Модел. и анализ информ. систем. Т. **17**, № **4** (2010) 78–87

УДК 519.7

Адаптивная редукция симметричных моделей в задаче верификации моделей программ для логики линейного времени

Коннов И.В., Захаров В.А.

Московский государственный университет им. М.В. Ломоносова

e-mail: konnov@cs.msu.su, zakh@cs.msu.su

получена 6 сентября 2010

Ключевые слова: верификация, логика линейного времени, симметрия

Показано, что метод адаптивной редукции симметричных моделей (ASR), предложенный в статье [9] для сокращения пространства поиска при решении проблемы достижимости в моделях программ, может быть с равным успехом применен для проверки выполнимости формул логики линейного времени ILTL на конечных моделях распределённых программ. Задача проверки выполнимости формул ILTL сводится к задаче проверки пустоты автомата Бюхи, решаемой при помощи комбинированного алгоритма, в котором процедура двойного поиска в глубину (DDFS) сочетается с последовательным уточнением пространства поиска на основе принципов ASR.

1. Введение

Основная трудность, с которой сталкиваются все инструментальные средства верификации моделей программ, — это проблема комбинаторного взрыва: число состояний размеченных систем переходов, используемых в качестве абстрактных моделей распределенных программ (систем взаимодействующих процессов), возрастает экспоненциально с увеличением числа процессов, из которых состоит программа. Для преодоления этой трудности наряду с методами символьных вычислений применяют разнообразные приемы, позволяющие в отдельных случаях сократить размер проверяемой модели за счет использования тех или иных особенностей ее устройства. К числу таких приемов относятся метод абстракции данных [3], редукция частичных порядков [8], редукция симметричных моделей [6] и др. Например, метод редукции симметричных моделей опирается на следующее алгебраическое свойство моделей: для любой группы автоморфизмов G системы размеченных переходов M фактор-модель M/G бисимуляционно эквивалентна модели M. Состояния фактор-модели M/G — это орбиты группы автоморфизмов G модели M; чем больше автоморфизмов имеет модель M, тем меньше размер фактор-модели M/G.

К сожалению, задача обнаружения нетривиальных автоморфизмов произвольной размеченной системы переходов является NP-полной. Возникает вопрос: нельзя ли осуществить построение сокращенной модели M/G «на лету», уточняя устройство орбит фактор-модели по ходу ее верификации? В статье [9] была предпринята попытка реализовать этот подход для проверки достижимости состояний в симметричных моделях. Суть предложенного метода адаптивной редукции симметричных моделей такова. Вначале предполагается, что модель M является вполне симметричной (т.е. все процессы взаимозаменяемы), и поэтому множество состояний модели M распадается на сравнительно небольшое число классов, которые называются метасостояниями. Каждое метасостояние — это некоторая аппроксимация орбит фактор-модели M/G. По мере построения маршрута в модели M эта аппроксимация уточняется: если при рассмотрении очередного перехода α обнаруживается, что устройство M несовместно с предположением о включении метасостояния $\widehat{s}$ в орбиту группы автоморфизмов модели M, то $\widehat{s}$ разбивается на подклассы $\widehat{s}_1, \widehat{s}_2, \ldots, \widehat{s}_n$, согласованные с переходом α.

Успешная разработка и применение метода адаптивной редукции симметричных моделей (ASR) для проверки достижимости состояний модели позволяет надеяться, что этот подход можно применять и для решения более общей задачи — проверки выполнимости формул логики линейного времени (LTL) на конечных размеченных системах переходов. В настоящей статье предложен новый теоретико-автоматный метод верификации моделей программ. Используя приемы, описанные в статье [9], мы сводим задачу проверки выполнимости формулы LTL φ на модели M к задаче проверки пустоты автомата Бюхи $A_{\varphi,M}$. Затем для проверки пустоты автомата $A_{\varphi,M}$ применяется оригинальный алгоритм, сочетающий двойной поиск в глубину с возвратом (DDFS) [5] и адаптивную редукцию симметрии [9]. В отличие от алгоритма DDFS для симметричных систем, описанного в работе [1], предложенный алгоритм не требует предварительного вычисления орбит модели.

