр Сергеевич. Курс математической логики и теории вычислимости: учебное пособие / Александр Сергеевич Герасимов. — ЛЕМА, 2011.

33. Barrett, Clark. CVC4 / Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, Cesare Tinelli // International Conference on Computer Aided Verification / Springer. — 2011. — P. 171–177.

34. Dutertre, Bruno. The Yices SMT Solver / Bruno Dutertre, Leonardo De Moura // Tool paper at http://yices. csl. sri. com/tool-paper. pdf. — 2006. — Vol. 2, no. 2. — P. 1–2.

35. Brummayer, Robert. Boolector: An Efficient SMT Solver for Bit-Vectors and Arrays / Robert Brummayer, Armin Biere // International Conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 2009. — P. 174–177.

36. Barrett, Clark. SMT-COMP: Satisfiability Modulo Theories Competition / Clark Barrett, Leonardo De Moura, Aaron Stump // International Conference on Computer Aided Verification / Springer. — 2005. — P. 20–23.

37. Silva, João P Marques. Grasp—A New Search Algorithm for Satisfiability / João P Marques Silva, Karem A Sakallah // The Best of ICCAD. — Springer, 2003. — P. 73–89.

38. Zhang, Lintao. Efficient Conflict Driven Learning in a Boolean Satisfiability solver / Lintao Zhang, Conor F Madigan, Matthew H Moskewicz, Sharad Malik // Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design / IEEE Press. — 2001. — P. 279–285.

39. Heule, Marijn JH. Proceedings of SAT Competition 2018 / Marijn JH Heule, Matti Juhani Järvisalo, Martin Suda et al. — 2018.

40. Heule, Marijn JH. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer / Marijn JH Heule, Oliver Kullmann, Victor W Marek // International Conference on Theory and Applications of Satisfiability Testing / Springer. — 2016. — P. 228–245.

41. Skolem, Thoralf Albert. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze, nebst einem Theoreme über dichte Mengen, von Th. Skolem / Thoralf Albert Skolem. — J. Dybwad, 1920.

42. Karmarkar, Narendra. A New Polynomial-Time Algorithm for Linear Programming / Narendra Karmarkar // Proceedings of the sixteenth annual ACM symposium on Theory of computing / ACM. — 1984. — P. 302–311.

43. Романовский, Иосиф Владимирович. Алгоритмы решения экстремальных задач / Иосиф Владимирович Романовский. — 1977.

44. Dutertre, Bruno. A Fast Linear-Arithmetic Solver for DPLL(T) / Bruno Dutertre, Leonardo De Moura // International Conference on Computer Aided Verification / Springer. — 2006. — P. 81–94.

45. McCarthy, John. Towards a Mathematical Science of Computation / John McCarthy // Program Verification. — Springer, 1993. — P. 35–56.

46. Stump, Aaron. A Decision Procedure for an Extensional Theory of Arrays / Aaron Stump, Clark W Barrett, David L Dill, Jeremy Levitt // Logic in Computer Science, 2001. Proceedings. 16th Annual IEEE Symposium on / IEEE. — 2001. — P. 29–37.

47. Bradley, Aaron R. What's Decidable about Arrays? / Aaron R Bradley, Zohar Manna, Henny B Sipma // International Workshop on Verification, Model Checking, and Abstract Interpretation / Springer. — 2006. — P. 427–442.

48. Nelson, Greg. Fast Decision Procedures Based on Congruence Closure / Greg Nelson, Derek C Oppen // Journal of the ACM (JACM). — 1980. — Vol. 27, no. 2. — P. 356–364.

49. Barrett, Clark. An Abstract Decision Procedure for a Theory of Inductive Data Types / Clark Barrett, Igor Shikanian, Cesare Tinelli // Journal on Satisfiability, Boolean Modeling and Computation. — 2007. — Vol. 3. — P. 21–46.

50. Oppen, Derek C. Reasoning about Recursively Defined Data Structures / Derek C Oppen // Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages / ACM. — 1978. — P. 151–157.

51. Suter, Philippe. Decision Procedures for Algebraic Data Types with Abstractions / Philippe Suter, Mirco Dotta, Viktor Kuncak // Acm Sigplan Notices. — 2010. — Vol. 45, no. 1. — P. 199–210.

