

УДК 004.832.23+004.942

*А.В. Колчин, А.А. Летичевский, С.В. Потиеенко*

Институт кибернетики имени В.М. Глушкова НАН Украины, Украина  
Украина, 03680 МСП, г. Киев, просп. Академика Глушкова, 40

**СТАТИЧЕСКИЙ МЕТОД  
УСТРАНЕНИЯ ИЗБЫТОЧНЫХ ИНФОРМАЦИОННЫХ  
СВЯЗЕЙ В ПРЕДУСЛОВИЯХ ПЕРЕХОДОВ  
ФОРМАЛЬНЫХ МОДЕЛЕЙ  
ТРАНЗИЦИОННЫХ СИСТЕМ**

*A. Kolchin, A. Letychevskyu, S. Potiyenko*

V.M. Glushkov Institute of cybernetics NAS of Ukraine, Ukraine  
Ukraine, 03680 MSP, Kiev, Akademika Glushkova ave., 40

**A STATIC METHOD FOR ELIMINATION  
OF REDUNDANT DEPENDENCIES  
IN PRECONDITIONS OF TRANSITIONS OF FORMAL  
MODELS OF TRANSITION SYSTEMS**

*О.В. Колчин, О.О. Летичевський, С.В. Потієнко*

Институт кібернетики ім.В.М. Глушкова НАН України, Україна  
Україна, 03680 МСП, м. Київ, просп. Академіка Глушкова, 40

**СТАТИЧНИЙ МЕТОД УСУНЕННЯ  
НАДЛИШКОВИХ ІНФОРМАЦІЙНИХ ЗВ'ЯЗКІВ У  
ПЕРЕДУМОВАХ ПЕРЕХОДІВ ФОРМАЛЬНИХ  
МОДЕЛЕЙ ТРАНЗИЦІЙНИХ СИСТЕМ**

Цель предлагаемого метода – сократить исследуемое пространство поведения формальных моделей, и, как следствие, повысить эффективность анализа ее свойств. В основу метода положен алгоритм статического символического анализа предусловий переходов модели.

**Ключевые слова:** верификация, редукция пространства поиска, информационная зависимость.

The objective of the proposed method is to reduce search space in the behavior of formal models, and, as a consequence, to increase efficiency of model checking. The core of the method is an algorithm of static symbolic analysis of preconditions of model transitions.

**Key words:** model checking, state space reduction, data dependency.

Мета запропонованого методу – скоротити простір, що досліджується, поведінки формальних моделей, та, як наслідок, підвищити ефективність аналізу її властивостей. В основі методу лежить алгоритм статичного символического аналізу передумов переходів моделі.

**Ключові слова:** верифікація, редукція простору пошуку, інформаційна залежність.

**Введение**

Проверка свойств формальных моделей программных систем сталкивается с проблемой комбинаторного взрыва вариантов поведения [1]. Часто верификаторы выполняют «лишний» перебор, рассматривая различные перестановки фактически независимых участков поведения. Одна из причин этой проблемы лежит в формальном определении и эффективности методов вычисления «независимости». Ввиду сложности вычисления семантической зависимости, на практике часто пользуются критерием синтаксического вхождения. Например, существующие методы устранения

избыточного интерливинга, опираясь на формальные определения семантического отношения независимости (перестановочности) [2, 3] на практике реализуют аппроксимацию на основе синтаксического анализа (например, [4]). Такое вычисление независимости само по себе выполняется эффективно, но в результате фактические информационные связи значительно преувеличиваются, и, как следствие, существенная часть избыточных перестановок не устраняется.

Определение отношения зависимости и алгоритмы его вычисления актуальны для современных методов анализа программ [5–8]. Чем точнее специфицировано это отношение, тем точнее можно рассуждать о тех или иных свойствах программ, строить более удачные абстракции, эффективнее сокращать исследуемое пространство поведения, осуществлять отладку, тестирование, слайсинг, распараллеливание, оптимизацию и т.п.

Непосредственная задача предлагаемого метода – эффективно вычислить достаточную для проверки свойств часть информационной зависимости в формальной модели на уровне пред- и постусловий ее переходов, причем статически, не прибегая к построению пространства ее состояний. Первый раздел статьи описывает мотивирующие примеры, суть предлагаемого метода изложена во втором разделе.