2. Основные определения

Мы полагаем, что читатель знаком с наиболее важными математическими конструкциями, относящимися к задаче верификации моделей программ для темпоральных логик (см. [3]). Поэтому мы ограничимся лишь определениями основных понятий, относящихся непосредственно к методу адаптивной редукции симметрии. Эти определения заимствованы из основополагающей работы [9].

Мы рассматриваем распределенные программы, состоящие из n параллельных взаимодействующих процессов. Чтобы воспользоваться преимуществами симметрии для сокращения пространства поиска при анализе модели программы, множество процессов программы нужно разбить на классы. *Разбиением* множества $\mathcal{N}_n = \{1, \ldots, n\}$ будем называть всякое семейство $P = \{C_i\}_{i=1}^m$ попарно непересекающихся подмножеств, объединение которых покрывает $\mathcal{N}_n$. Элементы разбиения называются *ячейками*. Для обозначения разбиений используются записи вида $|\,1,3\,|\,2,4\,|\,5\,|$, где $\{1,3\}$, $\{2,4\}$ и $\{5\}$ — это ячейки. Символом $\mathcal{P}_n$ обозначим множество всех разбиений множества $\mathcal{N}_n$. Всякое разбиение P порождает группу переста-

новок $\langle P\rangle$ на множестве $\mathcal{N}_n$: перестановка π принадлежит группе $\langle P\rangle$ тогда и только тогда, когда для любого числа i из $\mathcal{N}_n$ оба числа i и $\pi(i)$ принадлежат одной и той же ячейке. На множестве разбиений $\mathcal{P}_n$ вводится частичный порядок $\sqsubseteq$: отношение $P \sqsubseteq P'$ выполняется тогда и только тогда, когда $\langle P\rangle \subseteq \langle P'\rangle$. Упорядоченное таким образом множество разбиений является решеткой. Запись $P \sqcup P'$ обозначает точную нижнюю грань разбиений P и P'.

В качестве формальных моделей распределенных программ будут использоваться модели Крипке (размеченные системы переходов), которые индуцируются обобщенным описанием. Не ограничивая общности, мы предполагаем, что все процессы имеют одно и то же множество локальных состояний $\mathcal{L} = \{1, \ldots, \ell\}$. *Глобальное состояние* модели — это упорядоченный набор локальных состояний всех процессов. *Обобщенным описанием* $\mathcal{D}$ распределенной программы называется конечная система *метапереходов* вида $A \xrightarrow{\varphi,P} B$, где

- A и B — локальные состояния из множества $\mathcal{L}$,
- $\varphi(x, y)$ — двухместное отношение (*предикат-предохранитель*), первый аргумент которого — это глобальное состояние из множества $\mathcal{L}^n$, а второй аргумент — это номер активного процесса y, $y \in \mathcal{N}_n$,
- P — разбиение из множества $\mathcal{P}_n$ (*инвариант метаперехода*).

Операционная семантика метапереходов такова: если программа пребывает в глобальном состоянии s и при этом $\varphi(s, i) = true$, то процесс i может перейти из локального состояния A в локальное состояние B. О назначении разбиения P будет говориться далее.

Обобщенному описанию $\mathcal{D}$ распределенной программы соответствует модель Крипке $M_{\mathcal{D}} = \langle S, s_0, \to, L\rangle$, состоящая из

- множества глобальных состояний $S = \mathcal{L}^n$,
- начального состояния $s_0 = (1, 1, \ldots, 1)$,
- отношения переходов $\to \subseteq S \times S$, которое определено согласно следующему правилу:
  переход $(s_1, \ldots, s_n) \to (t_1, \ldots, t_n)$ возможен тогда и только тогда, когда существует такой индекс процесса i и такой метапереход $s_i \xrightarrow{\varphi,P} t_i$, для которого $\varphi((s_1, \ldots, s_n), i) = true$ и при этом $t_j = s_j$ для всех индексов j, $j \neq i$.