52. Reynolds, Andrew. A Decision Procedure for (Co)Datatypes in SMT Solvers / Andrew Reynolds, Jasmin Christian Blanchette // International Conference on Automated Deduction / Springer. — 2015. — P. 197–213.

53. Loos, Rüdiger. Applying Linear Quantifier Elimination / Rüdiger Loos, Volker Weispfenning // The computer journal. — 1993. — Vol. 36, no. 5. — P. 450–462.

54. Матиясевич, Юрий Владимирович. Алгоритм Тарского / Юрий Владимирович Матиясевич // Компьютерные инструменты в образовании. — 2008. — no. 6.

55. Hadarean, Liana. A Tale of Two Solvers: Eager and Lazy Approaches to Bit-Vectors / Liana Hadarean, Kshitij Bansal, Dejan Jovanović, Clark Barrett, Cesare Tinelli // International Conference on Computer Aided Verification / Springer. — 2014. — P. 680–695.

56. Liang, Tianyi. An Efficient SMT Solver for String Constraints / Tianyi Liang, Andrew Reynolds, Nestan Tsiskaridze, Cesare Tinelli, Clark Barrett, Morgan Deters // Formal Methods in System Design. — 2016. — Vol. 48, no. 3. — P. 206–234.

57. Nelson, Greg. Simplification by cooperating decision procedures / Greg Nelson, Derek C Oppen // ACM Transactions on Programming Languages and Systems (TOPLAS). — 1979. — Vol. 1, no. 2. — P. 245–257.

58. Shostak, Robert E. Deciding combinations of theories / Robert E Shostak // International Conference on Automated Deduction / Springer. — 1982. — P. 209–222.

59. Apt, Krzysztof R. From Logic Programming to Prolog / Krzysztof R Apt et al. — Prentice Hall London, 1997. — Vol. 362.

60. Robinson, A. Automatic Deduction with Hyper-Resolution / A. Robinson // Int. Journal of Computer Mathematics. — 1965. — Vol. 1, no. 3. — P. 227–234.

61. Kropf, Thomas. Introduction to Formal Hardware Verification / Thomas Kropf. — Springer Science & Business Media, 2013.

62. Basin, David. Model Checking Security Protocols / David Basin, Cas Cremers, Catherine Meadows // Handbook of Model Checking. — 2011.

63. Handbook of Model Checking / Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, Roderick Bloem. — Springer, 2018. — URL: https://doi.org/10.1007/978-3-319-10575-8.

64. Kupferman, Orna. Model Checking of Safety Properties / Orna Kupferman, Moshe Y Vardi // Formal Methods in System Design. — 2001. — Vol. 19, no. 3. — P. 291–314.

65. Burch, Jerry R. Symbolic Model Checking: 10ˆ20 States and Beyond / Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, L. J. Hwang // Inf. Comput. — 1992. — Vol. 98, no. 2. — P. 142–170. — URL: https://doi.org/10.1016/0890-5401(92)90017-A.

66. Biere, Armin. Symbolic Model Checking without BDDs / Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Yunshan Zhu // LNCS, TACAS. — Vol. 1579. — Springer, 1999. — P. 193–207.

67. McMillan, Kenneth L. Interpolation and SAT-Based Model Checking / Kenneth L. McMillan // LNCS, CAV. — Vol. 2725. — Springer, 2003. — P. 1–13.

68. De Angelis, Emanuele. VeriMAP: a Tool for Verifying Programs Through Transformations / Emanuele De Angelis, Fabio Fioravanti, Alberto Pettorossi, Maurizio Proietti // International Conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 2014. — P. 568–574.

69. Gallagher, John P. Analysis and transformation tools for constrained Horn clause verification / John P Gallagher, Bishoksan Kafle // arXiv preprint arXiv:1405.3883. — 2014.

70. Apt, Krzysztof. Verification of sequential and concurrent programs / Krzysztof Apt, Frank S De Boer, Ernst-Rüdiger Olderog. — Springer Science & Business Media, 2010.

71. Dijkstra, Edsger Wybe. A Discipline of Programming / Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Etats-Unis Informaticien, Edsger Wybe Dijkstra. — prentice-hall Englewood Cliffs, 1976. — Vol. 1.

72. Дейкстра, Эдсгер Вибе. Дисциплина программирования [Главы из книги] / Эдсгер Вибе Дейкстра. — Мир, 1978.

