В.С. Буренков (ЗАО «МЦСТ», МГТУ им. Н.Э. Баумана)

V. Burenkov

АНАЛИЗ ПРИМЕНИМОСТИ ФОРМАЛЬНЫХ МЕТОДОВ К ВЕРИФИКАЦИИ ПРОТОКОЛОВ КОГЕРЕНТНОСТИ КЭШ-ПАМЯТИ МАСШТАБИРУЕМЫХ СИСТЕМ

AN ANALYSIS OF APPLICABILITY OF FORMAL METHODS TO VERIFICATION OF CACHE COHERENCE PROTOCOLS OF SCALABLE SYSTEMS

Приведен анализ существующих методов верификации протоколов когерентности кэш-памяти масштабируемых систем. Предложены направления создания формального метода верификации протокола когерентности кэш-памяти сложной системы на кристалле.

This article analyzes existing methods of verification of cache coherence protocols of scalable systems. The paper proposes directions for developing a method for verification of cache coherence protocol of a complex system-on-chip.

Ключевые слова: формальный метод, проверка моделей, дедуктивная верификация, протокол когерентности кэш-памяти.

Keywords: formal method, model checking, deductive verification, cache coherence protocol.

Введение

Современные микропроцессорные системы являются масштабируемыми – для них характерно как увеличение числа процессорных ядер на кристалле, так и объединение кристаллов в кластеры. Масштабируемость систем приводит к необходимости создания формальных методов верификации, способных адаптироваться к ней. С ростом размера систем полностью автоматизируемый формальный метод проверки модели (model

checking) исчерпывает свои возможности из-за комбинаторного взрыва числа состояний.

Существующие формальные подходы к верификации, как правило, либо неприменимы к системам промышленного масштаба, либо требуют огромного объема ручной работы.

1. Основные методы верификации

Формальные методы позволяют получить математическое доказательство соответствия модели верифицируемого объекта его спецификации, т.е. набору свойств, которым он должен удовлетворять. Общей моделью реагирующих систем, в число которых входят протоколы когерентности, позволяющей систематически представлять компоненты систем, их координацию и взаимодействие, является система переходов [1].

Основными подходами к формальной верификации являются верификация с помощью метода *проверки модели* и *дедуктивная верификация*.

Проверка моделей [2] – метод, в ходе применения которого пространство состояний конечной модели верифицируемого протокола систематически исследуется с помощью специальных алгоритмов. К достоинствам метода относятся полная автоматизация и генерация контрпримеров, позволяющих отыскать источник ошибки, основным недостатком является «взрыв числа состояний» при увеличении сложности модели.

Рассмотрим проверку свойств безопасности, выражающихся формулой Gp темпоральной логики линейного времени LTL, где p – утверждение – логическая формула от переменных модели. Если утверждение истинно во всех состояниях модели, то p является инвариантом модели. В соответствии с методом дедуктивной верификации необходимо разработать вспомогательное утверждение φ, являющееся аппроксимацией сверху множества достижимых состояний, а затем показать, что p следует из φ. В основе метода лежит следующее правило вывода [3]:

I1. φ выполняется в начальных состояниях

I2. Все переходы между состояниями сохраняют выполнимость φ

I3. $\varphi \rightarrow p$

$$\frac{}{\mathrm{G}p} \qquad (1)$$

Утверждение φ, удовлетворяющее посылкам I1 и I2, называется *индуктивным утверждением*, или *индуктивным инвариантом*. Индуктивное утверждение всегда является аппроксимацией сверху множества достижимых состояний. Если p – инвариант верифицируемой системы, то всегда существует индуктивное утверждение φ, из которого следует p [3]. Исходное утверждение p является индуктивным лишь в редких случаях. Как правило, верификатору необходимо разработать вспомогательное утверждение и установить истинность посылок I1-I3.