Для глобального состояния $s = (s_1, \ldots, s_n)$ и перестановки π на множестве $\mathcal{N}_n$ запись $\pi(s)$ будет обозначать глобальное состояние $(s_{\pi(1)}, \ldots, s_{\pi(n)})$. Будем говорить, что метапереход $A \xrightarrow{\varphi,P} B$ является *корректным*, если равенство $\varphi(s, i) = \varphi(\pi(s), \pi(i))$ выполняется для любого глобального состояния s, процесса i и перестановки π из группы $\langle P\rangle$. Проверка корректности метапереходов — это отдельная задача, для решения которой могут быть привлечены алгоритмы проверки выполнимости логических формул. Решение данной задачи выходит за рамки настоящей статьи; в дальнейшем мы будем полагать, что все метапереходы, составляющие обобщенное описание программы, корректны.

Особенности симметричного устройства модели программы могут использоваться для эффективного решения задачи проверки достижимости. Для этого автор статьи [9] предложил обратиться к расширенному пространству состояний $\widehat{S} = \{1, \ldots, \ell\}^n \times \mathcal{P}_n$. Элементы множества $\widehat{S}$ будем называть *метасостояниями*. Фактически, каждое метасостояние $\widehat{s} = (s, P)$ определяет *орбиту* — множество глобальных состояний $orbit(\widehat{s}) = \{t \in S : \exists \pi(\pi \in \langle P \rangle \wedge \pi(s) = t)\}$. Например, множество $orbit((1,2,2), |\,1,2\,|\,3\,|)$ равно $\{(1,2,2),(2,1,2)\}$. Метасостояние $\widehat{s} \in \widehat{S}$ *поглощается* метасостоянием $\widehat{t} \in \widehat{S}$ (этот факт обозначается записью $\widehat{s} \lhd \widehat{t}$), если $orbit(\widehat{s}) \subseteq orbit(\widehat{t})$.

Для описания поведения распределенных программ рассматриваемого вида целесообразно использовать *индексированную логику линейного времени* (ILTL), описанную в статье [2]. Предположим, что распределенная программа состоит из n параллельных процессов, каждый из которых имеет ℓ локальных состояний $\{1, 2, \ldots, \ell\}$. *Атомарным высказыванием* назовем всякую пару (i, j), $i \in \mathcal{N}_n$, $j \in \mathcal{L}$; эта пара обозначает событие пребывания процесса i в локальном состоянии j. Запись AP будет обозначать множество всех атомарных высказываний для заданной распределенной программы. Формулы ILTL строятся из атомарных высказываний при помощи булевых связок и темпоральных операторов LTL. Формула $\psi(i)$ называется *базовой* формулой, если первой компонентой всех ее атомарных высказываний является одно и то же число i. Формула ψ называется *замкнутой*, если каждое атомарное высказывание (i, j), входящее в состав ψ, встречается только в подформулах вида $\bigvee\limits_{i=1}^{n} \psi(i)$ и $\bigwedge\limits_{i=1}^{n} \psi(i)$. Выполнимость атомарных формул определяется так: отношение $M, s \models (i, j)$ имеет место для заданной модели Крипке M и глобального состояния $s = (s_1, s_2, \ldots, s_n)$ тогда и только тогда, когда $s_i = j$. Семантика прочих формул ILTL определяется традиционным для темпоральных логик способом (см. [4]).

Задача верификации моделей программ состоит в том, чтобы для заданного обобщенного описания $\mathcal{D}$ распределенной программы, состоящей из n параллельных процессов и замкнутой ILTL формулы ψ, проверить отношение выполнимости $M_{\mathcal{D}}, s_0 \models \psi$.

Для решения поставленной задачи верификации моделей программ мы используем теоретико-автоматный подход. Для заданной темпоральной формулы φ строится автомат Бюхи $\mathcal{B}_{\neg\psi}$, допускающий те и только те вычисления моделей программ, на которых не выполняется формула φ. Модель программы M также рассматривается как автомат Бюхи $\mathcal{B}_M$, допускающий все вычисления модели программы. Вслед за этим проводится проверка пустоты композиции автоматов $\mathcal{B}_M$ и $\mathcal{B}_{\neg\psi}$, допускающей пересечение языков $L(\mathcal{B}_M)$ и $L(\mathcal{B}_{\neg\psi})$.