73. Barnett, Mike. Weakest-Precondition of Unstructured Programs / Mike Barnett, K Rustan M Leino // ACM SIGSOFT Software Engineering Notes / ACM. — Vol. 31. — 2005. — P. 82–87.

74. Beyer, Dirk. Software Model Checking via Large-Block Encoding / Dirk Beyer, Alessandro Cimatti, Alberto Griggio, M Erkan Keremoglu, Roberto Sebastiani // 2009 Formal Methods in Computer-Aided Design / IEEE. — 2009. — P. 25–32.

75. Flanagan, Cormac. Extended Static Checking for Java / Cormac Flanagan, Cormac Flanagan, K Rustan M Leino, Mark Lillibridge, Greg Nelson, James B Saxe, Raymie Stata // ACM Sigplan Notices / ACM. — Vol. 37. — 2002. — P. 234–245.

76. Leino, K Rustan M. Efficient Weakest Preconditions / K Rustan M Leino // Information Processing Letters. — 2005. — Vol. 93, no. 6. — P. 281–288.

77. Gurfinkel, Arie. Efficient predicate abstraction of program summaries / Arie Gurfinkel, Sagar Chaki, Samir Sapra // NASA Formal Methods Symposium / Springer. — 2011. — P. 131–145.

78. Promsky, AV. Towards C-light Program Verification: Overcoming the Obstacles / AV Promsky // Program Understanding-2009. — 2009. — P. 53–63.

79. Reynolds, John C. Definitional Interpreters for Higher-Order Programming Languages / John C Reynolds // Proceedings of the ACM annual conference-Volume 2 / ACM. — 1972. — P. 717–740.

80. Bjorner, Nikolaj. Higher-Order Program Verification as Satisfiability Modulo Theories with Algebraic Data-Types / Nikolaj Bjorner, Ken McMillan, Andrey Rybalchenko // arXiv preprint arXiv:1306.5264. — 2013.

81. Pham, Long. Defunctionalization of Higher-Order Constrained Horn Clauses / Long Pham, Steven J Ramsay, C-H Luke Ong // arXiv preprint arXiv:1810.03598. — 2018.

82. German, Steven M. Reasoning about Procedures as Parameters / Steven M German, Edmund M Clarke, Joseph Y Halpern // Workshop on Logic of Programs / Springer. — 1983. — P. 206–220.

83. Ramsay, Steven J. A Type-Directed Abstraction Refinement Approach to Higher-Order Model Checking / Steven J Ramsay, Robin P Neatherway, C-H Luke Ong // ACM SIGPLAN Notices / ACM. — Vol. 49. — 2014. — P. 61–72.

84. Xi, Hongwei. Dependent Types in Practical Programming / Hongwei Xi, Frank Pfenning // Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. — 1999. — P. 214–227.

85. Vazou, Niki. Refinement Types for Haskell / Niki Vazou, Eric L Seidel, Ranjit Jhala, Dimitrios Vytiniotis, Simon Peyton-Jones // ACM SIGPLAN Notices / ACM. — Vol. 49. — 2014. — P. 269–282.

86. Dependent Types and Multi-Monadic Effects in F* / Nikhil Swamy, Catalin Hritcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub et al. // 43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). — ACM, 2016. — P. 256–270.

87. Vekris, Panagiotis. Refinement Types for TypeScript / Panagiotis Vekris, Benjamin Cosman, Ranjit Jhala // ACM SIGPLAN Notices. — 2016. — Vol. 51, no. 6. — P. 310–325.

88. Kazerounian, Milod. Refinement Types for Ruby / Milod Kazerounian, Niki Vazou, Austin Bourgerie, Jeffrey S Foster, Emina Torlak // International Conference on Verification, Model Checking, and Abstract Interpretation / Springer. — 2018. — P. 269–290.

89. Jhala, Ranjit. HMC: Verifying Functional Programs Using Abstract Interpreters / Ranjit Jhala, Rupak Majumdar, Andrey Rybalchenko // International Conference on Computer Aided Verification / Springer. — 2011. — P. 470–485.

90. Rondon, Patrick M. Liquid Types / Patrick M Rondon, Ming Kawaguci, Ranjit Jhala // ACM SIGPLAN Notices / ACM. — Vol. 43. — 2008. — P. 159–169.

91. Cathcart Burn, Toby. Higher-Order Constrained Horn Clauses for Verification / Toby Cathcart Burn, C-H Luke Ong, Steven J Ramsay // Proceedings of the ACM on Programming Languages. — 2017. — Vol. 2, no. POPL. — P. 11.

