УДК 004.052.42

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

V. Burenkov

О КОНСЕРВАТИВНОМ ПРЕОБРАЗОВАНИИ ФОРМАЛЬНЫХ МОДЕЛЕЙ, ИСПОЛЬЗУЕМЫХ ПРИМЕНИТЕЛЬНО К МАСШТАБИРУЕМЫМ СИСТЕМАМ ДЛЯ ВЕРИФИКАЦИИ ПРОТОКОЛОВ КОГЕРЕНТНОСТИ ПАМЯТИ

ON CONSERVATIVE TRANSFORMATIONS OF FORMAL MODELS OF CACHE COHERENCE PROTOCOLS USED IN SCALABLE SYSTEMS

Приведена математическая модель рассматриваемых протоколов. Предложен метод построения консервативных преобразований модели, используемый при верификации протокола сложной системы на кристалле с архитектурой «Эльбрус».

This article formulates a mathematical model of cache coherence protocols of scalable systems. The paper proposes a method for synthesizing conservative transformations of the model used in verification of cache coherence protocol of a complex «Elbrus» system-on-chip.

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

Keywords: formal method, model checking, model transformations, cache coherence protocol.

Введение

В статье предлагается развитие приведенных в [1] направлений по разработке формальных методов верификации, способных адаптироваться к увеличению числа ядер в современных микропроцессорах.

Описываются консервативные, сохраняющие свойства-инварианты (например, невозможность нахождения строки в модифицированном состоянии в двух кэшах) преобразования моделей протоколов когерентности, позволяющие существенно уменьшить размер

модели.

1. Выбор математической модели для представления протоколов когерентности

1.1. Модель протокола когерентности

Возможность использования языка Promela для описания формальных моделей протоколов когерентности рассматривается в [2]. Математическая модель протоколов когерентности, используемая в данной работе, основана на формальной семантике языка Promela и базируется на модели, описанной в [3].

На первом, нижнем уровне модель представлена системой переходов – стандартной моделью аппаратных и программных систем. Второй уровень представлен канальной системой, описывающей асинхронную композицию графов программы, получаемых на основе операторов языка Promela. Система переходов получается путем «развертывания» канальной системы.

1.2. Система переходов

Система переходов TS задается шестеркой $TS = (S, Act, \rightarrow, I, AP, L)$, где S – множество состояний, Act – множество действий, $\rightarrow \subseteq S \times Act \times S$ – отношение переходов, $I \subseteq S$ – множество начальных состояний, AP – множество атомарных высказываний, $L : S \rightarrow 2^{AP}$ – функция пометок. Для краткости под обозначением $s \overset{\alpha}{\rightarrow} s'$ понимается $(s, \alpha, s') \in \rightarrow$. Если осуществляемое действие в рассматриваемом контексте неважно, записывается $s \rightarrow s'$.

1.3. Граф программы

Promela-модель состоит из конечного числа параллельных процессов, описываемых с помощью операторов этого языка. Множество всех переменных модели обозначим через Var, множество каналов (FIFO-очередей) модели – через $Chan$.

Переменные и каналы в Promela типизированы [4]. Обозначим через $dom(x)$ тип переменной x. Аналогичное обозначение применимо и для каналов $c \in Chan$: под типом

канала $dom(c)$ понимается тип переменных, которые могут быть переданы через канал. Помимо типа каждый канал имеет конечную емкость $cap(c)$, т.е. максимальное количество сообщений, которые он способен хранить.

Интерпретацией η переменных называется отображение, ставящее в соответствие каждой переменной $v \in Var$ значение $\eta(v) \in dom(v)$. Через $\eta[v := r]$ будем обозначать интерпретацию переменных, присваивающую значение r переменной v и оставляющую все остальные переменные неизменными. Через $Eval(Var)$ обозначим множество интерпретаций переменных.

