ИНФОРМАТИКА

2004 январь-март №1

УДК 519.71

Д.И. Черемисинов

ОБ ИНТЕРПРЕТАЦИИ ВРЕМЕННОЙ ЛОГИКИ ПРИ СИМВОЛИЧЕСКОЙ ВЕРИФИКАЦИИ

В последние годы инструменты для символической верификации на основе временной логики входят в состав большинства индустриальных САПР СБИС. В символических верификаторах используется логика ветвящегося времени CTL. Ее недостатком считается неинтуитивность. Предлагаются интерпретации временных логик, в которых выразительность и интуитивность CTL для интервальных событий не хуже, чем выразительность и интуитивность LTL для мгновенных событий, что позволяет упростить формулирование проверяемых временных свойств.

Введение

Интегральные схемы в настоящее время проектируются в результате выполнения ряда шагов, каждый из которых переводит абстрактную спецификацию, уточняя ее, во все более конкретное воплощение. Процесс проектирования начинается с «поведенческой модели», например с программы, которая описывает архитектуру процессора на уровне набора команд. Результатом проектирования является описание фактической топологии транзисторов и проводов на чипе. Каждый шаг проектирования должен давать описание, релевантное исходному поведению абстрактной модели. К сожалению, методы выполнения проверки соответствия спецификации и реализации становятся все более дорогими и трудными по мере роста сложности проекта. Для многих проектов размер группы проектировщиков, выполняющих проверку (верификацию) проекта, теперь превышает число участников группы проектирования.

Под верификацией понимается проверка корректности и правильности поведения системы. Считается, что система функционирует корректно, если ее поведение отвечает ряду априори сформулированных требований. Под дискретной системой понимается система, функционирование которой в конечном итоге может быть представлено в виде конечноавтоматной модели (finite state machine). Функционирование дискретной системы может быть описано в терминах других моделей, например сетей Петри и их модификаций, формальных грамматик, диаграмм переходов и т.д. Переход от таких моделей к конечноавтоматной модели в ряде случаев может быть легко автоматизирован. Набор свойств, которым должна удовлетворять верифицируемая система, задается как некоторое формальное представление ее поведения.

В моделировании, традиционном способе отладки описания СБИС, подтверждение правильности текущего описания СБИС – это результат прогонов моделирования с большим количеством испытательных ситуаций. Формальная проверка удовлетворения требований к поведению системы (символическая верификация), напротив, использует математические методы, чтобы исследовать полное пространство состояний модели для подтверждения соответствия наперед заданному поведению. Символическая верификация выполняется путем установления соответствия описаний поведения верифицируемой системы заданным требованиям. Чтобы иметь возможность верифицировать описание системы, нужно задать три формализма: язык описания поведения системы, язык описаний требований и отношение соответствия между выражениями этих языков.

За последние годы инструменты для символической верификации все шире применяются в процессе проектирования СБИС. С их помощью были обнаружены в дорогостоящих проектах тонкие дефекты, которые возникают при чрезвычайно невероятных условиях. До недавнего времени эти инструменты рассматривались как программы, имеющие только академический интерес, теперь же они рутинно используются в индустриальных САПР [1, 2]. В этих программах описание поведения и требования задаются как логические формулы, а отношение соответствия обозначает логическую импликацию. Формально символический верификатор доказывает истинность логической формулы импликации.

Ключевой вопрос при использовании инструмента для символической верификации – это формулировка временных условий на *языке спецификации временных условий*, который является одним из определяющих интерфейсов символического верификатора. (Другой важный интерфейс – это язык моделирования, в этом качестве обычно используется язык описания аппаратных средств ЭВМ, применяемый проектировщиками СБИС). Языки спецификации временных условий имеют в основе формализм временных логик [4]. В статье предлагаются интерпретации временных логик, которые позволяют упростить формулирование проверяемых временных свойств.

