На правах рукописи

Подкопаев Антон Викторович

ОПЕРАЦИОННЫЕ МЕТОДЫ В ПРИЛОЖЕНИИ К СЛАБЫМ МОДЕЛЯМ ПАМЯТИ

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

Автореферат
диссертации на соискание учёной степени
кандидата физико-математических наук

Санкт-Петербург — 2018

Работа выполнена на кафедре системного программирования Санкт-Петербургского государственного университета.

Научный руководитель: **КОЗНОВ Дмитрий Владимирович,** доктор технических наук, доцент, профессор кафедры системного программирования

Официальные оппоненты: **ГЕРГЕЛЬ Виктор Павлович,**
доктор технических наук, профессор,
Федеральное государственное автономное образовательное учреждение высшего образования «Нижегородский государственный университет им. Н.И. Лобачевского»,
директор института информационных технологий, математики и механики, заведующий кафедрой программной инженерии

ЛУКАШИН Алексей Андреевич,
кандидат технических наук,
Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»,
доцент кафедры «Телематика (при ЦНИИ РТК)» института прикладной математики и механики

Ведущая организация: Федеральное государственное бюджетное учреждение науки Институт системного программирования Российской академии наук (ИСП РАН)

Защита состоится 17 мая 2018 г. в 15 часов на заседании диссертационного совета Д 212.232.51 на базе Санкт-Петербургского государственного университета по адресу: 198504, Санкт-Петербург, Старый Петергоф, Университетский пр. 28, математико-механический факультет СПбГУ, ауд. 405.

С диссертацией можно ознакомиться в библиотеке Санкт-Петербургского государственного университета по адресу: 199034, Санкт-Петербург, Университетская наб., д. 7/9, а также на сайте СПбГУ: `http://spbu.ru/science/disser/`.

Автореферат разослан __ _____ 20__ года.

Ученый секретарь
диссертационного совета
Д 212.232.51,
д.ф.-м.н., профессор Демьянович Юрий Казимирович

Общая характеристика работы

Актуальность темы. Многопоточные программы являются источниками специфических, трудно обнаружимых ошибок, таких как состояние гонки, взаимная блокировка и др. Эти ошибки могут воспроизводиться очень редко, например, в одном из 10 тысяч запусков программы, однако даже это может быть критично. В связи с этим необходимо иметь специализированные методы для анализа многопоточных программ. Ключевым аспектом любого анализа программ является наличие хорошо определенной семантики языка программирования. Семантики языков программирования и систем (процессоров) с многопоточностью называют *моделями памяти* (memory model, MM).

Наиболее простой и естественной моделью памяти является *последовательная консистентность* (sequential consistency, SC); она подразумевает, что каждый сценарий поведения многопоточной программы может быть получен некоторым поочередным исполнением команд её потоков на одном ядре (процессоре). Однако эта модель не способна описать все сценарии поведения, встречаемые на практике. Сценарии поведения, которые не могут быть описаны моделью SC, называются *слабыми*.

Слабыми сценариями поведения обладают некоторые многопоточные программы с неблокирующей синхронизацией потоков. Это является следствием обработки программ оптимизирующими компиляторами и их исполнением на суперскалярных процессорах. Поскольку алгоритмы на базе неблокирующей синхронизации всё чаще используются при разработке важных и высокопроизводительных систем, таких как ядро Linux и системы управления базами данных, то слабые сценарии поведения требуют тщательного изучения.

Модели памяти, допускающие слабые сценарии поведения программ, называются *слабыми* (weak memory models). На данный момент научное сообщество в тесном сотрудничестве с индустрией разработало и продолжает совершенствовать множество таких моделей для языков программирования и процессорных архитектур. При этом процессорные и языковые модели существенно влияют на друг друга. Так, модель процессора должна отражать сценарии поведения, наблюдаемые при запуске программ на существующих процессорах, и оставлять возможности для развития обратно совместимых архитектур. В то же время, языковая модель должна предоставлять разумные и удобные абстракции для программиста, а также допускать основные компиляторные оптимизации и быть совместимой с моделям целевых архитектур, т.е. поддерживать эффективную трансляцию в низкоуровневый код без изменения семантики программы.