Отличительная особенность предложенного нами метода состоит в том, что мы избегаем построения автомата $\mathcal{B}_M$, размер которого может быть очень велик. Вместо поиска подходящего пути (последовательности переходов) в пространстве S глобальных состояний модели $M_{\mathcal{D}}$, мы осуществляем построение и анализ псевдовычислений — последовательностей метапереходов обобщенного описания программы $\mathcal{D}$ в пространстве метасостояний $\widehat{S}$. Интерес представляют лишь корректные псевдовычисления в пространстве метасостояний, покрывающие пути в модели $M_{\mathcal{D}}$. Чтобы соблюсти требования корректности, по ходу формирования псевдовычислений приходится расщеплять метасостояния, используя для этого предикаты-

предохранители и инварианты метапереходов. Совместив процедуру расщепления метасостояний (ASR) [9] и алгоритм двойного поиска в глубину (DDFS) [5], нам удалось построить новый алгоритм ASR-DDFS, позволяющий проводить более эффективный поиск компонент сильной связности в симметричных моделях Крипке.

3. Трансляция формул ILTL в автоматы Бюхи

Для трансляции ILTL формулы ψ в автомат Бюхи $\mathcal{B}_{\neg\psi}$ можно воспользоваться известным табличным алгоритмом [7]. Но для корректного применения правил редукции симметричных моделей состояния автомата Бюхи необходимо аннотировать ограничениями симметричности. *Аннотированным автоматом Бюхи* называется шестёрка $\mathcal{B} = \langle Q, Q_0, \delta, F, \lambda, \xi\rangle$, в которой Q — множество состояний автомата, $Q_0 \subseteq Q$ — множество начальных состояний, $\delta \subseteq Q \times Q$ — *отношение перехода*, $F \subseteq Q$ — множество *допускающих состояний*, $\lambda : Q \to 2^{AP}$ — *функция разметки*, а $\xi : Q \to \mathcal{P}_n$ — ограничение симметричности. Как обычно, автомат $\mathcal{B}$ *допускает* сверхслово $w_1 w_2 \dots w_i \dots$ из множества $(2^{AP})^\omega$ тогда и только тогда, когда найдётся такой *допускающий прогон* $\sigma = q_1, q_2, \dots, q_i, \dots$, что $q_1 \in Q_0$, $(q_i, q_{i+1}) \in \delta$ и $w_i \in \lambda(q_i)$ для всех $i \geq 1$, и, кроме того, хотя бы одно допускающее состояние $q \in F$ появляется бесконечно часто в последовательности σ.

Ограничение симметричности ξ используется только при построении синхронной композиции модели $\mathcal{B}_M$ и автомата $\mathcal{B}_{\neg\psi}$, когда для корректного выполнения метапереходов требуется расщеплять разбиения метасостояний. Далее мы будем рассматривать автоматы с *согласованным* ограничением симметричности ξ, удовлетворяющим соотношению $(i,j) \in \lambda(q) \Leftrightarrow (\pi(i), j) \in \lambda(q)$ для любых $q \in Q$, $\pi \in \langle \xi(q)\rangle$, $i \in \mathcal{N}$ и $j \in \mathcal{L}$. В связи с этим в табличный алгоритм, описанный в статье [7], вносятся следующие изменения:

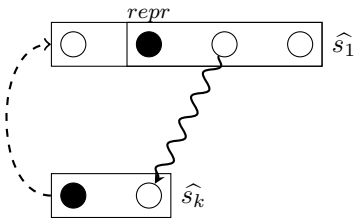
- Для состояний q, соответствующих атомарным высказываниям (i,j), значение $\xi(q)$ полагается равным $|\, i \,|\, 1, \dots, i-1, i+1, \dots, n \,|$, за исключением случая, когда j — связанная переменная объемлющей формулы $\bigwedge\limits_{i=1}^{n} \psi(i)$. В последнем случае $\xi(q)$ полагается равным $|\, 1, \dots, n \,|$.

- Для состояний q, соответствующих формулам $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 \,\mathsf{U}\, \psi_2$, $\psi_1 \,\mathsf{R}\, \psi_2$, значение $\xi(q)$ полагается равным $\xi(q_1) \sqcup \xi(q_2)$, где q_1, q_2 — состояния, соответствующие подформулам ψ_1 и ψ_2.

