электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Проблемы формальной верификации технических систем

77-30569/373672
04, апрель 2012

Буренков В. С., Иванов С. Р., Савельев А. Я.

УДК 004.052.4

МГТУ им. Н.Э. Баумана
ivanovsr@bmstu.ru
VanSBuren@mail.ru
a$aveluev@list.ru

Проблема обеспечения правильности программных и аппаратных систем на этапе их проектирования имеет первостепенное значение. Верификация является одним из основных методов ее решения. Верификация – подтверждение на основе представления объективных свидетельств того, что установленные требования были выполнены. Наряду с верификацией рассматривается валидация – подтверждение на основе представления объективных свидетельств того, что требования, предназначенные для конкретного использования или применения, выполнены. Таким образом, верификация – проверка того, что продукт удовлетворяет сформулированным требованиям, а валидация – проверка того, что разработано именно то, что требуется заказчику.

Широко используемыми при валидации методами являются моделирование и тестирование. Моделированию подвергается абстрактная схема или модель системы, а тестирование применяется непосредственно к самому продукту. В случае электронных схем моделированию подвергается проект разрабатываемой схемы (Verilog-описание), тогда как тестированию подвергается сама электронная схема.

Несмотря на то, что моделирование и тестирование зарекомендовали себя как эффективные методы на ранних стадиях отладки, когда разрабатываемая система еще содержит большое количество ошибок, результативность этих методов быстро снижается, по мере исправления найденных ошибок [1].

В области верификации в течение нескольких десятилетий прилагались серьезные усилия по разработке теории, алгоритмов и техники верификации. В настоящее время эти методы разрабатываются в трех основных направлениях:

- дедуктивная верификация;
- проверка эквивалентности;
- model checking (проверка модели).

Дедуктивная верификация обычно подразумевает применение аксиом и правил вывода для доказательства правильности функционирования системы. Эта

весьма сложная процедура не может быть полностью автоматизирована, она требует участия человека, действующего на основе предположений и догадок, использующего интуицию при построении инвариантов и нетривиальном выборе альтернатив. Преимущество дедуктивной верификации состоит в том, что она может быть применена к системам с бесконечным числом состояний.

Проверка эквивалентности связана с разработкой формальных моделей взаимодействующих процессов, выражением в рамках таких моделей как спецификации, так и реализации, и проверкой эквивалентности формально определенных моделей поведения**.**

Метод model checking связан с формальной проверкой выполнения на модели свойств поведения проектируемого объекта, специфицированных на языке формальной логики. Метод model checking может быть полностью автоматизирован. По этой причине он предпочтительнее дедуктивной верификации в тех случаях, когда может быть использован. Однако всегда будут приложения, для полной верификации которых необходимо проводить доказательство теорем. В связи с этим исследования ведутся и в направлении, предусматривающем такую интеграцию дедуктивной верификации и метода model checking, чтобы фрагменты системы, имеющие конечное число состояний, проверялись автоматически.

Главным недостатком верификации является то, что проверяется не реальная система, а ее абстрактная модель. Поскольку такая модель создается человеком, она может быть неадекватной, не сохранять существенных черт исходной системы, и в этом случае проверка спецификации для модели не выявит ошибок, имеющихся в реальной системе. С другой стороны, в модели могут появиться свойства, которых в реальной системе не существует.

Общепризнано, что на основании как моделирования и тестирования, так и верификации по отдельности, нельзя быть полностью уверенными в правильности разрабатываемых систем. Поэтому эти подходы можно считать взаимно дополняющими.

Одной из главных проблем, возникающих при применении метода model checking, является огромное число состояний моделей реальных систем. Верифицируемые системы обычно параллельны, и число состояний моделей таких систем растет экспоненциально с ростом числа компонентов. Этот эффект, названный «взрывом числа состояний», ограничивает применение обычных алгоритмов верификации model checking. Ранние системы верификации, реализующие алгоритмы model checking, могли работать с системами переходов, состоящими из 10^4–10^5 состояний [1].