Дедуктивная верификация предоставляет возможность работы с системами с бесконечным числом состояний. В их случае для установления истинности посылок обычно используются средства автоматизированного доказательства теорем – программные инструменты, определяющие истинность формул в заданной логике. Популярными инструментами являются ACL2, PVS, Isabelle. Выразительность логики в общем случае влечет за собой ее неразрешимость, что означает невозможность построения такой автоматической процедуры, которая, приняв на вход формулу, всегда может проверить ее истинность в данной логике. Использование средств автоматизированного доказательства теорем подразумевает взаимодействие с пользователем-экспертом и является сложным творческим процессом. Помимо этого, если доказательство теоремы завершается неудачей, очень сложно на основе этой информации найти ошибку в верифицируемой системе.

В [4] описано применение средства автоматизированного доказательства теорем PVS для параметризованной верификации протокола когерентности FLASH. В ходе доказательства авторы многократно вручную проводили поиск кандидатов на

индуктивные утверждения. В случае невозможности доказать их индуктивность, авторы анализировали причины и находили дополнительные условия, превращающие утверждение в индуктивный инвариант. Данный процесс является очень трудоемким, в связи с чем методы, основанные только на доказательстве теорем, могут найти лишь ограниченное применение при верификации протоколов когерентности.

2. Методы верификации масштабируемых систем

Разработка методов верификации масштабируемых систем может быть осуществлена в нескольких направлениях: 1) развитие методов, основанных на проверке моделей; 2) развитие методов дедуктивной верификации; 3) комбинация методов из первой и второй групп.

Методы верификации протоколов когерентности выпускаемых промышленностью микропроцессорных систем должны удовлетворять ряду требований: возможности проведения верификации за допустимое время, высокому уровню автоматизации, предоставлению информации о причинах ошибок. Ни проверка моделей, ни дедуктивная верификация по отдельности не удовлетворяют этим свойствам, поэтому создание общей инфраструктуры, объединяющей и развивающей методы проверки моделей и дедуктивной верификации, представляется наиболее перспективным направлением разработки новых методов верификации масштабируемых систем.

2.1. Использование абстракции и композиции при проверке моделей

Основными подходами, позволяющими применить метод проверки моделей к верификации масштабируемых систем, являются составление моделей более высокого уровня абстракции и композиционная верификация [2]. Методы абстракции позволяют сократить количество состояний верифицируемой модели и в то же время сохранить интересующие нас свойства исходной модели.

Отношения эквивалентности, гарантирующие, что модели будут вести себя одинаково, зачастую не приводят к существенному сокращению числа состояний, поэтому на практике используют отношения симуляции [2], связывающие модель с ее абстракцией. Симуляция гарантирует, что всякое поведение в исходной модели является также поведением в ее абстракции. Однако абстракция при этом может иметь поведения, которые невозможны в исходной модели.

Абстрактные пространства состояний могут быть получены с помощью методов аппроксимации снизу (under-approximation), удаляющих часть поведений, и методов аппроксимации сверху (over-approximation), добавляющих новые поведения. В итоге в случае аппроксимации снизу ошибка в абстрактной системе будет подразумевать ошибку в исходной системе, а в случае аппроксимации сверху корректность абстрактной системы подразумевает корректность исходной системы. В дальнейшем будут затрагиваться только аппроксимации сверху, называемые также консервативными абстракциями.

Построение абстрактных моделей требует нахождения компромисса между двумя конфликтующими целями: получением абстрактных моделей небольшого размера, которые могут быть верифицированы методом проверки моделей, и получением точных абстрактных моделей. Обычно чем проще модель, тем больше поведений она допускает. Это может привести к появлению ложных контрпримеров, не обнаруживаемых в исходной модели. Выходов из этой ситуации, по крайней мере, два: построение точной абстрактной модели или анализ контрпримера на ложность и модификация абстрактной модели на основании полученной информации (counterexample-guided abstraction refinement). Методы, позволяющие получать точные абстрактные модели (например, на основе абстракции со счетчиками и абстракции среды [5]) в случае сложных протоколов, как правило, приводят к большим моделям.

Идея композиционной верификации [6] заключается в использовании естественной

декомпозиции сложной распределенной системы на взаимодействующие процессы. Подход предполагает проверку процессов по отдельности с некоторым обобщенным окружением (упрощенных моделей) и последующее объединение результатов, сопровождающееся заключением о корректности исходной модели. Композиционный метод должен сопровождаться доказательством того, что упрощенная модель удовлетворяет тем же свойствам, что и исходная модель.