Существующие модели памяти для наиболее популярных языков программирования обладают рядом недостатков. Так, известно, что модель памяти Java некорректна по отношению к базовым оптимизациям, а модель памяти C/C++11 разрешает сценарии поведения программ, в которых появляются “значения из воздуха” (out-of-thin-air values). Модель памяти имеет проблему “значений из воздуха”, если для программы без арифметики, в которой явным образом не

встречается некоторая константа, (например, 42) допустим сценарий поведения, в котором эта константа появляется (например, записывается в память или читается из памяти). Такие сценарии не проявляются на практике, но тот факт, что модель C/C++11 их допускает, не позволяет формально доказывать многие полезные свойства программ в рамках этой модели. Наличие проблемы "значений из воздуха" связано с тем, что модель C/C++11 задана декларативно (аксиоматически), при этом сценарий поведения программы в рамках модели определяется как некоторая монолитная структура (граф), а не как последовательность действий некоторой абстрактной машины. Это оставляет открытым вопрос об интеграции модели с остальными компонентами языка, которые в стандарте C/C++11 определены операционно.

Таким образом, для развития инструментов анализа многопоточных программ необходимо разработать операционные подходы к заданию слабых моделей памяти.

Степень разработанности темы. С 1990-х годов велась работа по разработке семантики многопоточности с учетом слабых сценариев поведения. Формальные модели для наиболее распространенных процессорных архитектур (x86, Power, ARM) были разработаны J. Alglave, S. Ishtiaq, L. Maranget, F. Zappa Nardelli, S. Sarkar, P. Sewell и др. исследователями. Новые версии моделей продолжают появляться в связи с развитием процессорных архитектур. В частности, в 2016 и 2017 годах были представлены модели памяти для архитектур ARMv8.0 и ARMv8.3. В 1995 году была стандартизована слабая модель памяти для языка Java; в дальнейшем модель существенно менялась вплоть до 2005 года. В 2011 году появилась аксиоматическая модель памяти для языков C/C++.

В 2017 году исследователи J. Kang, C.-K. Hur, O. Lahav, V. Vafeiadis и D. Dreyer представили обещающую модель памяти (promising memory model, Promise), которая является перспективным решением проблемы задания семантики для языка с многопоточностью. Авторы доказали, что модель допускает большинство необходимых оптимизаций, а также показали корректность эффективных схем компиляции в архитектуры x86 и Power. Открытым остался вопрос о корректности компиляции в архитектуру ARM.

Целью данной работы является исследование применимости операционных подходов для описания реалистичных моделей памяти и анализа многопоточных программ на примере языков C/C++.

Для достижения поставленной цели были сформулированы следующие **задачи**.

1. Разработать операционную модель памяти C/C++11, свободную от проблемы "значений из воздуха".
2. Доказать корректность эффективной схемы компиляции из существенного подмножества обещающей модели в операционную модель памяти ARMv8 POP.

3. Доказать корректность эффективной схемы компиляции из существенного подмножества обещающей модели в аксиоматическую модель памяти ARMv8.3.

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); модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования (пункт 8).

Методология и методы исследования. Методология исследования базируется на подходах информатики к описанию и анализу формальных семантик языков программирования.

В работе используется представление операционной семантики программы с помощью помеченной системы переходов, а также метода вычислительных контекстов, предложенного M. Felleisen. Для доказательств корректности компиляции используется техника прямой симуляции. Программная реализация интерпретатора операционной модели памяти C/C++11 выполнена на языке Racket с использованием предметно-ориентированного расширения PLT/Redex.

Основные положения, выносимые на защиту.

1. Предложена операционная модель памяти C/C++11, для этой модели реализован интерпретатор.
2. Доказана корректность компиляции из существенного подмножества обещающей модели в операционную модель памяти ARMv8 POP.
3. Доказана корректность компиляции из существенного подмножества обещающей модели в аксиоматическую модель памяти ARMv8.3.

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

1. Альтернативная модель памяти для стандарта C/C++11, предложенная в работе, отличается от обещающей модели памяти тем, что является запускаемой, т.е. для нее возможно создание интерпретатора (что и было выполнено в рамках данной диссертационной работы). Это отличие является следствием того, что для получения эффекта отложенного чтения предложенная модель использует синтаксический подход (буферизация инструкций), тогда как обещающая модель — семантический (обещание потоком сделать запись в будущем).
2. Доказательство корректности компиляции из обещающей модели памяти в аксиоматическую модель ARMv8.3 не опирается на специфические свойства целевой модели, такие как представимость модели в виде набора оптимизаций поверх более строгой модели. Это отличает его от аналогичных доказательств для моделей x86 и Power (работы O. Lahav, V.Vafeiadis и других).