Интерпретацией ξ каналов называется отображение, ставящее в соответствие каждому каналу $c \in Chan$ последовательность $\xi(c) \in dom(c)^*$ такую, что $len(\xi(c)) \le cap(c)$, где $len(\cdot)$ – длина последовательности, * – звезда Клини.

Запись интерпретации вида $\xi(c) = v_1 v_2 \ldots v_k$, где $cap(c) \ge k$, показывает, что элемент v_1 находится в голове очереди c, v_2 – следующий за ним элемент и так далее, элемент v_k находится в хвосте очереди. Обозначим через $\xi[c := v_1 \ldots v_k]$ интерпретацию каналов, присваивающую последовательность $v_1 \ldots v_k$ каналу c и оставляющую все остальные каналы неизменными. Интерпретация ξ_0 отображает канал в пустую последовательность. Множество всех интерпретаций каналов обознается через $Eval(Chan)$.

Обозначим через $Cond(Var, Chan)$ множество логических выражений относительно переменных $v \in Var$ и каналов $c \in Chan$. Выражения относительно каналов строятся из вызовов функций empty, nempty, full, nfull, len [4] и их объединения с помощью операторов языка Promela.

Формальная семантика оператора с переменными из множества Var и каналами из множества $Chan$ представляется графом программы над $(Var, Chan)$ – орграфом, ребра которого помечены условиями над элементами $(v, c) \in Var \times Chan$ и действиями. Вершины

графа программы образуют управляющую функцию.

Множество действий будем рассматривать как объединение $Act \cup Comm$, элементы которого определяются только шестью базовыми операторами языка Promela: присваиванием, оператором assert, оператором print, выражением, отправкой сообщения в канал, извлечением сообщения из канала. Действия, выраженные первыми четырьмя операторами, составляют множество Act, а последними двумя операторами – множество $Comm$.

Графом программы (program graph) PG над $(Var, Chan)$ называется шестерка

$$PG = (Loc, Act, Effect, \Rightarrow, Loc_0, g_0),$$

где Loc – множество состояний (вершин) графа, Act – множество действий, $Effect : Act \times Eval(Var) \to Eval(Var)$ – функция, определяющая результат действий, $\Rightarrow \in Loc \times Cond(Var, Chan) \times (Act \cup Comm) \times Loc$ – отношение переходов, $Loc_0 \subseteq Loc$ – множество начальных состояний, $g_0 \in Cond(Var, Chan)$ – начальное условие.

Переход $(l, g, act, l') \in \Rightarrow$ будем сокращенно обозначать $l \overset{g:act}{\Rightarrow} l'$. Логическое условие g называется *защитой* перехода $l \overset{g:act}{\Rightarrow} l'$. Действие act может быть осуществлено только в том случае, если защита g истинна, а действие выполнимо.

1.4. Канальная система

Канальная система CS над $(Var, Chan)$ состоит из графов программы PG_i над $(Var_i, Chan)$, где $1 \le i \le n$ и $Var = \bigcup_{1 \le i \le n} Var_i$. Соответственно, для CS приято обозначение: $CS = [\, PG_1 | \ldots | PG_n \,]$.

В языке Promela возможны два типа взаимодействия между процессами посредством каналов: синхронная передача сообщения через канал нулевой емкости и асинхронная передача сообщения через канал ненулевой емкости. Поскольку при разработке моделей протоколов когерентности не обнаружена необходимость в использовании синхронной пе-

редачи сообщений, в дальнейшем такой тип взаимодействия рассматриваться не будет.

Семантика канальной системы формализуется посредством системы переходов. Пусть $CS = \left[PG_1 | \ldots | PG_n \right]$ – канальная система над $(Var, Chan)$. Состояния соответствующей системы переходов $TS(CS)$ являются кортежами вида $\langle l_1, \ldots, l_n, \eta, \xi \rangle$, где l_i – состояние графа PG_i, $1 \le i \le n$, $\eta \in Eval(Var)$ – текущая интерпретация переменных, $\xi \in Eval(Chan)$ – интерпретация каналов.