### Мотивирующие примеры

Существующие методы активно используют поток управления и часто предназначены только для структурированных программ. Предлагаемый метод редукции информационных связей особенно актуален для моделей транзиторных систем, заданных неупорядоченным множеством переходов без явно заданного потока управления.

**Пример 1.** Следующий пример типичный для систем управления: главный модуль последовательно обрабатывает информацию с внешних датчиков  $a$  и  $b$ :  
*Атрибуты модели:*  $Cf: \{Fa, Fb\}$ ;  $a\_state: \{1, 2\}$ ;  $b\_state: \{1, 2, 3\}$ .  
*Переходы модели:*

Таблица 1. Переходы модели для примера 1.

Имя перехода	Предусловие	Постусловие
Ta1	$Cf=Fa \ \& \ a\_state=1$	$Cf:=Fb; \ a\_state:=2$
Ta2	$Cf=Fa \ \& \ a\_state=2$	$Cf:=Fb; \ a\_state:=1$
Tb1	$Cf=Fb \ \& \ b\_state=1$	$Cf:=Fa; \ b\_state:=2$
Tb2	$Cf=Fb \ \& \ b\_state=2$	$Cf:=Fa; \ b\_state:=3$
Tb3	$Cf=Fb \ \& \ b\_state=3$	$Cf:=Fa; \ b\_state:=1$

*Начальное состояние:*  $Cf=Fa, \ a\_state=1, \ b\_state=1$ .

Заметим, что состояния датчиков не влияют на последовательность их обработки. Однако синтаксический анализ не выявит такой особенности. Таким образом, наивному алгоритму потребуется построить 13 состояний (последнее будет таким же, как и первое):

Трасса, ведущая к циклу:

Ta1, Tb1, Ta2, Tb2, Ta1, Tb3, Ta2, Tb1, Ta1, Tb2, Ta2, Tb3.

В данном примере количество состояний определяется формулой, выражающей произведение количества датчиков на наименьшее общее кратное для количества их состояний, в данном случае:  $2 * \text{НОК}(2,3)=12$ .

1)	Cf:Fa, a_state:1, b_state:1
2)	Cf:Fb, a_state:2, b_state:1
3)	Cf:Fa, a_state:2, b_state:2
4)	Cf:Fb, a_state:1, b_state:2
5)	Cf:Fa, a_state:1, b_state:3
6)	Cf:Fb, a_state:2, b_state:3
7)	Cf:Fa, a_state:2, b_state:1
8)	Cf:Fb, a_state:1, b_state:1
9)	Cf:Fa, a_state:1, b_state:2
10)	Cf:Fb, a_state:2, b_state:2
11)	Cf:Fa, a_state:2, b_state:3
12)	Cf:Fb, a_state:1, b_state:3
13)	Cf:Fa, a_state:1, b_state:1

Рис. 1. Состояния модели для примера 1.

**Пример 2.** Приведем простую модификацию первого примера: добавим недетерминированные «пустые» переходы  $Ta\_empty, Tb\_empty$ :

Таблица 2. Переходы модели для примера 2.

Имя перехода	Предусловие	Постусловие
$Ta\_empty$	Cf=Fa	Cf:=Fb
$Tb\_empty$	Cf=Fb	Cf:=Fa

Модифицированный пример моделирует параллелизм (изменение значений датчиков происходит по сути асинхронно), а количество состояний растет уже экспоненциально с количеством датчиков. Заметим, что такой интерливинг не будет устранен методами с синтаксическим анализом информационных связей.

### Описание метода

Данный метод развивает работы [9–11] по сокращению анализируемого пространства поведения формальных моделей. Далее приведено описание формальной модели и принципа построения графа зависимости.

### Формальная модель

Пусть задано конечное множество атрибутов  $A = v_1, v_2, \dots, v_n$  и пусть также для каждого атрибута  $v_i \in A$  определена конечная область допустимых значений  $D(v_i)$ .

**Определение 1.** Состоянием называется множество пар атрибутов и их значений (констант) вида  $\{(v = d) : v \in A \wedge d \in D(v)\}$ .

Назовем предусловием некоторую бескванторную формулу логики предикатов над атрибутами множества  $A$  и константами из соответствующих областей допустимых значений  $D$ . Назовем постусловием множество присваиваний вида  $v := expr(B)$ , где  $B \subseteq A$  – подмножество атрибутов, а  $expr(B)$  – некоторая функция над этими атрибутами.

**Определение 2.** Переходом называется тройка вида  $(t, \alpha, \beta)$ , где  $t$  – имя перехода,  $\alpha$  – его предусловие, а  $\beta$  – постусловие.