92. Kundu, Sudipta. Proving Optimizations Correct Using Parameterized Program Equivalence / Sudipta Kundu, Zachary Tatlock, Sorin Lerner // Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation. — New York, NY, USA: Association for Computing Machinery, 2009. — P. 327–337.

93. Lerner, Sorin. Automatically Proving the Correctness of Compiler Optimizations / Sorin Lerner, Todd Millstein, Craig Chambers // Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation. — New York, NY, USA: Association for Computing Machinery, 2003. — P. 220–231.

94. Pnueli, Amir. Translation Validation / Amir Pnueli, Michael Siegel, Eli Singerman // International Conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 1998. — P. 151–166.

95. Collberg, Christian. A Taxonomy of Obfuscating Transformations: Tech. Rep. 148 / Christian Collberg, Clark Thomborson, Douglas Low: Department of Computer Sciences, The University of Auckland, 1997.

96. Новикова, Татьяна Анатольевна. Математические модели и методы в решении задач реорганизации программ: Ph.D. thesis / Московский государственный университет имени М.В. Ломоносова. — 2016.

97. Goguen, Joseph A. Security Policies and Security Models / Joseph A Goguen, José Meseguer // 1982 IEEE Symposium on Security and Privacy / IEEE. — 1982. — P. 11–11.

98. Левина, Алла Борисовна. Комбинированные атаки по сторонним каналам: взлом COMP128 / Алла Борисовна Левина, Михаил Георгиевич Коровкин // Безопасность информационных технологий. — 2014. — Vol. 21, no. 3.

99. Felten, Edward W. Timing Attacks on Web Privacy / Edward W Felten, Michael A Schneider // Proceedings of the 7th ACM conference on Computer and communications security / ACM. — 2000. — P. 25–32.

100. Haeberlen, Andreas. Differential Privacy Under Fire. / Andreas Haeberlen, Benjamin C Pierce, Arjun Narayan // USENIX Security Symposium. — 2011.

101. Kocher, Paul C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems / Paul C Kocher // Annual International Cryptology Conference / Springer. — 1996. — P. 104–113.

102. Brumley, David. Remote Timing Attacks are Practical / David Brumley, Dan Boneh // Computer Networks. — 2005. — Vol. 48, no. 5. — P. 701–716.

103. Al Fardan, Nadhem J. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols / Nadhem J Al Fardan, Kenneth G Paterson // 2013 IEEE Symposium on Security and Privacy / IEEE. — 2013. — P. 526–540.

104. Chen, Shuo. Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow / Shuo Chen, Rui Wang, XiaoFeng Wang, Kehuan Zhang // 2010 IEEE Symposium on Security and Privacy / IEEE. — 2010. — P. 191–206.

105. Lahiri, Shuvendu K. Symdiff: A Language-Agnostic Semantic Diff Tool for Imperative Programs / Shuvendu K Lahiri, Chris Hawblitzel, Ming Kawaguchi, Henrique Rebêlo // International Conference on Computer Aided Verification / Springer. — 2012. — P. 712–717.

106. Lahiri, Shuvendu K. Differential Assertion Checking / Shuvendu K Lahiri, Kenneth L McMillan, Rahul Sharma, Chris Hawblitzel // Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering / ACM. — 2013. — P. 345–355.

107. Sousa, Marcelo. Verifying Semantic Conflict-Freedom in Three-Way Program merges / Marcelo Sousa, Isil Dillig, Shuvendu Lahiri // arXiv preprint arXiv:1802.06551. — 2018.

108. Chen, Jia. Program Analysis Techniques for Algorithmic Complexity and Relational Properties: Ph.D. thesis. — 2019.

109. Beckert, Bernhard. Using Relational Verification for Program Slicing / Bernhard Beckert, Thorsten Bormer, Stephan Gocht, Mihai Herda, Daniel Lentzsch, Mattias Ulbrich // International Conference on Software Engineering and Formal Methods / Springer. — 2019. — P. 353–372.

110. Sirone, Deepak. Improving Lazy Self-Composition for Secure Information Flow: Ph.D. thesis / Indian Institute of Technology Kanpur. — 2019.

111. Almeida, José Bacelar. Verifying Constant-Time Implementations / José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, François Dupressoir, Michael Emmi // USENIX. — USENIX Association, 2016. — P. 53–70.