3. Доказательства корректности компиляции из обещающей модели памяти в модели ARMv8 POP и ARMv8.3, представленные в работе, являются первыми результатами о компиляции для данных моделей.

Теоретическая и практическая значимость работы. Диссертационное исследование предлагает новый операционный способ представления реалистичной семантики многопоточности с помощью меток времени и фронтов, который может быть полезен при верификации многопоточных программ с неблокирующей синхронизацией, а также при анализе реализации примитивов блокирующей синхронизации.

Предложенный в диссертационном исследовании метод доказательства корректности компиляции из обещающей в аксиоматические модели памяти может быть использован для доказательств корректности компиляции из обещающей модели в архитектуры других процессоров. Последнее актуально в свете того, что в комитетах по стандартизации языков C и C++ активно обсуждается вопрос о смене модели памяти, и обещающая модель является одной из возможных альтернатив.

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

Основные результаты работы докладывались на следующих научных конференциях и семинарах: внутренний семинар ННГУ им. Лобачевского (13 декабря 2017, Нижний Новгород, Россия), открытая конференция ИСП РАН им. В.П. Иванникова (30 ноября–1 декабря 2017, РАН, Москва, Россия), семинар “Технологии разработки и анализа программ” (16 ноября 2017, ИСП РАН, Москва, Россия), внутренние семинары School of Computing of the University of Kent (август 2017, Кентербери, Великобритания), внутренние семинары Department of Computer Science of UCL (август 2017, Лондон, Великобритания), внутренние семинары MPI-SWS (май 2017, Кайзерслаутерн, Германия), The European Conference on Object-Oriented Programming (ECOOP, 18–23 июня 2017, Барселона, Испания), конференция “Языки программирования и компиляторы” (PLC, 3–5 апреля 2017, Ростов-на-Дону, Россия), Verified Trustworthy Software Systems workshop (VTSS, 4-7 апреля 2016, Лондон, Великобритания), POPL 2016 Student Research Competition (21 января 2016, Санкт-Петербург, Флорида, США).

Публикации по теме диссертации. Основные результаты по теме диссертации изложены в пяти 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. Одна статья опубликована в издании, входящем в базы цитирования SCOPUS и Web of Science.

Личный вклад автора в публикациях, выполненных с соавторами, распределён следующим образом. В статьях [1, 2] автор предложил схему доказательства корректности компиляции для аксиоматических семантик и выполнил само доказательство для модели ARMv8.3; соавторы участвовали в обсуждении основных идей доказательства. В работах [3, 4] автор выполнил формализацию семантики ARMv8 POP и доказал корректность компиляции методом симуляции; соавторы участвовали в обсуждении идей доказательства и редактировали тексты статей. В работе [5] личный вклад автора заключается в предложении идеи меток времени и фронтов как способа операционного задания модели памяти, а также в создании компонентного метода задания семантики и реализации интерпретатора; соавторы предложили синтаксический способ обработки отложенных операций.

Объем и структура работы. Диссертация состоит из введения, четырех глав, заключения и приложения. Полный объем диссертации **190** страниц текста с **29** рисунками и **4** таблицами. Список литературы содержит **109** наименований.

Содержание работы

Во **введении** обосновывается актуальность исследований, выполненных в рамках данной диссертационной работы, приводится краткий обзор научной литературы по изучаемой проблеме, формулируется цель, ставятся задачи работы, излагается научная новизна и практическая значимость представленного исследования.

Первая глава посвящена обзору области исследования. Рассматриваются требования к реалистичным моделями памяти языков программирования, предъявляемые через призму наблюдаемых сценариев поведения многопоточных программ, применяемых компиляторами оптимизаций, а также моделей памяти процессорных архитектур. Описывается модель памяти C/C++11. Рассматриваются проблемы модели C/C++11, в том числе проблема “значений из воздуха”. Приводится описание существующих слабых моделей памяти без “значений из воздуха”, в частности, обещающей модели. На основе выполненного обзора делаются следующие выводы.