В [7] изложен метод параметризованной верификации протоколов когерентности памяти, основанный на композиционной проверке моделей и реализованный в системе Cadence SMV. Задав свойство, изначально делают попытку проверить его на абстрактной модели, в которой отражены процессы, номера которых содержатся в свойстве, а все остальные процессы заменены абстрактным процессом. В случае получения контрпримера вероятной причиной его появления является сообщение, порожденное абстрактным процессом. Чтобы избавиться от контрпримера, предлагается две стратегии: выделить отправителя сообщения в окружение в виде отдельного процесса или ввести лемму, исключающую некорректную отправку сообщения. Процесс повторяется до тех пор, пока не будут исключены все ложные контрпримеры и проверены все свойства спецификации.

Преимущество метода заключается в отсутствии необходимости предоставлять индуктивные инварианты, поскольку соответствующая информация (множество достижимых состояний) получается путем проверки абстрактных моделей. Тем не менее, объем ручной работы по введению лемм остается существенным. Метод был применен для верификации протокола FLASH.

2.2. Использование методов дедуктивной верификации

Особый интерес в исследованиях представляют методы, позволяющие

автоматически получать индуктивные инварианты. В работах [8, 9] определяется математическая модель $S(N)$, позволяющая представлять протоколы когерентности систем с N узлами (в качестве примера описывается исследование протокола German). Показывается, как для данной модели свести проверку посылок правила (1) к проверке модели $S(N_0)$, где N_0 – некоторая константа. Также приводятся эвристики поиска утверждения-кандидата φ.

Реализация метода предполагает его интеграцию в инструмент проверки моделей. В представленных работах приводится пример описания протоколов на академическом языке SPL, и при этом неясно, каким образом применять метод при верификации с использованием современных инструментов (какие ограничения необходимо наложить на использование соответствующих языков). Поскольку значение N_0 определяется размером локального пространства состояний процесса системы, то для сложных протоколов оно может быть слишком большим. В литературе не отражено применение метода к сложным системам.

По проблеме автоматического построения индуктивных инвариантов ведутся активные исследования [10-12].

3. Метод композиционной верификации

В [13] предлагается метод проверки свойств безопасности протоколов когерентности, работающих в системах с любым числом процессоров. Он базируется на комбинации проверки моделей и доказательства теорем и, в частности, основан на идеях из [7]. В качестве основного инструмента, используемого в процессе верификации, выступает средство автоматизированной проверки моделей Murphi. Метод принимает на вход описание симметричного протокола для N процессоров, отношение переходов которого задано с помощью набора правил. Правило является защищенной командой,

условие которой (защита) определяет, будут ли выполняться действия, отраженные в команде. Верификация состоит из итеративного осуществления следующих действий до тех пор, пока не будут отсутствовать контрпримеры:

1) построение абстрактной модели, которая должна представлять систему с N процессорами. Предлагается синтаксическая абстракция описания модели на языке Murphi;

2) проверка выполнения спецификации на абстрактной модели;

3) анализ полученного контрпримера;

4) модификация абстрактной модели в случае ложности контрпримера (если такая последовательность действий невозможна в исходной модели). Предлагается лемма, которая ограничивает модель. Определяется правило абстрактной модели, вызвавшее ложный переход. Условия, отраженные в лемме, добавляются в защиту найденного правила. Также лемма добавляется как проверяемое свойство-инвариант, которое должно выполняться в каждом состоянии. В случае истинности контрпримера исправляется ошибка в исходной модели.

Особенностью метода является его масштабируемость и использование синтаксической абстракции. Второй аспект позволяет автоматизировать процесс получения абстрактной модели путем создания инструмента, работающего с описанием модели на языке моделирования. При этом не нужно модифицировать средство автоматизированной проверки моделей, а возможно его использование напрямую для проверки абстрактных моделей. Более того, в [14] утверждается, что метод корректен для любого симметричного протокола и любой консервативной процедуры абстракции.

Основная сложность метода заключается в нахождении лемм. Добавление лемм – трудоемкий процесс, занимающий много времени и требующий глубокого понимания протокола. В [14, 15] предлагается способ частичной автоматизации процесса, однако

полная автоматизация не достигается.