112. Farzan, Azadeh. Automated Hypersafety Verification / Azadeh Farzan, Anthony Vandikas // Computer Aided Verification / Ed. by Isil Dillig, Serdar Tasiran. — Springer International Publishing, 2019. — P. 200–218.

113. Sousa, Marcelo. Cartesian Hoare Logic for verifying k-safety properties / Marcelo Sousa, Isil Dillig // PLDI. — ACM, 2016. — P. 57–69.

114. Chen, Jia. Precise Detection of Side-Channel Vulnerabilities using Quantitative Cartesian Hoare Logic / Jia Chen, Yu Feng, Isil Dillig // CCS / Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, Dongyan Xu. — ACM, 2017. — P. 875–890.

115. Pick, Lauren. Exploiting Synchrony and Symmetry in Relational Verification / Lauren Pick, Grigory Fedyukovich, Aarti Gupta // LNCS, CAV, Part I. — Vol. 10981. — Springer, 2018. — P. 164–182.

116. Yang, Hongseok. Relational Separation Logic / Hongseok Yang // Theoretical Computer Science. — 2007. — Vol. 375, no. 1-3. — P. 308–334.

117. Dietsch, Daniel. Ultimate TreeAutomizer (CHC-COMP Tool Description) / Daniel Dietsch, Matthias Heizmann, Jochen Hoenicke, Alexander Nutz, Andreas Podelski // Proceedings of the Sixth Workshop on Horn Clauses for Verification and Synthesis and Third Workshop on Program Equivalence and Relational Reasoning, HCVS/PERR@ETAPS 2019, Prague, Czech Republic, 6-7th April 2019. — 2019. — P. 42–47.

118. Jovanovic, Dejan. Property-Directed k-Induction / Dejan Jovanovic, Bruno Dutertre // FMCAD. — IEEE, 2016. — P. 85–92.

119. Kafle, Bishoksan. Constraint Specialisation in Horn Clause Verification / Bishoksan Kafle, John P Gallagher // Science of Computer Programming. — 2017. — Vol. 137. — P. 125–140.

120. Bjørner, Nikolaj. Horn Clause Solvers for Program Verification / Nikolaj Bjørner, Arie Gurfinkel, Ken McMillan, Andrey Rybalchenko. — 2015. — 09. — P. 24–51.

121. Syntax-Guided Synthesis / Rajeev Alur, Rastislav Bodík, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, Abhishek Udupa // FMCAD. — IEEE, 2013. — P. 1–17.

122. Clarke, Edmund M. Counterexample-Guided Abstraction Refinement / Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, Helmut Veith // LNCS, CAV. — Vol. 1855. — Springer, 2000. — P. 154–169.

123. Хорошилов, М. Мандрыкин и В. Мутилин и А. Введение в метод CEGAR — уточнение абстракции по контрпримерам / М. Мандрыкин и В. Мутилин и А. Хорошилов // Труды Института системного программирования РАН. — 2018. — Vol. 24, no. 0.

124. Cimatti, Alessandro. IC3 Modulo Theories via Implicit Predicate Abstraction / Alessandro Cimatti, Alberto Griggio, Sergio Mover, Stefano Tonetta // LNCS, TACAS. — Vol. 8413. — Springer, 2014. — P. 46–61.

125. Hoder, Krystof. Generalized Property Directed Reachability / Krystof Hoder, Nikolaj Bjørner // LNCS, SAT. — Vol. 7317. — Springer, 2012. — P. 157–171.

126. TOOLympics 2019: An Overview of Competitions in Formal Methods / Ezio Bartocci, Dirk Beyer, Paul E. Black, Grigory Fedyukovich, Hubert Garavel, Arnd Hartmanns, Marieke Huisman, Fabrice Kordon, Julian Nagele et al. // Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III. — 2019. — P. 3–24.

127. Shoham, Sharon. Undecidability of Inferring Linear Integer Invariants / Sharon Shoham // arXiv preprint arXiv:1812.01069. — 2018.

128. Padon, Oded. Decidability of Inferring Inductive Invariants / Oded Padon, Neil Immerman, Sharon Shoham, Aleksandr Karbyshev, Mooly Sagiv // ACM SIGPLAN Notices / ACM. — Vol. 51. — 2016. — P. 217–231.