Одним из подходов к решению проблемы «взрыва числа состояний» является символьная верификация. Символьные, или неявные алгоритмы, используют эффективный метод представления дискретных данных – множеств и отношений – в виде булевых функций в форме бинарных решающих диаграмм (BDDs, Binary Decision Diagrams) – канонических представлений булевых функций, зачастую являющихся существенно более компактными, чем представления стандартными способами, – и позволяют увеличить число состояний верифицируемых систем до астрономических значений. Ранние алгоритмы символьной верификации могли

применяться к системам более чем с 10^{20} состояниями, дальнейшие их модификации позволяют работать с моделями, содержащими более чем 10^{120} состояний [2, 3].

Алгоритмы проверки модели для темпоральной логики CTL (логика ветвящегося времени, Computational Tree Logic) (как явные, так и неявные) для каждой подформулы заданной темпоральной формулы вычисляют множество состояний модели (структуры Крипке), на которых эта подформула выполняется. Эти алгоритмы выполняют преобразования множеств состояний структуры Крипке итеративно до тех пор, пока множество, к которому эти преобразования применяются, не перестает изменяться, то есть вычисляют неподвижные точки преобразования. Можно показать, что для каждой темпоральной формулы CTL (EFφ, AFφ, E(φ_1Uφ_2), A(φ_1Uφ_2), EGφ, AGφ, где A, E – кванторы пути, F, G – темпоральные операторы, φ – формула пути) множество состояний структуры Крипке, на которых выполняется данная формула, является наименьшей или наибольшей неподвижной точкой некоторого монотонного оператора [4]. Теорема Тарского дает алгоритмы вычисления наибольших и наименьших неподвижных точек таких операторов. Эти вычисления могут быть реализованы либо явными алгоритмами, работающими с каждым элементом обрабатываемых множеств, либо неявными (символьными) алгоритмами, работающими с булевыми функциями в форме BDD, представляющими эти множества.

Сложность BDD зависит от порядка переменных, и хотя для большинства булевых функций представление в BDD линейно или полиномиально при любом порядке переменных, для некоторых функций число вершин в представляющих их BDD может существенно различаться при разных порядках переменных. Более того, найдены классы булевых функций, для которых представление в BDD экспоненциально для любых порядков переменных.

К проверке выполнимости формул темпоральной логики LTL (логика линейного времени Linear Time Logic) наиболее практичным является подход, основанный на теории автоматов. Если конечным образом задать язык, содержащий все цепочки, возможные в произвольной структуре Крипке M, а также язык, содержащий все цепочки, не удовлетворяющие заданной формуле φ логики LTL, и построить пересечение этих языков, проверив его на пустоту, то можно доказать, что φ не выполняется на M, и найти контрпример – те вычисления M, которые удовлетворяют формуле $\overline{\varphi}$.

Цепочки, характеризующие вычисления реагирующих систем **(**класса информационных систем, основной функцией которых является поддержание взаимодействия с окружением**)**, являются бесконечными и называются ω-словами. Множества ω-слов называются ω-языками. Некоторые ω-языки можно задать конечной моделью, которая называется автоматом Бюхи. Теоретико-автоматный подход к верификации реагирующих систем состоит в том, что строится автомат Бюхи, задающий все траектории анализируемой системы, другой автомат Бюхи, задающий все неправильные траектории, и анализируется синхронная композиция этих автоматов [4]**.**

Построение автомата Бюхи, допускающего все цепочки, удовлетворяющие заданной LTL-формуле (отрицанию формулы, представляющей проверяемое требование), является одним из самых трудоемких этапов в теоретико-автоматном подходе к верификации. Существует несколько подходов к процедуре такой трансляции, все они концептуально сложны и обычно проводятся в два шага. Первый шаг – построение автомата Бюхи, гарантированно удовлетворяющего поставленным требованиям. На этом шаге обычно получается автомат, число состояний которого экспоненциально относительно длины (числа подформул) рассматриваемой формулы. Для того чтобы сделать эффективными следующие этапы процесса верификации, обычно выполняется второй шаг, который заключается в минимизации полученного автомата. К настоящему времени неизвестны эффективные общие процедуры получения по формуле LTL минимального автомата Бюхи; наилучшие известные методы основаны на использовании эвристик для такой минимизации [4].

Наибольшее распространение среди систем верификации, основанных на теоретико-автоматном подходе, получила система Spin. В системе Spin используется несколько приемов, позволяющих бороться с проблемой «взрыва числа состояний» и увеличить размер тех систем, для которых возможен исчерпывающий анализ.