- Для состояний q, соответствующих формулам $\bigvee\limits_{i=1}^{n} \psi(i)$, значение $\xi(q)$ полагается равным $\xi(q) = |\, 1 \,| \dots |\, n \,|$, если в формуле содержатся темпоральные операторы, и равным $|\, 1, \dots, n \,|$ в противном случае.

- При обработке $\bigwedge\limits_{i=1}^{n} \psi(i)$ ограничение $\xi(q)$ формирующегося состояния q полагается равным $\xi(q_1)$, где q_1 — состояние, соответствующее подформуле $\psi(i)$.

Рис. 1: Иллюстрация понятия выполнимого псевдоцикла.



- Для состояний, соответствующих формулам вида $\neg\psi$ и $\mathsf{X}\ \psi$, значение $\xi(q)$ полагается равным $\xi(q_1)$, где q_1 — состояние, соответствующее подформуле ψ.

4. Алгоритм ASR-DDFS

Алгоритм ASR-DDFS предназначен для проверки пустоты языка автомата Бюхи $\mathcal{B}_M \otimes \mathcal{B}_{\neg\psi}$. Поиск допускающего прогона проводится в пространстве пар $\widehat{S} \times Q$, где $\widehat{S}$ — множество метасостояний модели M, а Q — множество состояний аннотированного автомата Бюхи $\mathcal{B}_{\neg\psi}$. Элементы множества $\widehat{S} \times Q$ будем называть *SQ-парами*. Каждой SQ-паре $(\widehat{s}, q)$ соответствует множество состояний $orbit(\widehat{s}) \times \{q\}$ автомата $\mathcal{B}_M \otimes \mathcal{B}_{\neg\psi}$. На множестве SQ-пар вводится отношение перехода $(\widehat{s}, q) \xrightarrow{\alpha}_{\mathrm{ASR}} (\widehat{s'}, q')$. Две SQ-пары $((s, P), q)$ и $((s', P'), q')$ находятся в отношении $(\widehat{s}, q) \xrightarrow{\alpha}_{\mathrm{ASR}} (\widehat{s'}, q')$ для метаперехода $\alpha : A \xrightarrow{\varphi,Q} B$ тогда и только тогда, когда: 1) найдутся такие состояния $t \in orbit(s, P)$ и $t' \in orbit(s', P')$ автомата $\mathcal{B}_M$ и индекс процесса i, $1 \le i \le n$, что $t \xrightarrow{\alpha} t'$, то есть верно $\varphi(t, i)$, $t_i = A$, $t'_i = B$ и $t'_j = s'_j$ для всех $j : 1 \le j \le n \wedge j \ne i$; 2) $P' = P \sqcup Q \sqcup \xi(q')$; 3) $L(s') \in \lambda(q')$.

Всякая конечная последовательность переходов $(\widehat{s}_1, q_1) \xrightarrow{\alpha_1}_{\mathrm{ASR}} \ldots \xrightarrow{\alpha_{k-1}}_{\mathrm{ASR}} (\widehat{s}_k, q_k)$ будет называться *псевдовычислением*. Чтобы отыскать допускающий прогон автомата $\mathcal{B}_M \otimes \mathcal{B}_{\neg\psi}$, алгоритм ASR-DDFS строит и анализирует *псевдоциклы* — псевдотрассы $(\widehat{s}_1, q_1) \xrightarrow{\alpha_1}_{\mathrm{ASR}} \ldots \xrightarrow{\alpha_{k-1}}_{\mathrm{ASR}} (\widehat{s}_k, q_k)$, в которых $\widehat{s}_k \lhd \widehat{s}_1$ и $q_k = q_1$.

Псевдоцикл называется *выполнимым*, если существует такое состояние $repr \in S$, что $repr \in orbit(\widehat{s}_1)$, $repr \in orbit(\widehat{s}_k)$ и для некоторого i, $1 \le i \le n$, верно $\varphi_1(repr, i)$, где $\alpha_i : A \xrightarrow{\varphi_i,Q} B$. Рисунок 1 иллюстрирует данное понятие. Волнистой линией изображается путь в пространстве поиска, а пунктиром — отношение поглощения между SQ-парами. Состояния внутри одного прямоугольника принадлежат орбите одного метасостояния. Как только выполнимый псевдоцикл обнаружен, алгоритм прекращает работу и сообщает о существовании цикла, содержащего допускающее состояние, в автомате Бюхи $\mathcal{B}_M \otimes \mathcal{B}_{\neg\psi}$.