Семантика переходов аналогична охраняемым командам Дейкстры [12]: если в некотором состоянии  $s$  предусловие перехода  $t$  истинно, то модель может выполнить этот переход и перейти в новое состояние  $s' = t(s)$ , которое отличается от предыдущего значениями тех атрибутов, которым было выполнено присваивание новых значений в постусловии. Переходы детерминированы и выполняются атомарно за конечное время.

**Определение 3.** Формальной моделью называется семерка вида  $M = \langle S, s_0, T, A, D, I, F \rangle$ , где  $S$  – конечное множество состояний,  $s_0 \in S$  – начальное состояние,  $T$  – конечное множество переходов,  $A$  – атрибутов,  $D$  – соответствующие области допустимых значений. Интерпретация атомарных формул предусловий задана функцией  $I: S \times \Pi \rightarrow \{\mathbf{T}, \mathbf{F}\}$ , где  $\Pi$  – множество всех атомарных формул из предусловий переходов и проверяемых свойств,  $\mathbf{T}$  и  $\mathbf{F}$  соответственно обозначают «истина» и «ложь».  $F: (s, v) \rightarrow D(v)$  – функция, вычисляющая значение атрибута  $v$  на состоянии  $s$ .

Такая формальная модель служит удобным математическим аппаратом для описания поведения широкого спектра систем асинхронно-взаимодействующих процессов и их абстракций. Атрибуты могут быть логически разбиты на подмножества в соответствии с их принадлежностью процессам; переходы могут быть параметризованы идентификатором процесса.

**Определение 4.** Истинность формулы  $\phi$  на состоянии  $s$ , записывается  $s \models \phi$ , определяется индуктивно по структуре формулы  $\phi$ :

$$s \models a \quad \equiv \quad I(s, a) = \mathbf{T} \text{ – если атомарный предикат } a \text{ истинен в } s, \text{ иначе } \mathbf{F};$$

$$s \models \neg \phi \quad \equiv \quad s \not\models \phi \text{ – если формула } \phi \text{ не выполняется в } s;$$

$$s \models \phi_1 \wedge \phi_2 \quad \equiv \quad s \models \phi_1 \wedge s \models \phi_2 \text{ – если в } s \text{ выполняется формула } \phi_1 \text{ и } \phi_2;$$

$$s \models \phi_1 \vee \phi_2 \quad \equiv \quad s \models \phi_1 \vee s \models \phi_2 \text{ – если в } s \text{ выполняется формула } \phi_1 \text{ или } \phi_2.$$

Запись  $s \xrightarrow{t} s'$  означает, что предусловие перехода  $t$  выполняется в состоянии  $s$ , т.е.  $s \models \alpha_t$  и выполнение присваиваний постусловия  $\beta_t$  преобразует  $s$  в  $s'$ .

**Определение 5.** Трассой в  $M$  из состояния  $s_i$  в состояние  $s_j$  называется такая последовательность состояний и переходов  $s_i \xrightarrow{t_i} s_{i+1} \xrightarrow{t_{i+1}} s_{i+2} \dots s_j$ , что  $s_k \in S \wedge t_k \in T$  для всех  $k \in i..j$ .

**Определение 6.** Состояние  $s$  достижимо в модели  $M$ , если существует трасса, ведущая из начального состояния  $s_0$  в  $s$ .

Под поведением модели в общем случае понимается множество ее трасс, выходящих из начального состояния.

### Выполнение переходов и граф зависимостей

Для выполнения перехода  $t$  из состояния  $s$  в новое состояние  $s'$  данная работа использует процедуру  $\text{transit}(s, t)$ ; ее подробное описание приведено в [9]. Так, выполнение  $\text{transit}$  построит множество  $R$  атрибутов, значений которых достаточно для вычисления истинности предусловия перехода  $t$  на состоянии  $s$ , и, если переход выполним, новое состояние  $s'$ , множество  $W$  атрибутов, которым осуществлялось присваивание, а также для каждого атрибута  $w \in W$  множество  $V[w]$  атрибутов, которые входят в выражение, формирующее значение  $w$ . Пример ее работы с переходом  $\text{Ta1}$  продемонстрирован на рис. 2.