Пусть $CS = \left[PG_1 | \ldots | PG_n \right]$ – канальная система над $(Var, Chan)$, и $PG_i = (Loc_i, Act_i, Effect_i, \Rightarrow_i, Loc_{0,i}, g_{0,i})$, $1 \le i \le n$.

Системой переходов $TS(CS)$, соответствующей данной канальной системе, называется шестерка:

$$TS(CS) = (S, Act, \to, I, AP, L),$$

где $S = (Loc_1 \times \ldots \times Loc_n) \times Eval(Var) \times Eval(Chan)$, $Act = \bigcup_{0<i\le n} Act_i \cup \{\tau\}$, где τ – специальный символ, представляющий все коммуникационные действия, при которых происходит обмен данными, $\to$ определяется приведенными ниже правилами, $I = \{l_1, \ldots, l_n, \eta, \xi_0 \mid \forall 0 < i \le n : (l_i \in Loc_{0,i} \wedge (\eta, \xi_0) \models g_{0,i})\}$, $AP = \bigcup_{0<i\le n} Loc_i \cup Cond(Var, Chan)$, $L(\langle l_1, \ldots, l_n, \eta, \xi \rangle) = \{l_1, \ldots, l_n\} \cup \{g \in Cond(Var, Chan) \mid (\eta, \xi) \models g\}$.

Правила, определяющие отношение переходов $\to$ системы $TS(CS)$:

1. Интерливинг для $\alpha \in Act_i$:

$$\frac{l_i \overset{g:\alpha}{\Rightarrow} l_i' \ \wedge \ (\eta, \xi) \models g}{\langle l_1, \ldots, l_i, \ldots, l_n, \eta, \xi \rangle \overset{\alpha}{\to} \langle l_1, \ldots, l_i', \ldots, l_n, \eta', \xi \rangle}, \quad \text{где } \eta' = Effect(\alpha, \eta).$$

2. Асинхронная передача сообщения для $c \in Chan$, $cap(c) > 0$:

а) получение значения по каналу c и присваивание его переменной x:

$$\frac{l_i \overset{g:c?x}{\Rightarrow} l_i' \ \wedge \ (\eta,\xi) \models g \ \wedge \ len(\xi(c)) = k > 0 \ \wedge \ \xi(c) = v_1 \ldots v_k}{\langle l_1,\ldots,l_i,\ldots,l_n,\eta,\xi\rangle \overset{\tau}{\rightarrow} \langle l_1,\ldots,l_i',\ldots,l_n,\eta',\xi'\rangle},$$

где $\eta' = \eta[x := v_1]$ и $\xi' = \xi[c := v_2 \ldots v_k]$;

б) отправка значения $v \in dom(c)$ по каналу c:

$$\frac{l_i \overset{g:c!v}{\Rightarrow} l_i' \ \wedge \ (\eta,\xi) \models g \ \wedge \ len(\xi(c)) = k < cap(c) \ \wedge \ \xi(c) = v_1 \ldots v_k}{\langle l_1,\ldots,l_i,\ldots,l_n,\eta,\xi\rangle \overset{\tau}{\rightarrow} \langle l_1,\ldots,l_i',\ldots,l_n,\eta,\xi'\rangle},$$

где $\xi' = \xi[c := v_1 v_2 \ldots v_k v]$.

2. Синтез совокупности преобразований, приводящих к получению абстрактной модели

2.1. Абстрактная модель протоколов когерентности

В данной работе рассматриваются протоколы когерентности, в которых исполнение запросов происходит под управлением координирующего устройства. В микропроцессорах с архитектурой «Эльбрус» таким устройством является системный коммутатор home-процессора, к памяти которого происходит обращение.

В качестве математической модели протокола когерентности будем использовать канальную систему $CS = [PG_0 \,|PG_1|\ldots|PG_n]$, где PG_0 – граф программы, соответствующий системному коммутатору home-процессора, $PG_1,\ldots,PG_n$ – идентичные графы программы, соответствующие контроллерам кэш-строки, находящимся в кэшах верифицируемой системы.