Поиск псевдоциклов начинается из SQ-пары $((s_0, |\, 1, \ldots, n\,|), q_0)$, где s_0 — начальное глобальное состояние программы, q_0 — начальное состояние автомата $\mathcal{B}_{\neg\psi}$. Эта SQ-пара содержит начальное состояние автомата $\mathcal{B}_{\neg\psi}$. Процедура внешнего поиска dfs строит псевдотрассы, оканчивающиеся SQ-парой $(\widehat{s}^*, q^*)$, содержащей некоторое допускающее состояние q^* автомата $\mathcal{B}_{\neg\psi}$. Процедура вложенного поиска $ndfs$ ищет выполнимый псевдоцикл, начинающийся SQ-парой $(\widehat{s}^*, q^*)$, обнаружен-

ной процедурой dfs. Функция *successors* вычисляет множество SQ-пар, следующих за SQ-парой $((s,P),q)$ и представителей орбит, порождающих переход, то есть $\{(u,(\widehat{t},r)) \mid ((s,P),q) \to_{\mathrm{ASR}} (\widehat{t},r) \wedge u \in orbit(s,P) \wedge \exists v : v \in orbit(\widehat{t}) \wedge u \to v\}$. Представители орбит используются для проверки выполнимости псевдоцикла.

Листинг 1: Процедура dfs

```
1  global repr, ŝ*, q*, ndfs_queue;
2
3  procedure dfs(ŝ, q)
4   if ∃ŵ((ŵ,q) ∈ Visited ∧ ŝ ◁ ŵ)
5    return; //  откат
6   Visited := Visited ∪ {(ŝ,q)};
7   forall (_,(ŝ',q')) ∈ successors(ŝ,q)
8    dfs(ŝ', q');
9   if q ∈ F
10   ndfs_queue := [(ŝ, q)];
11   while ndfs_queue != []
12    //  первый вложенный поиск
13    //  или уточняющий
14    pop (ŝ*, q*) from ndfs_queue;
15    Visited2 := ∅;
16    ndfs(ŝ*, q*);
```

Листинг 2: Процедура ndfs

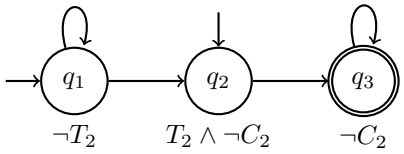
```
17 procedure ndfs(ŝ, q)
18  if ∃ŵ((ŵ,q) ∈ Visited2 ∧ ŝ ◁ ŵ)
19   return //  откат
20  Visited2 := Visited2 ∪ {(ŝ,q)};
21  forall (u,(ŝ',q')) ∈ successors(ŝ,q)
22   (s, P) := ŝ; (s', P') := ŝ';
23   if (ŝ,q) = (ŝ*,q*)
24    repr := u;
25   if q' = q* ∧ ŝ' ◁ ŝ* ∧
26    if repr ∈ orbit(ŝ')
27     report pseudo-cycle; exit;
28    else
29     push (ŝ', q') to ndfs_queue;
30     Visited2 := Visited2 ∪ {(ŝ',q')};
31   else
32    ndfs(ŝ', q');
```

Листинг 3: Функция successors

```
33 function successors((s, P), q)
34  succ := ∅
35  for all meta-transitions A --φ,P'--> B
36   for all q' ∈ δ(q)
37    R := P ⊔ P' ⊔ ξ(q')
38    U := unwind(s, P, R) //  найти U : ⋃_{u∈U} orbit(u,R) = orbit(s,P)
39    for all states u ∈ U do
40     for all cells c ∈ R do
41      if ∃i ∈ c : u_i = A ∧ φ(u,i)
42       v := (u_1, ..., u_{i-1}, B, u_{i+1}, ..., u_n)
43       if L(v) ∈ λ(q') //  согласован с B_¬ψ
44        canonicalize(v, R) //  сортировка состояний в ячейках
45        succ := succ ∪ {(u,(v,R,q'))}
46  return succ
```