- Модель памяти промышленного языка программирования должна удовлетворять, как минимум, трём критериям. Во-первых, должна существовать корректная схема компиляции в модель целевой процессорной архитектуры. Во-вторых, основные компиляторные оптимизации должны быть корректны в рамках модели. В-третьих, у модели должна отсутствовать проблема “значений из воздуха”.
- При разработке новой модели памяти языка программирования нужно доказывать корректность эффективной компиляции в модели памяти целевых процессорных архитектур.
- Существующие модели памяти промышленных языков программирования не удовлетворяют всем приведённым выше критериям.

– Требуется разработать операционную модель памяти с синтаксисом модели C/C++11, которая не имеет проблемы “значений из воздуха”.

Вторая глава посвящена описанию предложенной в диссертации операционной модели памяти OpC11 для C/C++. Модель представлена в виде операционной семантики малого шага с помощью редукционных контекстов. Основное отличие слабых моделей памяти от модели последовательной консистентности заключается в том, что первые не гарантируют для локации в памяти единственность значения, которое может быть прочитано в каждый конкретный момент времени. Так, например, следующая программа может завершиться с результатом $[a = 1, b = 0]$, хотя, казалось бы, $a = 1$ гарантирует, что в локацию d уже записано новое значение 239:

$$\begin{array}{c|c|c} [f] := 0; & & [d] := 0; \\ [d] := 239; & \| & a := [f]; \\ [f] := 1 & & b := [d] \end{array}$$

Как следствие, оперативная память (далее просто “память”) в рамках слабых моделей памяти не может быть представлена как функция из локации в значения.

Память в модели OpC11 представляется как множество *сообщений*. Каждое сообщение содержит целевую локацию, записываемое значение и *метку времени* — натуральное число, которое определяет полный порядок на сообщениях, относящихся к одной локации. Последнее нужно для того, чтобы гарантировать последовательную консистентность для программ, оперирующих только над одной локацией — эту гарантию предоставляют большинство слабых моделей памяти, в том числе модель C/C++11. При выполнении инструкции чтения из некоторой локации поток может недетерминировано выбрать сообщение, относящееся к этой локации, и выполнить из него чтение. чтение из него.

Недетерминированность чтения из локации ограничена гарантией, которую также предоставляет модель C/C++11: после того, как поток прочитал или записал сообщение в локацию x с меткой времени t, он (поток) больше не может читать из сообщений с меткой времени, которая меньше t. Для реализации данного ограничения в модели OpC11 у каждого потока есть т.н. *базовый фронт* (current view, current viewfront) — функция из локаций в метки времени, определяющая осведомленность потока о сообщениях в памяти.

Некоторые программы имеют слабые сценарии поведения, разрешенные моделью C/C++11, которые не могут быть смоделированы только недетерминированной памятью, а также требуют исполнения инструкций не по порядку. Например, следующая программа может завершиться с результатом $[a = 1, b = 1]$ в модели C/C++11:

$$\begin{array}{c|c|c} [x] := 0; & & [y] := 0; \\ a := [x]; & \| & b := [y]; \\ [y] := 1 & & [x] := 1 \end{array}$$

Для представления таких сценариев поведения в модели OpC11 у каждого потока есть *буфер отложенных операций*. Так, модель позволяет потоку в каждый

момент отложить текущую операцию вместо её выполнения. С помощью этого механизма модель OpC11 может исполнить приведенную выше программу и получить результат $a = 1, b = 1$ следующим образом. Сначала левый поток откладывает чтение из локации x и выполняет запись в y. После этого правый поток читает из вновь добавленного сообщения и записывает 1 в x. Далее левый поток исполняет отложенное чтение из сообщения, добавленного правым потоком, и получает результат $[a = 1, b = 1]$.

Для поддержки высвобождающих (release) барьеров и записей, а также приобретающих (acquire) барьеров и чтений модель OpC11 использует дополнительные фронты — высвобождающий и приобретающий для каждого потока, а также фронт сообщений для каждого сообщения в памяти. Для поддержки чтений с модификатором доступа consume модель использует динамическую пометку инструкций, зависимых от consume-чтения.

Для предложенной модели был реализован интерпретатор на языке Racket с помощью библиотеки описания редукционных семантик PLT/Redex. Код проекта доступен по адресу `github.com/anlun/OperationalSemanticsC11`.