Каждый элемент массивов рассматривается как отдельный элемент соответствующего множества Var_i или $Chan_i$, в зависимости от типа массива.

Проверяемые свойства протокола затрагивают не более двух кэшей. Поскольку все кэш-контроллеры идентичны и взаимозаменяемы, выбор двух конкретных индексов графов $PG_1,\ldots,PG_n$ не имеет значения. Поэтому без потери общности можно считать, что рассматриваются графы PG_1 и PG_2, и все проверяемые свойства сформулированы только

относительно части системы, определяемой этими графами.

Цель предлагаемого метода заключается в том, чтобы найти такую канальную систему CS_{abs}, где система переходов $TS(CS_{abs})$, описывающая ее семантику, симулирует исходную систему переходов и при этом имеет меньшее число состояний.

Соответственно [3], симуляцией для пары систем переходов (TS_1, TS_2), где $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, $i = 1,2$, называется отношение $\mathsf{R} \subseteq S_1 \times S_2$ такое, что:

$$\text{(A) } \forall s_1 \in I_1 : (\exists s_2 \in I_2 : (s_1, s_2) \in \mathsf{R}).$$

(B) $\forall (s_1, s_2) \in \mathsf{R}: \; L_1(s_1) = L_2(s_2)$ и, если $\exists s_1' : (s_1 \rightarrow_1 s_1')$, то $\exists s_2' : (s_2 \rightarrow_2 s_2')$ такое, что $(s_1', s_2') \in \mathsf{R}$.

Доказано, что отношение симуляции сохраняет свойства-инварианты [3, 5].

Первым этапом предлагаемого метода является замена исходной модели *CS* на абстрактную модель $CS_{abs} = [PG_0 \,|\, PG_1 \,|\, PG_2 \,|\, PG_3]$, где граф PG_3 соответствует абстрактному процессу, представляющему кэш-контроллеры исходной системы с номерами 3,…,*n*. При этом, если $dom(v) = \{1,...,n\}$, $v \in Var_i$ или $v \in Chan_i$, $i = 0,...,n$, то осуществляется замена $dom(v)$ на $dom_{abs}(v) = \{1,2,ABS\}$, где $ABS > 2$ – некоторая константа. Работа на уровне канальной системы и графов программ предполагает проведение процесса абстракции на синтаксическом уровне.

Путь получения абстрактной канальной системы основан на следующем положении: абстрактная система переходов допускает все возможные поведения процессов, соответствующих системному коммутатору home-процессора и двум кэш-контроллерам, включая поведения, вызванные воздействиями процессов, представляющих остальные кэш-контроллеры, на выделенные три процесса.

Пометим состояния системы переходов $TS(CS)$, заметив, что поскольку работа процессов PG_1 и PG_2 (состояние которых нас интересует) координируется процессом PG_0, то состояние последнего также нужно принимать во внимание. Определим множество

атомарных высказываний $AP = \bigcup_{i=1,2} Loc_i \cup \bigcup_{i=0,1,2} Cond(Var_i, Chan_i)$ и функцию пометок состояний $L(\langle l_0,\ldots,l_n,\eta,\xi\rangle) = \{l_1, l_2\} \cup \{g \in \bigcup_{i=0,1,2} Cond(Var_i, Chan_i) \mid (\eta,\xi) \models g\}$.

Определим абстрактные преобразования с помощью функции $f : S \to S_{abs}$, ставящей в соответствие каждому состоянию исходной модели $\langle l_0,\ldots,l_n,\eta,\xi\rangle \in S$ состояние абстрактной модели $\langle l_{0abs}, l_{1abs}, l_{2abs}, l_{3abs}, \eta_{abs}, \xi_{abs}\rangle \in S_{abs}$.