Теорема 1 (Корректность). *Пусть алгоритм ASR-DDFS обнаружил выполнимый псевдоцикл* $(\widehat{u}_1,r_1) \to_{\mathrm{ASR}} \ldots \to_{\mathrm{ASR}} (\widehat{u}_m,r_m) \to_{\mathrm{ASR}} (\widehat{u}_{m+1},r_{m+1})$. *В таком случае в автомате* $\mathcal{B}_M \otimes \mathcal{B}_{\neg\psi}$ *найдётся вычисление, в котором бесконечно часто появляется допускающее состояние* r_1.

Рис. 2: Автомат Бюхи для отрицания формулы $\varphi = \mathsf{G}\,(T_2 \to \mathsf{F}\,C_2)$.



Теорема 2 (Полнота). *Если язык автомата $\mathcal{B}_M \otimes \mathcal{B}_{\neg\psi}$ не пуст, то поиск ASR-DDFS обнаружит выполнимый псевдоцикл.*

Пример. В качестве примера рассмотрим параметризованный алгоритм критической секции, схожий с примером [9], однако в нашем примере первый процесс будет иметь наивысший приоритет доступа к критической секции, второй — средний приоритет, остальные процессы — низший приоритет. Для удобства сопоставим локальным состояниям процессов мнемонические имена: $1 \to N$ (нейтральное), $2 \to T$ (запрос секции), $3 \to C$ (критическая секция). Система представляется следующим обобщённым описанием (локальный переход, условие, разбиение):

$$
\begin{array}{ll}
N \to T, true, & |\,1, \ldots, n\,| \\
C \to N, true, & |\,1, \ldots, n\,| \\
T \to C, i = 1 \wedge \forall j : 1 \le j \le n . s_j \ne C, & |\,1\,|\,2, \ldots, n\,| \\
T \to C, i = 2 \wedge s_1 \ne T \wedge \forall j : 1 \le j \le n . s_j \ne C, & |\,2\,|\,1, 3, \ldots, n\,| \\
T \to C, i > 2 \wedge s_1 \ne T \wedge s_2 \ne T \wedge \forall j : 1 \le j \le n . s_j \ne C, & |\,1, 2\,|\,3, \ldots, n\,|
\end{array}
$$

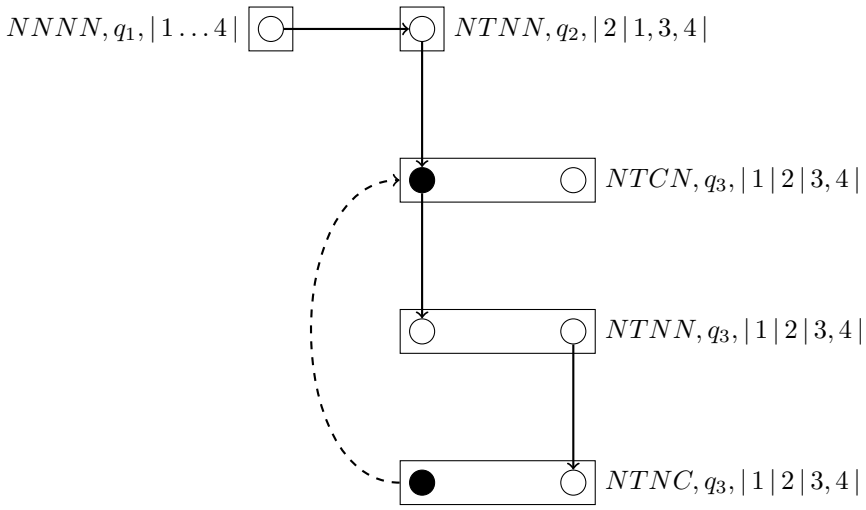
Рассмотрим одно из требований, предъявляемых к алгоритму, а именно свойство обязательного перехода процесса из состояния T в состояние C, например, $\varphi = \mathsf{G}\,(T_2 \to \mathsf{F}\,C_2)$. Диаграмма Мура для отрицания формулы φ приведена на рисунке 2.