В Spin не производится полного преобразования системы процессов в структуру Крипке: проверка выполнения свойств системы производится «на лету» (“on-the-fly”), только для части построенной системы переходов, которая постепенно достраивается в ограниченной области памяти.

Spin использует хэш-таблицу для хранения векторов состояний (данных, хранимых для каждого состояния), которые уже были исследованы. На основании информации из вектора состояния хэш-функция вычисляет индекс элемента массива, в котором хранятся указатели на векторы состояний. В случае хэш-конфликтов (ситуаций, когда разные векторы состояний отображаются хэш-функцией в один и тот же слот хэш-таблицы) используются связные списки [5]. Важно, чтобы было как можно меньше хэш-конфликтов, чтобы исключить поиск в списках. Чем меньше размер вектора состояния, тем большее число элементов может поместиться в хэш-таблицу заданного размера. Чем больше число элементов в хэш-таблице, тем менее вероятно возникновение хэш-конфликта. При проведении верификации в Spin можно изменять размер используемой хэш-таблицы, добиваясь повышения эффективности верификации.

Spin позволяет использовать метод, определенным образом кодирующий вектор состояния, и тем самым позволяющий уменьшить его размер.

Векторы состояний могут быть сохранены без использования хэш-таблицы, а с применением представления, схожего с бинарными решающими диаграммами, – минимального автомата. В данном случае необходимый системе объем памяти может быть существенно снижен, однако время проведения верификации может увеличиться в несколько раз.

Одним из самых важных методов оптимизации, реализованных в Spin, является редукция частичных порядков, состоящая в том, что при верификации часть всех возможных путей (всех полных порядков частично упорядоченного

множества операторов параллельной системы) отбрасывается. Например, интерливинг (перекрытие) операторов, работающих с локальными данными в каждом процессе, можно моделировать только одной последовательностью их выполнения: все остальные перестановки выполнения этих операторов будут эквивалентны с точки зрения любого проверяемого свойства. Этот прием позволяет системе Spin существенно снижать сложность проверяемых моделей параллельных систем [4].

Одним из направлений применения метода model checking является стоящая перед азработчиком задача – верификация протоколов когерентности памяти. Методы поиска ошибок в устройствах, реализующих такие протоколы, основанные на симуляции со случайными данными, являются неэффективными. Ошибки могут проявиться лишь при возникновении длинных последовательностей событий, таких как кэш-промахи и прием сообщений различными частями системы. Поскольку число таких последовательностей является комбинаторным, вероятность их возникновения во время моделирования со случайными данными резко уменьшается при увеличении их длины.

В [6] говорится о том, что системы, основанные на неявном представлении состояний модели (то есть использующие BDD, например, NuSMV), менее надежны и эффективны при верификации протоколов когерентности, чем системы с явным представлением (например, Spin). На то приводится ряд причин. Во-первых, скорость и необходимый объем памяти при использовании явного представления не сильно зависит от деталей структур данных, используемых в протоколах, в то время как BDD, например, не очень эффективно представляют FIFO. Во-вторых, верификация с явным представлением состояний может использовать преимущество редукции за счет симметрии, в то время как такая редукция в системах с неявным представлением может только уменьшить количество доказываемых случаев, но не сложность самих доказательств.

В [6, 7] сообщается, что существующие средства формальной верификации, основанные на алгоритмах model checking (Murphi, Spin), применяются для верификации промышленных протоколов когерентности памяти, и хорошо себя зарекомендовали. Однако при этом имеется ограничение на число узлов процессорной системы, отраженных в модели. В [6] этот параметр ограничивается значением 4.

В различных источниках – в частности, [6, 8], – поднимается проблема параметризованной верификации – доказательства корректности протокола при любых значениях числа процессоров в системе (либо каких-то других параметров). Для решения этой проблемы возможно использование некой комбинации методов model checking, композиционной верификации, абстракции и симметрии. Однако, например, при использовании техники абстракции возникает вопрос, всякое ли свойство, выполнимое на абстрактной модели, будет соблюдаться и для реальной системы. Это означает, что возможно появление ложных контрпримеров, не являющихся контрпримерами для реальной системы, и в этом случае нужно решать не только вопрос о том, как проводить абстракцию, но и как бороться с ложными

контпримерами, а также насколько возможно и необходимо автоматизировать эти процедуры.