При этом $l_{iabs} = l_i$, $i = 0,1,2$ и, как показано в разделе 3, каждому переходу в исходной модели $\langle\ldots,l_i,\ldots,l_n,\eta,\xi\rangle \to \langle\ldots,l_i',\ldots,l_n,\eta',\xi'\rangle$, $i > 2$ соответствует переход в абстрактной модели $\langle\ldots,l_{3abs},\eta_{abs},\xi_{abs}\rangle \to \langle\ldots,l_{3abs}',\eta_{abs}',\xi_{abs}'\rangle$.

Опишем действие этой функции с помощью синтаксических преобразований исходной Promela-модели.

2.2. Абстрактные преобразования элементов множеств Act_i

Множество выражений ограничено множеством $Cond(Var, Chan)$. Преобразования присваиваний и выражений приведены в табл. 1 и 2 соответственно. Преобразование операторов assert и print осуществляется посредством преобразования выражений, используемых в этих операторах. Символ $\varnothing$ означает устранение соответствующего оператора из модели с образованием петли из текущего состояния графа программы, защита которой всегда истинна, а действие является пустым.

Таблица 1

Преобразование присваиваний

Оператор в исходной модели	Оператор в абстрактной модели
$v = val$ $(v \in Var, val \in dom(v))$	$v = val$, если $v \in \bigcup_{i=0,1,2} Var_i$, $\varnothing$, если $v \in \bigcup_{i>2} Var_i \setminus \bigcup_{i=0,1,2} Var_i$

Таблица 2

Преобразование выражений

Оператор в исходной модели	Оператор в абстрактной модели
(v,c) $((v,c) \in Cond(V,C))$	(v,c), *если* $V = \bigcup_{i=0,1,2} Var_i$, и $C = \bigcup_{i=0,1,2} Chan_i$, $(v,true)$, *если* $V = \bigcup_{i=0,1,2} Var_i$, и $C = \bigcup_{i>2} Chan_i \setminus \bigcup_{i=0,1,2} Chan_i$, $(true,c)$, *если* $V = \bigcup_{i>2} Var_i \setminus \bigcup_{i=0,1,2} Var_i$, и $C = \bigcup_{i=0,1,2} Chan_i$, $(true,true)$, *если* $V = \bigcup_{i>2} Var_i \setminus \bigcup_{i=0,1,2} Var_i$, и $C = \bigcup_{i>2} Chan_i \setminus \bigcup_{i=0,1,2} Chan_i$

2.3. Абстрактные преобразования элементов множеств $Comm_i$

Предполагается, что по каналу можно передавать сообщения, являющиеся парами $m = \langle opc, sender_{id} \rangle$, где *opc* – код операции (например, соответствующий исходному запросу определенного типа), а $sender_{id}$ – идентификатор отправителя. В абстрактной модели сообщение m_{abs} получается следующим образом: *opc* остается неизменным, а $sender_{id} = i$ для $i = 0,1,2$ и $sender_{id} = ABS$ для $i > 2$, где $ABS > 2$ – некоторая константа. Преобразования коммуникационных действий приведены в табл. 3.

Таблица 3

Преобразование коммуникационных действий

Оператор в исходной модели	Оператор в абстрактной модели
$c!m$ $(m \in dom(c))$	$c!m_{abs}$, если $c \in \bigcup_{i=0,1,2} Chan_i$, $\varnothing$, если $c \in \bigcup_{i>2} Chan_i \setminus \bigcup_{i=0,1,2} Chan_i$
$c?\langle x_1, x_2 \rangle$ $(x_i \in Var,\ i = 1,2,$ $dom(x_1) \supseteq dom(c.opc),\ dom(x_2) \supseteq dom(c.sender_{id}))$	$c?\langle x_1, x_2 \rangle$, если $c \in \bigcup_{i=0,1,2} Chan_i$, $\varnothing$, если $c \in \bigcup_{i>2} Chan_i \setminus \bigcup_{i=0,1,2} Chan_i$