Апробация предложенной семантики была выполнена на наборе, состоящем более чем из 40 тестов (litmus tests), взятых из тематической литературы, а также на алгоритме RCU (Read-Copy-Update). Поведение модели OpC11 совпадает с поведением модели C/C++11 на большинстве этих тестов. Отличие наблюдается на двух следующих категориях тестов. Первая категория — это программы, которые имеют исполнения со “значениями из воздуха” в рамках модели C/C++11. Для таких тестов модель OpC11 не выдает исполнения со “значениями из воздуха”, что является её положительным свойством. Вторая категория — это программы, в которых существуют антизависимости по управлению, адресу или значению, ведущие к инструкциям записи. На таких программах OpC11 не способна получить все возможные в рамках C/C++11 исполнения, поскольку выполнение инструкций не по порядку в OpC11 реализовано синтаксическим способом. Этот недостаток модели не позволяет ей поддержать все необходимые компиляторные оптимизации.

Одновременно с моделью памяти OpC11 исследователями J. Kang, C.-K. Hur, O. Lahav, V. Vafeiadis и D. Dreyer была представлена обещающая модель памяти, которая очень близка OpC11, но использует другой механизм для выполнения инструкций не по порядку. Этот механизм позволяет поддержать больше компиляторных оптимизаций, чем модель OpC11. Из-за данного преимущества диссертант принял решение продолжить свою исследовательскую работу в рамках обещающей модели.

В **третьей главе** приводится описание обещающая модель памяти и операционной модели ARMv8 POP, а также представлено доказательство корректности компиляции из существенного подмножества обещающей модели памяти в модель ARMv8 POP.

Обещающая модель памяти является операционной моделью для синтаксиса модели C/C++11. Она использует те же базовые понятия, что и предложенная диссертантом модель OpC11, метки времени и фронты, однако вместо механизма откладывания выполнения инструкций она, в соответствии со своим названием, использует механизм *обещаний*. Так, в каждый момент исполнения поток обещающей модели памяти может совершить одно из двух действий: либо выполнить следующую инструкцию, либо пообещать сделать запись в локацию. Последнее может быть выполнено вне зависимости от того, какая инструкция является следующей. Если поток выбирает пообещать сделать запись в локацию, то он добавляет соответствующее сообщение в память, делая это сообщение видимым для других потоков. Далее в ходе исполнения поток должен будет выполнить соответствующую инструкцию записи, таким образом выполняя сделанное ранее обещание.

Для того, чтобы запретить "значения из воздуха", после каждого исполненного шага каждый поток должен выполнить т.н. *сертификацию* — предъявить, что он может быть локально исполнен таким образом, что выполнит все оставшиеся обещания. Известно, что задача сертификации является алгоритмически неразрешимой для языков, полных по Тьюрингу. Следовательно, для обещающей модели невозможно разработать интерпретатор, что является её недостатком по сравнению с представленной в диссертации моделью OpC11.

Несмотря на этот недостаток, обещающая модель обладает рядом существенных достоинств. В частности, обещающая модель не имеет проблемы "значений из воздуха", что делает возможным для неё разработать выразительную программную логику. Также для модели была доказана корректность существенного класса компиляторных оптимизаций и корректность эффективной компиляции в модели памяти процессоров x86 и Power.

Открытой проблемой является доказательство корректности компиляции из обещающей модели памяти в модели памяти архитектуры ARM, которая, наравне с x86 и Power, является одной из наиболее распространённых процессорных архитектур на данный момент.

Здесь и далее под корректностью компиляции понимается следующее утверждение.

Определение. Для языков L и L' с моделями памяти M и M' соответственно схема компиляции $\mathsf{compile} : L \to L'$ называется *корректной*, если выполняется следующее условие:

$$\forall Prog \in L.\ [\![\mathsf{compile}(Prog)]\!]_{M'} \subseteq [\![Prog]\!]_M,$$

где $[\![Prog]\!]_M$ — множество результатов сценариев поведения программы $Prog$ в модели памяти M. В доказательствах корректности компиляции в ARMv8 POP и ARMv8.3 результатом сценария поведения считается финальное состояние памяти.