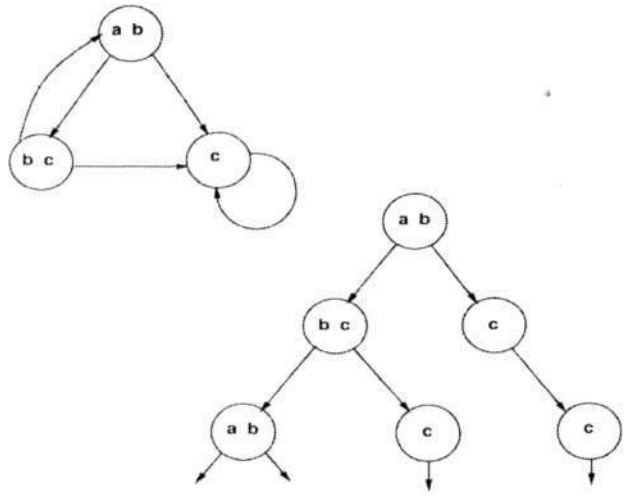
Логики для описания временных свойств в параллельных системах

Временная логика представляет собой модальную логику, интерпретированную как описание временной упорядоченности событий. Временные логики оказываются полезными при описании параллельных систем, потому что они могут описывать порядок событий во времени. Они были первоначально предложены математиками как способ исследований высказываний естественного языка, касающихся времени [3]. Предлагалось множество различных временных логик, и эти формализмы часто классифицируются согласно предположениям о типе структуры (линейной или ветвящейся), на которой интерпретируются выражения формализма. Эта классификация может иногда вводить в заблуждение, так как семантика временной логики может быть задана на универсальной структуре, которая по историческим причинам называется моделями Kripke [3].

Формулы линейной временной логики LTL составлены из набора атомарных суждений, использующих обычные связки булевой логики и временные связки: G («всегда»), F («в будущем»), X («затем») и U («пока»). Формулы временной логики ветвящегося времени CTL* дополнительно, по сравнению с LTL, расширяются кванторами E («существует вычисление») и A («для всех вычислений»). Логика ветвящегося времени CTL – это фрагмент CTL*, в котором каждый временный связке предшествует квантор. Наконец, логика ветвящегося времени ∀CTL - фрагмент CTL, в формулах которой допускается только квантор всеобщности. В LTL неявно предполагается вхождение квантора всеобщности в начале каждой формулы.

Интерпретация формул временной логики на модели Kripke

Модель Kripke представляет собой помеченный граф переходов [3]. Если задано некоторое начальное состояние, то модели Kripke можно сопоставить бесконечное дерево так, как показано на рис. 1. Пути в дереве можно интерпретировать как возможные вычисления некоторой программы, поэтому бесконечное дерево, полученное из модели Kripke, задает дерево вычислений программы. Временные логики отличаются интерпретацией ветвлений в дереве вычислений.



Граф переходов или модель Kripke

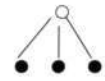
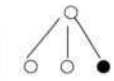
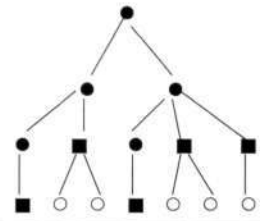
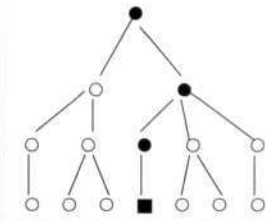
Раскрутка модели в бесконечное дерево

Рис. 1. Модель вычислений для временной логики

Формулы в линейной временной логике интерпретируются как описания событий по отдельному пути вычисления. В логике ветвящегося времени временные формулы интерпретируются как пути (цепочки событий), достижимые из данного состояния. Ниже приведена интерпретация формул логики ветвящегося времени CTL* [4-6] на этой модели (табл. 1).