3. Математическое доказательство корректности процедуры абстракции

Система переходов, соответствующая абстрактной канальной системе, $TS(CS_{abs}) = (S_{abs}, Act_{abs}, \rightarrow_{abs}, I_{abs}, AP, L_{abs})$, определяется на основе исходной системы $TS(CS) = (S, Act, \rightarrow, I, AP, L)$ и функции абстракции $f : S \rightarrow S_{abs}$ следующим образом: S_{abs}

определено функцией $f : S \to S_{abs}$, Act_{abs} определено в таблицах 1-3, $\to_{abs}$ определено правилом $\dfrac{s \overset{\alpha}{\to} s'}{f(s) \overset{\alpha}{\to}_{abs} f(s')}$, $I_{abs} = \{ f(s) \mid s \in I \}$, $\forall s \in S : L_{abs}(f(s)) = L(s)$.

Теорема. Абстрактная система переходов симулирует исходную систему переходов.

Доказательство. Отношение $\mathsf{R} = \{(s, f(s)) \mid s \in S\}$ является отношением симуляции для (TS, TS_{abs}). Установим истинность данного утверждения на основании определения отношения симуляции.

Начальным состояниям исходной модели соответствуют начальные состояния абстрактной модели. Пусть $(s, f(s)) \in \mathsf{R}$ и существует переход $s \overset{\alpha}{\to} s'$, порожденный одним из переходов $l_i \overset{g:\alpha}{\Rightarrow} l_i'$, где i – номер графа программы исходной модели. Данный переход описывается одним из правил, приведенных в разделе 1.4. Абстрактные преобразования построены таким образом, что для всех случаев существует соответствующее правило, описывающее переход в абстрактной системе. Если $i = 0,1,2$, то для абстрактной системы имеем аналогичное правило перехода. Если $i > 2$, и $L(s') \neq L(s)$, то в абстрактной системе существует переход, порожденный переходом в графе PG_3, защита которого является логическим следствием защиты g, а действие – α. Если $i > 2$, и $L(s') = L(s)$, то в абстрактной системе состояния s и s' объединяются в состояние $f(s)$, и добавляется петля $f(s) \overset{true}{\to} f(s')$. Таким образом, в абстрактной системе всегда существует состояние $f(s')$ такое, что $f(s) \to f(s')$ и $(s', f(s')) \in \mathsf{R}$.

Заключение

Предложенные абстрактные преобразования формальных моделей протоколов когерентности памяти позволяют уменьшить число процессов верифицируемой модели с большого числа до нескольких (в данной работе до четырех). Синтаксическая природа преобразований позволяет их автоматизировать, а всю работу по созданию и анализу соот-

ветствующей системы переходов предоставить средству автоматизированной проверки моделей Spin. Это позволяет переиспользовать все реализованные в нем алгоритмы, включая множество алгоритмов оптимизации, с помощью которых можно далее сократить число состояний исследуемой системы переходов.

При выполнении работы автор разработал подход к описанию протоколов когерентности, который позволил лаконично описать протокол когерентности системы «Эльбрус-4С» и провести верификацию системы из трех ядер. Применение предложенных абстрактных преобразований к модели с большим числом ядер показало, что преобразования необходимо развить дальше, чтобы избавиться от чрезмерного объема каналов, который все еще присутствует в получаемых абстрактных моделях.

Данная коррекция, а также автоматизация процесса верификации являются направлениями дальнейших исследований.

Литература

1. Буренков В.С. Анализ применимости формальных методов к верификации протоколов когерентности кэш-памяти масштабируемых систем // Вопросы радиоэлектроники. – 2015. – Сер. ЭВТ. – Вып. 1. – С. 105–116.

2. Буренков В.С. Анализ применимости инструмента Spin к верификации протоколов когерентности памяти // Вопросы радиоэлектроники. – 2013. – Сер. ЭВТ. – Вып. 3. – С. 126–134.

3. Baier C., Katoen J-P. Principles of Model Checking. – MIT Press, 2008, 984 pp.

4. Holzmann G. The Spin Model Checker: Primer and Reference Manual. – Addison-Wesley Professional, 2003, 608 pp.

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