Рассмотренное подмножество обещающей модели памяти (Promise) состоит из расслабленных (relaxed, rlx) записей и чтений, а также высвобождаю-

щих (release, rel) и приобретающих (acquire, acq) барьеров памяти. При этом подразумевается следующая схема компиляции:

Promise:	$[x]_{\mathsf{rlx}} := a$	$a := [x]_{\mathsf{rlx}}$	$\mathsf{fence}(\mathtt{acq})$	$\mathsf{fence}(\mathtt{rel})$
ARMv8 POP:	$[x] := a$	$a := [x]$	$\mathsf{fence}(\mathtt{ld})$	$\mathsf{fence}(\mathtt{sy})$

Первый и второй столбцы подразумевают, что расслабленные операции записи и чтения из языка, на котором определена обещающая модель, переходят в обычные операции записи и чтения в терминах ARMv8-ассемблера, а приобретающий и высвобождающий барьеры — в барьер по чтению и в полный барьер. Такая схема компиляции считается эффективной и применяется в компиляторах GCC и LLVM. Поскольку схема компиляции в данном случае является биекцией, то далее в этой главе предполагается, что язык задания обещающей и ARMv8 POP моделей совпадает.

Основной результат главы сформулирован следующим образом.

Теорема. Для любой программы *Prog* на языке задания модели и её сценария поведения в модели ARMv8 POP существует такой сценарий поведения *Prog* в обещающей модели, что финальное состояние памяти в сценариях поведения совпадает.

В рамках доказательства теоремы по сценарию поведения программы в модели ARMv8 POP строится сценарий поведения в обещающей модели. Поскольку обе модели заданы операционным способом, то существуют две абстрактные машины, которые представляют данные модели. Обычно для решения задачи построения сценария поведения одной машины по сценарию другой используют технику симуляции, которая является специальной формой индукции. В рамках данной техники вводится отношение симуляции, которое связывает состояния машин, и доказываются две следующие леммы: отношение симуляции связывает начальные состояния машин (база индукции); для любого шага симулируемой машины существует ноль или более шагов симулирующей машины, после выполнения которых новые состояния машин опять связаны отношением симуляции (индукционный переход).

В доказательстве индукционного перехода сложным является то, что между моделями имеется два существенных различия. Во-первых, обещающая модель может исполнять не по порядку только инструкции записи, в то время как модель ARMv8 POP может исполнять инструкции в несколько шагов, не по порядку и спекулятивно. Во-вторых, в обещающей модели в тот момент, когда сообщение попадает в память, этому сообщению присваивается некоторая метка времени, которая служит его порядковым номером в множестве сообщений, относящихся к той же локации. В модели ARMv8 POP меток времени нет и порядок сообщений одной локации определяется не сразу после того, как сообщения попадут в её подсистему памяти и станут видимыми для других потоков.

Для того, чтобы обойти первое различие, автор использовал технику “запаздывающей” симуляции. В рамках данной техники отношение симуляции представляется как объединение двух взаимоисключающих отношений, напри-

мер, A и B. Далее индукционный переход формулируется следующим образом. Если состояние симулируемой машины x связано с состояние симулирующей машины y отношением A, т.е. выполняется $(x,y) \in A$, то для любого состояния x', в которое может перейти симулируемая машина, выполняется $(x', y) \in A \cup B$. С другой стороны, если $(x, y) \in B$, то существует состояние y', в которое может перейти симулирующая машина, что $(x, y') \in A \cup B$. Тогда при условии, что не существует такой бесконечной цепочки $\{y'_i\}_{i \in \mathbb{N}}$, что y'_i переходит y'_{i+1} и $(x, y'_i) \in B$ для всех i, из нового варианта индукционного перехода следует изначальное утверждение симуляции.

В доказательстве теоремы отношение A символизирует, что обещающая машина ждёт, пока ARM-машина выполнит действие, которое обещающая машина может повторить, а отношение B означает, что обещающая машина может симулировать несколько действий, уже выполненных ARM-машиной.

Для того, чтобы снять второе различие между обещающей и ARMv8 POP моделями, в доказательстве определяется ограниченная версия ARM-машины, которая добавляет метки времени к сообщениям записи в подсистеме памяти, тем самым определяя порядок на сообщениях к одной локации раньше, чем это делает обычная ARMv8 POP модель. Это изменение также добавляет дополнительные ограничения на сценарии поведения ARM-машины. Тем не менее, автор приводит доказательство того, что новая модель эквивалентна исходной, что позволяет свести доказательство теоремы к доказательству корректности компиляции из обещающей модели в модифицированную ARMv8 POP модель.