Таблица 1

Интерпретация формул логики ветвящегося времени

Тип формулы	Графическое представление	Описание	Графическое представление	Описание
Атомарная	●	Формула f истинна в текущем состоянии	■	Формула g истинна в текущем состоянии
Пропозициональная	○	$!f$	□	$!g$
Пропозициональная		$F \wedge g$		
Следующее состояние		AX f		EX f
После		F AU g		F EU g

Имеются два типа формул в CTL*: формулы состояний (которые истинны в определенном состоянии) и формулы пути (которые истинны для определенного пути). Пусть AP – это набор имен атомарных суждений. Синтаксис формул состояний задается следующими правилами:

- если p ∈ AP, то p – формула состояния;
- если f и g – формулы состояния, то $\neg f$ и $f \vee$ g – формулы состояния;
- если f – формула пути, то E (f) – формула состояния.

Следующие два дополнительных правила определяют синтаксис формул пути:

- если f – это формула состояния, то f также формула пути;
- если f и g – формулы пути, то $f, f \vee g$, Xf и f U g – тоже формулы пути.

CTL* – это множество формул состояний, образованных по вышеупомянутым правилам. Семантика CTL* относительно модели Kripke M = (S, R, L) определяется следующим образом. S – это произвольное множество, элементы которого называются состояниями; R ⊆ S × S – функция переходов, которая должна быть полна (т.е. для всех состояний s ∈ S существует состояние s' ∈ S такое, что (s, s') ∈ R), и L: S → AP является функцией, которая маркирует каждое состояние набором атомных суждений, истинных в этом состоянии. Предполагается, что S конечно, т.е. рассматриваются только конечные структуры Kripke.

Путь в M – это бесконечная последовательность состояний π = s_0, s_1,..., таких, что для каждого $i > 0$ $(s_i, s_{i+1}) \in R$. π^i обозначает суффикс π, начинающийся с s_i. Если f – формула состояний, то выражение $M, s \models f$ означает, что f истинна в состоянии s для модели M. Точно так же, если f – это формула пути, то $M, \pi \models f$ означает, что f истинна для пути π в модели M. Когда ясно из контекста, что модель задана, ее упоминание в формуле обычно опускается. Отношение ⊨ определено индуктивно следующим образом (предполагается, что f_1 и f_2 являются формулами состояний, а g_1 и g_2 - это формулы пути):

1) $s \models p \quad \Leftrightarrow \quad p \in L(s)$;
2) $s \models \neg f_1 \quad \Leftrightarrow$ не имеет места $s \models f_1$;
3) $s \models f_1 \vee f_2 \quad \Leftrightarrow \quad s \models f_1$ или $s \models f_2$;
4) $s \models E(g_1) \quad \Leftrightarrow$ существует путь π, начинающий с s, такой, что $\pi \models g_1$;
5) $\pi \models f_1 \quad \Leftrightarrow$ s - первое состояние π и $s \models f_1$;
6) $\pi \models \neg g_1 \quad \Leftrightarrow$ не имеет места $\pi \models g_1$;
7) $\pi \models g_1 \vee g_2 \quad \Leftrightarrow \quad \pi \models g_1$ или $\pi \models g_2$;
8) $\pi \models X\, g_1 \quad \Leftrightarrow \quad \pi^1 \models g_1$;
9) $\pi \models g_1\, U\, g_2 \quad \Leftrightarrow$ существует $k > 0$, такое, что $\pi^k \models g_2$ и для всех $0 \le k < j$, $\pi^j \models g_1$.

Остальные операторы CTL* определены как следующие сокращения:

- $f_1 \wedge f_2 \equiv \neg(\neg f_1 \vee \neg f_2)$, $\quad F\, f \equiv \text{true}\, U\, f$;
- $A(f) \equiv \neg E(\neg f)$, $\quad G\, f \equiv \neg F\, \neg f$.

CTL [5] – подмножество CTL*, в которой синтаксис формул путей задается следующими двумя правилами:

- если f и g – формулы состояний, то Xf и fUg - формулы пути;
- если f – формула пути, $\neg f$ – тоже формула пути.

Интерпретация наиболее употребительных операторов CTL проиллюстрирована в табл. 1.

Линейная временная логика (LTL) состоит из формул, которые имеют форму Af, где f – формула пути, в которой подформулы являются только атомарными суждениями. В LTL синтаксис формул путей задается следующими правилами:

- если $p \in AP$, то p – формула пути;
- если f и g – формулы пути, то $\neg f$, $f \vee g$, Xf и fUg – тоже формулы пути.