В [13] утверждается, что метод может быть использован с любым средством автоматизированной проверки моделей. При этом показан лишь пример описания протокола когерентности German на языке Murphi, полученный непосредственным переводом описания данного протокола из [8]. Не объясняется, каким образом допустимо описывать формальные модели, и какие существуют ограничения, а ограничения существуют, что находит отражение в литературе.

В [16] приводится анализ метода с точки зрения пользователя системы Abster, в которой он реализован. Данная система упоминается в [14, 15] и не находится в открытом доступе. В [16] приводится ряд рекомендаций разработчикам протоколов когерентности и утверждается, что следование данным рекомендациям позволит получать протоколы, совместимые с Abster. Поскольку анализ приводится с пользовательской точки зрения, неясно, какие из указанных ограничений являются фундаментальными ограничениями метода, а какие – ограничениями инструмента Abster.

В протоколах когерентности при сборе когерентных ответов зачастую применяются счетчики числа узлов системы. В [16] это запрещается, что является одним из наиболее актуальных ограничений. Вместо таких счетчиков предлагается использовать битовые векторы с длиной, равной числу узлов в системе. Ограничение объясняется интуитивно: использование счетчика запрещено, т.к. оно предполагает сравнение с параметризованным значением.

В [17] вводится язык первого порядка, описывающий синтаксис системы переходов. В терминах этого языка приводятся преобразования, приводящие к абстрактной модели. Далее показывается консервативность абстракции путем формулировки теоремы, утверждающей, что введенное отношение является отношением симуляции, и неполного ее доказательства. Поскольку для описания моделей в [13] используется язык Murphi (а в

последующих работах, затрагивающих данный метод, используется та же самая модель), необходимо установить соответствие между языком Murphi и математическим языком из [17]. Из данного соответствия уже должны следовать подмножество языка Murphi, на котором допустимо описание формальных моделей протоколов когерентности, а также дополнительные ограничения на конструкции в моделях.

4. Предложенные пути создания метода формальной верификации протокола когерентности системы «Эльбрус-4С»

Планируется применить модификацию метода композиционной верификации к проверке протокола когерентности микропроцессора «Эльбрус-4С». Для этого предполагается разрешить проблемы, описанные в разделе 3, в контексте инструмента проверки моделей Spin и языка описания моделей Promela, при этом повысив уровень автоматизации метода. Выбор Spin обусловлен тем, что он является современным и постоянно развивающимся инструментом, поддерживающим множество оптимизаций и режимов верификации, а язык Promela предоставляет удобные средства описания моделей распределенных систем и, в частности, протоколов когерентности кэш-памяти. Кроме того, Spin может быть использован в качестве основы генераторов тестовых программ, направленных на проверку реализаций протоколов когерентности памяти [18].

Протоколы когерентности могут быть рассмотрены как асинхронные системы из взаимодействующих процессов, причем их математической моделью является система взаимодействующих конечных автоматов. Promela-модель специфицирует поведение асинхронно исполняемых процессов распределенной системы. Каждый процесс представляет расширенный конечный автомат, в котором каждый переход помечен действием [19].

Поведение системы из нескольких процессов представляется расширенным

конечным автоматом, являющимся асинхронной композицией автоматов системы. Для обоснования применения метода на основе синтаксической абстракции необходимо показать, что данный автомат находится в отношении симуляции с автоматом, полученным посредством абстракции.

Язык Promela позволяет описывать модели протоколов когерентности различными способами. В [18] описана модель, полученная выделением групп устройств системы на кристалле в процессы и естественной организацией взаимодействия процессов посредством каналов. Автор также разработал модель протокола когерентности «Эльбрус-4С» в стиле модели из [8] без использования каналов, взамен которых взаимодействие реализовано через разделяемые переменные (массивы). Такая модель менее интуитивна, поскольку требует введения вспомогательных переменных и составления более сложных условий осуществления переходов (например, приема и отправки сообщений). С другой стороны, высокоуровневая организация процессов остается такой же, как и в [18], а избавление от каналов приводит к существенному снижению размеров результирующей системы переходов. Так, объем используемой памяти для модели из трех кэшей уменьшился с сотен до единиц мегабайт. Таким образом, к Promela-моделям могут быть применены различные синтаксические абстракции.