В **четвертой главе** обсуждается аксиоматическая модель памяти ARMv8.3. Приводятся рассуждения о том, почему метод доказательства корректности компиляции из обещающей модели памяти, использованный её авторами для аксиоматических моделей архитектур x86 и Power, не подходит для модели ARMv8.3. Далее приводится доказательство корректности компиляции из обещающей модели памяти в подмножество модели ARMv8.3. Доказательство основано на построении операционной семантики обхода аксиоматических сценариев поведения программ в модели ARMv8.3.

В рамках аксиоматической (или декларативной) модели памяти сценарий поведения программы представляется в виде графа, в котором вершинами являются *события* (операции над памятью), а ребрами — различные отношения на событиях, такие как программный порядок, отношение “читает из” и др. При этом граф считается согласованным с моделью, если выполняются *аксиомы* модели, которые обычно формулируются как наличие некоторого полного порядка на подмножестве событий или отсутствие в графе путей определённого типа.

При доказательстве корректности компиляции из обещающей модели в аксиоматическую техника симуляции напрямую неприменима, поскольку сценарий поведения в аксиоматической модели не является последовательностью шагов исполнения некоторой абстрактной машины. Поэтому в доказательстве корректности компиляции в модели x86 и Power авторы использовали другой метод. Этот метод состоит из двух частей. Во-первых, доказывается, что модели

x86 и Power могут быть представлены как набор программных оптимизаций поверх более простых моделей. Эти оптимизации являются доказано корректными в рамках обещающей модели, из чего следует, что доказательство корректности компиляции может быть сведено к аналогичному доказательству для более простых моделей. Далее показывается, что эти более простые модели могут быть симулированы моделью, которая является аксиоматическим аналогом обещающей модели без механизма обещаний.

Автору работы не удалось применить такой подход для модели ARMv8.3, поскольку эта модель не представима как набор тех же оптимизаций над упрощенной моделью. Поэтому был разработан альтернативный подход, который заключается в построении операционной семантики обхода аксиоматических сценариев поведения, которая может быть непосредственно симулирована обещающей моделью.

Обходом в диссертационном исследовании называется последовательность переходов между *конфигурациями исполнения* — упорядоченными парами подмножеств вершин сценария поведения $\langle C, I\rangle$. Подмножество C называется *множеством покрытых событий*, а подмножество I — *множеством выпущенных событий*; элементы этих подмножеств называются *покрытыми* и *выпущенными* соответственно.

Конфигурация обхода называется *корректной*, если выполняются следующие условия.

- Множество покрытых событий префикс-замкнуто по отношению программного порядка.
- Множество выпущенных событий содержит только события записи.
- Если событие записи покрыто, то оно также является выпущенным.

При доказательстве симуляции обещающей моделью обхода покрытые события будут соответствовать инструкциям, выполненным обещающей машиной, а выпущенные события — сообщениям в памяти обещающей машины.

Шаги обхода задаются следующим образом:

$$\frac{a \in \mathsf{Next}(G, C) \cap \mathsf{Coverable}(G, C, I)}{G \vdash \langle C, I\rangle \to_{\mathrm{TC}} \langle C \cup \{a\}, I\rangle} \qquad \frac{w \in \mathsf{Issuable}(G, C, I) \setminus I}{G \vdash \langle C, I\rangle \to_{\mathrm{TC}} \langle C, I \cup \{w\}\rangle}$$

Здесь первое правило соответствует покрытию события a, а второе — выпуску события w; $\mathsf{Next}(G, C)$ — обозначает множество событий, непосредственно следующих в отношении программного порядка за покрытыми; $\mathsf{Coverable}(G, C, I)$ и $\mathsf{Issuable}(G, C, I)$ — это события, покрываемые и выпускаемые в текущей конфигурации, которые определены в соответствии с требованиями обещающей модели к исполнению инструкции и обещанию сообщения соответственно.

Далее автор работы приводит доказательство следующей теоремы о полноте обхода.

Теорема. Для любого корректного сценария поведения G в модели ARMv8.3 существует обход $\langle W^{\mathrm{init}}, W^{\mathrm{init}}\rangle \to^*_{\mathrm{TC}} \langle E, W\rangle$, где W^{init} — множество инициализи-

рующих записей сценария G, E — все события сценария G, W — все события записи сценария G.