Выразительные возможности LTL и CTL несравнимы [7], в то время как CTL* более выразительна по сравнению как с LTL, так и с CTL [4].

Интерпретации временных логик на модели Kripke позволяют установить много важных свойств этих формализмов [14]. Однако эта модель не отражает важные характеристики временного поведения моделей, используемые при проектировании дискретных устройств. Предположение о временном упорядочении событий присуще любой кибернетической системе. И модель Kripke отражает только ту характеристику поведения систем, которая связана с упорядочением событий. Понятие события в этой модели представлено состоянием. О временных аспектах самих событий никаких предположений не делается.

В нормативной семантике таких языков, как Verilog и VHDL, используется понятие событий, которые не имеют длительности. События в этих языках представляют собой изменения значений в проводах и регистрах, т.е. фронты сигналов. В то же время в теории автоматов события – это состояния автомата, т.е. интервалы времени, когда сигналы сохраняют некоторые значения неизменными.

Временные логики являются именно логиками и, значит, предполагают наличие множества интерпретаций. Очевидно, выражения временных логик могут иметь другие интерпретации, учитывающие временные аспекты поведения, связанные с длительностью событий. Ниже описываются интерпретации выражений временной логики, учитывающие длительность событий. Разные предположения о длительности событий приводят к разным логикам. Таким образом, предположения о длительности событий позволяют различать логики не только формально (по правилам конструирования выражений), но и на содержательном уровне.

Интуитивная интерпретация формул временной логики

Объектами временной логики являются события. События можно представлять себе или отметками на оси времени или временными интервалами конечной, но не нулевой длины.

Пусть все события моментальны, т.е. не имеют длительности и порождаются некоторым процессом. Предположим:

а) следующее событие уникально;

б) события симметричны относительно времени, т.е. предполагается, что настоящий и все будущие моменты времени принципиально одинаковы.

Пусть $\pi = s_0, s_1, \ldots$ – некоторая история функционирования процесса. Утверждение p истинно для всей истории π ($\pi \vdash p$), если и только если p истинно в состоянии s_0. Обозначим утверждение, что p истинно в следующем состоянии, через $\circ p$, а утверждение, что p истинно для всех состояний, – через $\Box p$.

Утверждение $\neg p$ истинно для истории π, т.е. $\pi \vdash \neg p$, если не имеет места $\pi \vdash p$, и

$\pi \vdash p \vee q$, если и только если $\pi \vdash p$ и $\pi \vdash q$;

$\pi \vdash \circ p$, если и только если $\pi^1 \vdash p$;

$\pi \vdash \Box p$, если и только если для каждого $i \geq 0$ $s_i, s_{i+1}, s_{i+2} \ldots \vdash p$.

Полученное логическое исчисление может быть дополнено временным оператором «иногда» – $\Diamond$, который определяется как сокращение $\Diamond p \equiv \neg\Box\neg p$, и остальными логическими связками, которые вводятся обычным способом как сокращения выражений через имеющиеся связки. При таком определении оператор «иногда» обозначает «не никогда». С учетом дополнений введенные обозначения образуют формализм LTL. Линейность в названии логики объясняется тем, что в число ее аксиом входит предположение о существовании для каждого события только одного следующего события.

Известна формализация понятия процесса и на основе другой системы аксиом, использующей в качестве базы представление об операции. Ограничившись рассмотрением только одного свойства операции – протяженности во времени, будем считать, что операция – это интервальное событие, т.е. временной интервал конечной, но не нулевой длины.

Пусть T – множество операций, которые могут выполняться в рассматриваемом объекте. Обозначим R_{ab} утверждение, что операция a предшествует операции b при функционировании объекта. С помощью предиката R выразим следующие временные отношения между операциями $a, b \in T$:

$A_{ab} = R_{ab} \wedge \neg\exists c(R_{ac} \wedge R_{cb})$ – граничит;

$O_{ab} = (R_{ab} \wedge R_{ba})$ – перекрывается;

$Cab = \forall c(O_{ca} \wedge O_{cb})$ – содержится.