Рассмотрим основные идеи методов композиционной верификации, абстракции и симметрии.

Многие конечные системы переходов образованы из большого числа процессов, работающих параллельно. Спецификации таких систем часто могут быть разбиты на отдельные свойства, описывающие поведение небольших фрагментов системы. Тогда (в рамках подхода, называемого композиционной верификацией) возможна следующая стратегия: проверить каждое локальное свойство, используя только тот фрагмент системы, который описывается этим свойством. Если мы сможем прийти к заключению, что система удовлетворяет всем локальным требованиям, и если мы знаем, что конъюнкция локальных свойств влечет выполнимость всей спецификации, то мы сможем прийти к выводу о том, что вся система в целом также удовлетворяет этой спецификации.

Одним из основных подходов в рамках метода абстракции является абстракция данных, которая применяется к описаниям системы на высшем ее уровне, еще до того как построена ее модель. Тем самым удается избежать построения нередуцированной модели, которая может оказаться слишком большой, чтобы поместиться в память. Абстракция данных предусматривает поиск отображения реальных значений данных, используемых в системе, в небольшое множество абстрактных значений данных. Распространив это отображение на состояния и переходы, можно построить абстрактную систему, которая симулирует исходную, но обычно имеет гораздо меньший размер. Из-за такого сокращения размера абстрактную систему зачастую удается верифицировать гораздо легче, чем исходную.

В основу методов редукции, использующих симметрию, положена идея о том, что проявление симметрии в параллельных системах с конечным числом состояний влечет за собой существование нетривиальной группы перестановок, которая сохраняет как разметку состояний, так и отношение переходов. Такие группы можно использовать для определения отношений эквивалентности в пространстве состояний систем. Модель, порожденная этим отношением, часто имеет меньший размер, нежели исходная модель. Кроме того, она бисимуляционно эквивалентна исходной модели. Поэтому ее можно использовать для верификации любого свойства исходной модели, выраженного CTL*-формулой [1].

Помимо верификации самих протоколов когерентности памяти, важна верификация средств, реализующих эти протоколы. Для этого необходимы тесты (на языке ассемблера), которые можно получить на основании результатов верификации формальной модели протокола. В [9] приводится пример такой комбинации формальных и неформальных методов: на основании данных, получаемых при формальной верификации, строятся воздействия, являющиеся входными воздействиями для системы, моделирующей соответствующую Verilog-реализацию.

Предполагается применить формальную верификацию с использованием инструмента Spin к протоколу когерентности памяти системы на кристалле «Эльбрус-2S». Стандартная конфигурация системы на кристалле «Эльбрус-2S»

представляет собой четыре четырехъядерных процессорных модуля, связанных друг с другом высокоскоростными каналами. Когерентность доступа нескольких процессоров в общую память поддерживается посредством запросов проверки когерентности, рассылаемых специальным устройством – системным коммутатором – в процессоры системы. Кэш-память второго уровня реализует протокол когерентности MOSI (от названия основных состояний кэш-строки – Modified, Owned, Shared, Invalid). Описание протокола планируется выполнить на языке Promela (входном языке пакета Spin), который предоставляет возможность пользователю строить модели параллельных систем для последующей проверки в их поведении аспектов координации и взаимодействия параллельных процессов. Для этого необходимо абстрагироваться от совокупности множества устройств, реализующих протокол, и представить протокол в виде набора параллельных процессов, решив проблемы их взаимодействия и синхронизации. На основании представлений о том, как должен работать протокол когерентности памяти и что он должен обеспечить, нужно сформулировать требования к протоколу сначала на естественном языке, а затем привести их к формальному виду с помощью формул темпоральной логики. Имея модель и набор требований к ней, можно провести формальную верификацию. На основании результатов верификации предполагается, помимо заключений о корректности протокола (и его модели), получить тесты для Verilog-модели системы (и/или C++-модели), и исследовать их поведение.