129. Scott, Dana S. Domains for Denotational Semantics / Dana S Scott // International Colloquium on Automata, Languages, and Programming / Springer. — 1982. — P. 577–610.

130. Clarke, Edmund M. Program Invariants as Fixedpoints / Edmund M. Clarke // Computing. — 1979. — Vol. 21, no. 4. — P. 273–294.

131. Виленкин, Наум Яковлевич. Комбинаторика / Наум Яковлевич Виленкин. — Наука, 1969.

132. Dean, Jeffrey. MapReduce: Simplified Data Processing on Large Clusters / Jeffrey Dean, Sanjay Ghemawat // Communications of the ACM. — 2008. — Vol. 51, no. 1. — P. 107–113.

133. Цейтин, Григорий Самуилович. О сложности вывода в исчислении высказываний / Григорий Самуилович Цейтин // Записки научных семинаров ПОМИ. — 1968. — Vol. 8, no. 0. — P. 234–259.

134. Craig, William. Three Uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory / William Craig // The Journal of Symbolic Logic. — 1957. — Vol. 22, no. 3. — P. 269–285.

135. Cooper, David C. Theorem Proving in Arithmetic without Multiplication / David C Cooper // Machine intelligence. — 1972. — Vol. 7, no. 91-99. — P. 300.

136. Bjørner, Nikolaj. Playing with Quantified Satisfaction. / Nikolaj Bjørner, Mikolás Janota // LPAR (short papers). — 2015. — Vol. 35. — P. 15–27.

137. Fedyukovich, Grigory. Lazy but Effective Functional Synthesis / Grigory Fedyukovich, Arie Gurfinkel, Aarti Gupta // International Conference on Verification, Model Checking, and Abstract Interpretation / Springer. — 2019. — P. 92–113.

138. Bonet, María Luisa. Resolution for MAX-SAT / María Luisa Bonet, Jordi Levy, Felip Manyà // Artificial Intelligence. — 2007. — Vol. 171, no. 8-9. — P. 606–618.

139. Bjørner, Nikolaj. νZ-Maximal Satisfaction with Z3. / Nikolaj Bjørner, Anh-Dung Phan // Scss. — 2014. — Vol. 30. — P. 1–9.

140. Barrett, Clark. The SMT-LIB Standard: Version 2.0 / Clark Barrett, Aaron Stump, Cesare Tinelli et al. // Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, England). — Vol. 13. — 2010. — P. 14.

141. Gent, Ian P. Watched Literals for Constraint Propagation in Minion / Ian P Gent, Chris Jefferson, Ian Miguel // International Conference on Principles and Practice of Constraint Programming / Springer. — 2006. — P. 182–197.

142. de Moura, Leonardo. Model-Based Theory Combination / Leonardo de Moura, Nikolaj Bjørner // Electronic Notes in Theoretical Computer Science. — 2008. — Vol. 198, no. 2. — P. 37–49.

143. De Moura, Leonardo. Efficient E-matching for SMT Solvers / Leonardo De Moura, Nikolaj Bjørner // International Conference on Automated Deduction / Springer. — 2007. — P. 183–198.

144. Karr, Michael. Affine Relationships among Variables of a Program / Michael Karr // Acta informatica. — 1976. — Vol. 6, no. 2. — P. 133–151.

145. Bjørner, Nikolaj. Automatic Generation of Invariants and Intermediate Assertions / Nikolaj Bjørner, Anca Browne, Zohar Manna // Theoretical Computer Science. — 1997. — Vol. 173, no. 1. — P. 49–87.

146. Weiser, M. Program Slicing / M. Weiser // ICSE. — IEEE, 1981. — P. 439–449.

147. McMillan, Kenneth. Computing Relational Fixed Points using Interpolation / Kenneth McMillan, Andrey Rybalchenko // Technical report, Technical report, 2012. — 2013.

148. Sato, Ryosuke. Towards a Scalable Software Model Checker for Higher-Order Programs / Ryosuke Sato, Hiroshi Unno, Naoki Kobayashi // Proceedings of the ACM SIGPLAN 2013 workshop on Partial evaluation and program manipulation. — 2013. — P. 53–62.

149. Mordvinov, Dmitry. Verifying Safety of Functional Programs with Rosette/Unbound / Dmitry Mordvinov, Grigory Fedyukovich // CoRR. — 2017. — Vol. abs/1704.04558. — https://github.com/dvvrd/rosette.

Список рисунков

Список таблиц