Основные свойства предиката R задаются аксиомами:

$\neg R_{aa}$ – иррефлексивность;

$R_{ab} \wedge R_{bc} \supset R_{ac}$ – транзитивность;

$R_{ab} \supset (A_{ab} \vee \exists c(Aac \wedge A_{cb}))$ – связность.

Пусть p – некоторое утверждение, касающееся операций из T. Обозначим $a \vdash p$ утверждение, что p истинно для операции $a \in T$. Введем модальности

$\Diamond p = \forall a,b((R_{ab} \vee a \vdash p) \supset b \vdash p)$ – «иногда»;

$\Box p = \forall a\ \exists b((R_{ab} \vee a \vdash p) \wedge b \vdash p)$ – «всегда»,

которые позволяют при описании процессов не употреблять в явном виде предикат R.

Построенное исчисление – это CTL. Система аксиом этого исчисления базируется на предположении о связности операций в процессах. Связность операций означает, что за окончанием выполнения одной операции без временного интервала всегда следует начало выполнения другой.

В интуитивной интерпретации несравнимость LTL и CTL очевидна, эти формализмы описывают системы принципиально разных событий. Системы аксиом линейного и ветвящегося времени в интуитивной интерпретации не сводимы друг к другу. Следствием несводимости систем аксиом является независимость предположений о связности операций и уникальности следующего момента времени. Кроме того, эти интерпретации непосредственно используют предположения о свойствах событий, принятые в распространенных формализмах для описания поведения, таких как Verilog и VHDL или дискретный автомат. Общность свойств событий дает возможность на содержательном уровне интерпретировать выражения временной логики и таким образом упростить формулировку временных свойств для символической верификации.

Заключение

Дискуссия относительно сравнительных достоинств логик линейного и ветвящегося времени началась еще в 1980 г. [7-15]. В работе [14] приведено господствующее мнение относительно сравнительных достоинств логик линейного и ветвящегося времени, состоящее в следующем. Эти формализмы нельзя предпочесть на основе аргументов сложности доказательства, так как ни один из подходов не доминирует над другим с точки зрения «вычислительной сложности в самом плохом случае». В то же время эти формализмы нельзя предпочесть на основе аргументов выразительности, так как известно, что LTL и CTL выразительно не сравнимы [7]. Таким образом, кажется, нельзя сделать выбор формализма на основе исключительно теоретических аргументов. Учитывая возрастающее использование верификаторов в промышленности [16, 17], выбор будет сделан рынком, а не сообществом исследователей.

Учитывая предлагаемые интерпретации, можно утверждать, что замена линейной логики ветвящейся или наоборот соответствует кодированию событий одной природы другими событиями. В большинстве случаев это ухудшает понимание временных формул, кроме того, это кодирование не может быть выполнено формально.

Несмотря на феноменальный успех верификаторов на основе CTL, считается [14], что как язык спецификаций временных свойств CTL не интуитивен, поэтому его трудно использовать [13]; язык не приспособлен для работы с иерархически устроенными моделями, поэтому он трудно совместим с гибридными технологиями верификации [15]. Коммуникативные свойства логики линейного времени объясняются ее интуитивностью для модели Kripke. Поэтому считается [14], что логика линейного времени выразительна и интуитивна, поддерживает структурные рассуждения и технологию гибридной верификации и применима в верификаторах, использующих комбинацию перечислительного и редукционного метода обработки пространства состояний.

Если использовать интуитивную интерпретацию CTL, то ее выразительность и интуитивность для интервальных событий не хуже, чем выразительность и интуитивность LTL для мгновенных событий.

Список литературы

1. Methodology and system for practical formal verification of reactive hardware / I. Beer, S. Ben-David, D. Geist et al. // Proc. 6th Conference on Computer Aided Verification. - Vol. 818 of Lecture Notes in Computer Science. – Stanford, 1994. – P. 182–193.