Заключение

Для формальной верификации реализаций протоколов когерентности памяти в масштабируемых системах предлагаются методы проверки моделей и дедуктивной верификации. Проверка моделей характеризуется полной автоматизацией, но подвержена проблеме «взрыва числа состояний», а дедуктивная верификация, допускающая масштабируемость, требует большого объема ручной работы эксперта. Перспективной

представляется комбинация двух этих подходов, призванная сделать метод верификации масштабируемым при приемлемом объеме ручной работы. В литературе приведен ряд примеров такой комбинации. Характерными являются использование устаревших инструментов и отсутствие необходимых компонентов в открытом доступе. На основе работ [13-15] предложены направления создания такой комбинации с использованием современного инструмента Spin.

Литература

1. Manna Z., Pnueli A. The Temporal Logic of Reactive and Concurrent Systems: Specification. – Springer-Verlag, 1992, 427 pp.

2. Clarke E.M., Grumberg O., Peled D. Model Checking. – MIT Press, 1999, 314 pp.

3. Manna Z., Pnueli A. Temporal Verification of Reactive Systems: Safety. – Springer-Verlag, 1995, 512 pp.

4. Park S., Dill D. Verification of FLASH Cache Coherence Protocol by Aggregation of Distributed Transactions. Proceedings of the 8th annual ACM symposium on parallel algorithms and architectures, 1996, pp. 288-296.

5. Clarke E., Talupur M., Veith H. Environment Abstraction for Parameterized Verification. Verification, Model Checking, and Abstract Interpretation, 2006, vol. 3855, pp. 126-141.

6. Clarke E., Long D., McMillan K. Compositional Model Checking. Proceedings of the Fourth IEEE Symposium on Logic in Computer Science, 1989.

7. McMillan K. Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking. Proceedings of the 11th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods, 2001, pp. 179-195.

8. Pnueli A., Ruah. S., Zuck L. Automatic Deductive Verification with Invisible Invariants.

Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, 2001, vol. 2031, pp. 82-97.

9. Arons T., Pnueli A., Ruah. S., Xu Y., Zuck L. Parameterized Verification with Automatically Computed Inductive Assertions. Computer Aided Verification. Lecture Notes in Computer Science, 2001, vol. 2102, pp. 221-234.

10. Lahiri S., Bryant R. Constructing Quantified Invariants via Predicate Abstraction. Verification, Model Checking, and Abstract Interpretation. Lecture Notes in Computer Science, 2004, vol. 2937, pp. 267-281.

11. McMillan K. Quantified Invariant Generation Using an Interpolating Saturation Prover. Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, 2008, vol. 4963, pp. 413-427.

12. Conchon S., Goel A., Krstic S., Mebsout A., Zaidi F. Invariants for Finite Instances and Beyond. Formal Methods in Computer-Aided Design, 2013, pp. 61-68.

13. Chou C., Mannava P., Park S. A Simple Method for Parameterized Verification of Cache Coherence Protocols. Formal Methods in Computer-Aided Design, 2004, vol. 3312, pp. 382-398.

14. Talupur M., Tuttle M. Going with the Flow: Parameterized Verification Using Message Flows. Formal Methods in Computer-Aided Design, 2008, pp. 1-8.

15. O'Leary J., Talupur M., Tuttle M. Protocol Verification Using Flows: An Industrial Experience. Formal Methods in Computer-Aided Design, 2009, pp. 172-179.

16. Zhang M., Bingham J., Erickson J., Sorin D. PVCoherence: Designing Flat Coherence Protocols for Scalable Verification. High Performance Computer Architecture, 2014, pp. 392-403.

17. Krstic S. Parameterized System Verification with Guard Strengthening and Parameter Abstraction. Automated Verification of Infinite State Systems, 2005.

18. Буренков В.С. Генератор тестов для верификации протокола когерентности кэш-памяти. – «Вопросы радиоэлектроники», сер. ЭВТ, 2014, вып. 3, с. 56-63.

19. Holzmann G. Design and Validation of Computer Protocols. – Prentice Hall, 1990, 512 pp.