На рисунке 3 приведён пример поиска ASR-DDFS для $n = 4$ с выполнимым псевдоциклом.

5. Заключение

В статье описан алгоритм двойного поиска в глубину, применимый для обнаружения сильных компонент связности в размеченных системах переходов с предусловиями их выполнения, неявно описывающими ограничения симметрии. Ограничения на объем статьи не позволяют привести полные доказательства теорем корректности и полноты. Заметим лишь, что корректность алгоритма следует из определения выполнимого псевдоцикла и конечности числа возможных перестановок индексов процессов. Обоснование полноты опирается на свойство монотонного убывания по отношению поглощения орбит тех метасостояний, которые возникают по ходу работы алгоритма. Оценка эффективности алгоритма, проведение экспериментальных исследований и очерчивание области применения предложенного алгоритма — это

Рис. 3: Двойной поиск с применением ASR и выполнимый псевдоцикл.



тема нашей последующей работы. Особого внимания также заслуживает вопрос о том, в какой мере в новом алгоритме могут быть учтены свойства симметричности проверяемых спецификаций — формул ILTL.

Список литературы

1. Bosnacki D. Nested depth first search algorithm for model checking with symmetry reduction // *Proceedings of FORTE 2002*, 2002, Lecture Notes in Computer Science. V. 2529/2002. P. 65–80.

2. Browne M.C., Clarke E.M., Grumberg O. Reasoning about networks with many identical finite-state processes // *Information and Computation*. 1989. 81(1). P. 13–31.

3. Clarke E.M., Grumberg O., Long O.E. Model checking and abstraction // *ACM Transactions on Programming Languages and Systems*. 1994. V. 16, N 5. P. 1512–1542.

4. Кларк Э.М., Грамберг О., Пелед Д. Верификация моделей программ: Model Checking. М.: Изд-во МЦНМО, 2002. 416 с.

5. Courcoubetis C., Vardi M.Y., Wolper P., Yannakakis M. Memory efficient algorithms for the verification of temporal properties // *Formal Methods in System Design*. 1992. V. 1. P. 275–288.

6. Emerson E.A., Sistla A.P. Symmetry and model checking // *Proceedings of the 5-nd Workshop on Computer-Aided Verification* 1993, 1993, Lecture Notes in Computer Science. V. 697/1993. P. 463–478.

7. Gerth R., Peled D., Vardi M., and Wolper P. Simple on-the-fly automatic verification of linear temporal logic // *Protocol Specification Testing and Verification*. Warsaw, Poland, 1995. P. 3–18.

8. Godefroid P. Using Partial orders to improve automatic verification methods. *Proceedings of the 2-nd Workshop on Computer-Aided Verification. ACM Transactions on Programming Languages and Systems*, 1990, Lecture Notes in Computer Science. V. 531/1990. P. 176-185.

9. Wahl T. Adaptive symmetry reduction. Proceedings of *Computer Aided Verification* 2007, Lecture Notes in Computer Science. V. 4590/2007. P. 393–405.

The application of adaptive symmetry reduction for LTL model checking

Konnov I.V., Zakharov V.A.

Keywords: model checking, LTL, symmetry

Adaptive symmetry reduction is a technique which exploits the similarity of components in systems of regular structure. It helps to reduce the effect of state explosion when exploring reachable states of a system. It assumes the perfect symmetry of states initially and tracks symmetry violations on-the-fly by exploring an extended state space. In this paper we show that the technique is applicable to LTL model checking as well. To succeed in this we incorporate the operations over the extended state space into the classic double depth-first search algorithm. The nested search is modified to detect a feasible pseudo-cycle via an accepting state of Buchi automaton instead of a cycle. We show that the existence of a pseudo-cycle results in satisfiability of an Indexed LTL formula on a model of the system and vice versa. This result opens the way for implementing adaptive symmetry reduction in a LTL model checker.

Сведения об авторах:
Коннов Игорь Владимирович,
МГУ им. М.В. Ломоносова, факультет ВМиК, мл. науч. сотр.;
Захаров Владимир Анатольевич,
МГУ им. М.В. Ломоносова, факультет ВМиК, доцент