Литература

1. E. M. Clarke, O. Grumberg, D. Peled. Model Checking. // MIT Press, 1999 – 314 pp.
2. J.R. Burch, E.M. Clarke, K.L. McMillan. Symbolic Model Checking: 1020 States and Beyond. Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science, June 1990, pp. 428-439.
3. J.R. Burch, E.M. Clarke, D.E. Long, K.C. McMillian, D.L. Dill. Symbolic Model Checking for Sequential Circuit Verification. Technical Report CMU-CS-93-211.Carnegie Mellon University, Pittsburg, PA 15213, July 1993. 50 p.
4. Карпов Ю.Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010 – 560 с.
5. Mordechai Ben-Ari. Principles of the Spin Model Checker. – Springer, 2008 – 232 pp.
6. C. Chou, P. K. Mannava, S. Park. A Simple Method for Parameterized Verification of Cache Coherence Protocols. Proc. Conf. on Formal Methods in Computer-Aided Design (FMCAD04), volume 3312 of Lecture Notes in Computer Science, Springer, 2004. pp. 382–398.
7. J. Harrison. Formal Methods at Intel – An Overview. // Second NASA Formal Methods Symposium, 2010.
8. K. L. McMillan. Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking. Proc. Conf. on Correct Hardware Design and Verication Methods (CHARME ’01), 2001, vol. 2144 of LNCS, pp. 179–195.
9. D. Abts, S. Scott, D. Lilja. So Many States, So Little Time: Verifying Memory Coherence in the Cray X1. Proc. 17th Int'l Symp. Parallel and Distributed Processing, April 2003.

electronic scientific and technical periodical
SCIENCE and EDUCATION
EL № FS 77 - 30569. №0421100025. ISSN 1994-0408

Issues of formal verification of engineering systems

77-30569/373672

04, April 2012

Burenkov V.S., Ivanov S.R., Savel'ev A.Ya.

Bauman Moscow State Technical University
ivanovsr@bmstu.ru
VanSBuren@mail.ru
a$aveluev@list.ru

Improving the verification of hardware and software systems is associated with the usage of formal methods among which the authors consider the model checking method which provides an automated consideration of all possible "paths of functioning" of the system. The problem of “explosion of the number of states” inherent in the method of verification of model checking is analyzed in the article; various methods of dealing with this problem are discussed. Particular attention is paid to verification of coherence memory protocols; possible approaches to solving this problem are outlined.

Publications with keywords: verification, formal methods, validation, coherence protocol memory
Publications with words: verification, formal methods, validation, coherence protocol memory

References

1. Clarke E. M., Grumberg O., Peled D. *Model Checking*. MIT Press, 1999. 314 p.

2. Burch J.R., Clarke E.M., McMillan K.L., Dill D.L., Hwang J. Symbolic Model Checking: 10^{20} States and Beyond. *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, June 1990, pp. 428-439.

3. Burch J.R., Clarke E.M., McMillan K.L., Dill D.L. *Symbolic Model Checking for Sequential Circuit Verification.* Technical Report CMU-CS-93-211.Carnegie Mellon University, Pittsburg, PA 15213, July 1993. 50 p.

4. Karpov Iu.G. *MODEL CHECKING. Verifikatsiia parallel'nykh i raspredelennykh programmnykh system* [MODEL CHECKING.Verification of parallel and distributed software systems]. SPb., BKhV-Peterburg, 2010. 560 p.

5. Mordechai Ben-Ari. *Principles of the Spin Model Checker*. Springer, 2008. 232 p.

6. Chou C., Mannava P. K., Park S. A Simple Method for Parameterized Verification of Cache Coherence Protocols. *Proc. Conf. on Formal Methods in Computer-Aided Design (FMCAD04)*, volume 3312 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 382–398.

7. Harrison J. Formal Methods at Intel – An Overview. *Second NASA Formal Methods Symposium*, 2010.

8. McMillan K. L. Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking. *Proc. Conf. on Correct Hardware Design and Verication Methods (CHARME '01)*, 2001, vol. 2144 of LNCS, pp. 179–195.

9. Abts D., Scott S., Lilja D. So Many States, So Little Time: Verifying Memory Coherence in the Cray X1. *Proc. 17th Int'l Symp. Parallel and Distributed Processing*, April 2003.