Используя данную теорему для построения операционного исполнения программы в модели ARMv8.3, автор работы доказывает, что обещающая модель может симулировать сценарии поведения модели ARMv8.3.

В **заключении** приведены основные результаты работы.

1. Разработана операционная модель памяти C/C++11. Данная модель допускает такие же сценарии поведения, что и модель C/C++11 на большинстве тестов, приведенных в литературе, но не обладает сценариями поведения со “значениями из воздуха”. В отличие от обещающей модели, предлагаемая модель является запускаемой, что упрощает разработку средств анализа программ для неё. Недостатком модели является то, она накладывает синтаксические ограничения на поведения программ.
2. Доказана корректность компиляции из существенного подмножества обещающей модели в операционную модель памяти ARMv8 POP.
3. Доказана корректность компиляции из существенного подмножества обещающей модели в аксиоматическую модель памяти ARMv8.3.

В рамках **рекомендации по применению результатов работы** в индустрии и научных исследованиях указывается, что модель памяти промышленного языка программирования должна быть лишена сценариев поведения, имеющих “значения из воздуха”, а также либо быть представленной в операционной форме, либо иметь эквивалентный операционный аналог. Последнее позволяет реализовать интерпретатор модели и выполнять отладку программ в рамках модели.

Также были определены **перспективы дальнейшей разработки тематики**, основным из которых является разработка обобщенной аксиоматической модели памяти для процессорных архитектур, которая будет определена для синтаксиса модели C/C++11 и окажется строгим надмножеством существующих моделей памяти x86, Power и ARM, а также для которой будет применим предложенный метод доказательства корректности компиляции из обещающей модели памяти. Это позволит свести дальнейшие доказательства корректности компиляции из обещающей модели к доказательству корректности компиляции в обобщенную аксиоматическую модель, что сводится к рассуждениям об ациклично-сти и вложенности путей на графах. Кроме того, актуальной является задача разработки эффективной программной логики на базе логики многопоточного разделения (concurrent separation logic) для операционного аналога модели памяти C/C++11 и обещающей модели памяти. Такая логика позволит формально доказывать в рамках моделей сложные свойства программ, такие как соответствие спецификации.

Публикации автора по теме диссертации

Ниже приведён перечень публикаций, где были представлены основные результаты данной диссертационной работы.

Статьи 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

1. Подкопаев, А. В. О корректности компиляции подмножества обещающей модели памяти в аксиоматическую модель ARMv8.3 / А.В. Подкопаев, О. Лахав, В. Вафеядис // Научно-технические ведомости СПбГПУ. Информатика, Телекоммуникации. Управление. — 2017. — Т. 10, № 4. — С. 51–69.

2. Подкопаев, А. В. Обещающая компиляция в ARMv8.3 / А.В. Подкопаев, О. Лахав, В. Вафеядис // Труды ИСП РАН. — 2017. — Т. 29, № 5. — С. 149–164.

Статьи в изданиях, входящих в базы цитирования Web of Science и Scopus

3. Podkopaev, A. Promising compilation to ARMv8 POP / A. Podkopaev, O. Lahav, V. Vafeiadis // 31st European Conference on Object-Oriented Programming (ECOOP 17), Leibniz International Proceedings in Informatics (LIPIcs). — 2017. — P. 22:1–22:28.

Статьи в других изданиях

4. Подкопаев, А. В. Обещающая компиляция в ARMv8 / А.В. Подкопаев, О. Лахав, В. Вафеядис // Языки программирования и компиляторы. Труды конференции. Ростов-на-Дону, Россия. — 2017. — С. 223–226.

5. Podkopaev, A. Operational Aspects of C/C++ Concurrency / A. Podkopaev, I. Sergey, A. Nanevski [Электронный ресурс]. — URL: `http://arxiv.org/abs/1606.01400` (дата обращения: 14.11.2017).

Подкопаев Антон Викторович

ОПЕРАЦИОННЫЕ МЕТОДЫ В ПРИЛОЖЕНИИ К СЛАБЫМ МОДЕЛЯМ ПАМЯТИ

Автореф. дис. на соискание ученой степени канд. физ.-мат. наук

Подписано в печать ___.___._____. Заказ № _____
Формат 60×90/16. Усл. печ. л. 1. Тираж 100 экз.
Типография _____