2. Goering R. Model checking expands verification's scope // Electronic Engineering Today, February 1997.

3. Hughes G.E., Creswell M.J. Introduction to Modal Logic. - London, Methuen, 1977.

4. Lamport L. Sometimes is sometimes «not never» - on the temporal logic of programs // Proc. 7th ACM Symp. on Principles of Programming Languages, January 1980. - P. 174–185.

5. Clarke E.M., Emerson E.A. Synthesis of synchronization skeletons for branching time temporal logic // Logic of Programs: Workshop, Yorktown Heights, NY, May 1981. – Vol. 131 of Lecture Notes in Computer Science. - Springer-Verlag, 1981.

6. Clarke E.M., Emerson E.A., Sistla A.P. Automatic verification of finite-state concurrent systems using temporal logic specifications // ACM Transactions on Programming Languages and Systems. - N 2. – Vol. 8. – 1986. – P. 244-263.

7. Emerson E.A., Halpern J. Y. «Sometimes» and «not never» revisited: on branching versus linear time temporal logic // Journal of the ACM. - Vol. 33. - N 1. – 1986. – P. 151 - 178.

8. Pnueli A. The temporal logic of programs // Proc. 18th IEEE Symp. on Foundation of Computer Science, 1977. – P. 46–57.

9. Emerson E.A., Clarke E.M. Characterizing correctness properties of parallel programs using fixpoints // Proc. 7th Int'l Colloq. on Automata, Languages and Programming. – 1980. – P. 169–181.

10. Ben-Ari M., Pnueli A., Manna Z. The temporal logic of branching time // Acta Informatica. – Vol. 20. – 1983. – P. 207–226.

11. Pnueli A. Linear and branching structures in the semantics and logics of reactive systems // Proc. 12th Int. Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science. – Springer-Verlag, 1985. – P. 15–32.

12. Emerson E.A., Lei C.-L. Modalities for model checking: Branching time logic strikes back // Proc. 20th ACM Symp. on Principles of Programming Languages, New Orleans, January 1985. – P. 84–96.

13. Clarke E.M., Draghicescu I.A. Expressibility results for linear-time and branching-time logics // Proc. Workshop on Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency. – Vol. 354 of Lecture Notes in Computer Science. - Springer-Verlag, 1988. – P. 428–437.

14. Vardi M.Y. Linear vs. branching time: A complexity-theoretic perspective // Proc. 13th IEEE Sym. on Logic in Computer Science. – 1998. – P. 394–405.

15. Vardi M.Y. Sometimes and not never re-revisited: on branching vs. linear time // Proc. 9th Int'l Conf. on Concurrency Theory, Lecture Notes in Computer Science 1466. – 1998. – P. 1–17.

16. A Hybrid Verification Approach: Getting Deep into the Design / S. Hazelhurst, O. Weissberg et al. // Design Automation Conference DAC 2002, June 1-14, 2002. - New Orleans, Louisiana, USA. – P. 111-116.

17. Kurshan R.P. Formal verification in a commercial setting // Design Automation Conference. - Anaheim, CA, USA. - 1997. – P. 258-262.

Поступила 16.01.04

Объединенный институт проблем информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail:cher@newman.bas-net.by

D.I. Cheremisinov

TEMPORAL LOGIC INTERPRETATION FOR SYMBOLIC MODEL CHECKING

Verification is understood as check of a correctness of behavior of concurrent system. It is supposed that the system behaves correctly if its behavior satisfies a number of a priori formulated requirements. Temporal logics have proved to be useful for specifying concurrent systems, because they can describe the ordering of events in time without introducing time explicitly. For last years tools for symbolical verification on the basis of temporal logic are routinely used in industrial applications. Temporal-logic model checkers use a CTL temporal logic. The weakness of CTL is nonintuitivity. Interpretations of the temporal logic in which expressiveness and intuitivity CTL for interval events is not worse, than expressiveness and intuitivity LTL for momentary events are offered. Thouse interpretations allow to simplify a formulation of checked temporal properties.