<p><b>Предусловие перехода Ta1</b> : <math>Cf=Fa \wedge a\_state=1</math>  <b>Постусловие перехода Ta1</b> : <math>Cf:=Fb ; a\_state:=2</math></p>
<p><b>Вход</b> : <math>((Cf=Fa, a\_state=1, b\_state=1), Ta1)</math>  <b>Выход</b> :  <math>(result=T; s'=(Cf=Fb, a\_state=2, b\_state=1);</math>  <math>R=\{Cf, a\_state\}; W=\{Cf, a\_state\}; V[]=\emptyset)</math></p>

Рис. 2. Пример работы процедуры transit для перехода Ta1

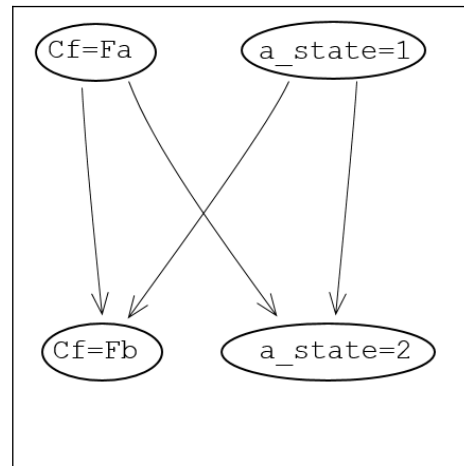


Рис. 3. Подграф для Ta1.

На рис. 3 показан подграф, построенный при выполнении перехода Ta1.

**Определение 7.** Графом зависимости называется размеченный ориентированный ациклический граф, разметки вершин которого представляют собой пары вида «атрибут=значение», а дуги отмечают факт зависимости.

Далее предполагается, что информационная зависимость модели представлена графом, удовлетворяющим определению 7 (далее для краткости D-граф). В основу данного метода положена идея сокращения множества дуг D-графа путем наложения дополнительных условий для их порождения.

#### Алгоритм определения достаточного подмножества информационных связей

Состояния изменяются постусловиями переходов модели, которые в свою очередь, по определению, состоят из множества присваиваний. Очевидно, что, если одинаковое присваивание встречается во всех переходах, то можно игнорировать информационную зависимость такого присваивания от предусловий (исключением может стать тупиковая ситуация). Интуитивно, суть алгоритма такова: для каждого присваивания построить дизъюнкцию предусловий всех переходов, в постусловиях которых встречается это присваивание. Результирующая формула будет достаточным условием для выбранного присваивания, следовательно, любую дополнительную связь можно считать избыточной, и в дальнейшем, при выполнении перехода, ее можно игнорировать без ущерба для точности проверяемых свойств.

**Определение 8.** Достаточным общим условием для выполнения присваивания называется дизъюнкция предусловий всех переходов модели, содержащих данное присваивание.

**Алгоритм 1.** Построение условий для порождения информационных связей, достаточных для сохранения трассовой эквивалентности.

*Вход.* Формальная модель.

*Выход.* Массив SUFFICIENT[] достаточных общих условий для выполнения каждого присваивания.

*Описание.*

Шаг 1. Построить множество различных присваиваний ASSIGNS из постусловий всех переходов модели.

Для каждого присваивания  $assign \in ASSIGNS$  выполнить шаги 2, 3:

Шаг 2. Построить множество  $TA$  переходов, постуловия которых включают присваивание  $assign$ :  $TA := \{ t: t \in T \wedge assign \in \beta_t \}$ .

Шаг 3. Построить формулу, выражающую достаточное общее условие для выполнения присваивания  $SUFFICIENT[assign] := \text{simplify}(\bigvee_{t \in TA} \alpha_t)$ .

В третьем шаге использована функция  $\text{simplify}$ , суть которой заключается в упрощении логических выражений. Чем меньше различных атрибутов останется в упрощенной формуле, тем большим будет сокращение количества дуг в  $D$ -графе.

Далее, для сокращения информационных связей, полученный массив  $SUFFICIENT[]$  должен быть использован согласно следующему алгоритму:

**Алгоритм 2.** Сокращение предусловий.

Не теряя общности, для простоты описания предполагается, что предусловия представляют собой конъюнкцию предикатов.

*Вход.* Формальная модель и массив  $SUFFICIENT[assign]$ .

*Выход.* Массив  $PRE[]$  сокращенных предусловий для каждого перехода.

*Описание.* Для каждого перехода  $t \in T$  выполнить:

Шаг 1. Построить исходную формулу предусловия  $PRE[t] := \alpha_t$ .

Шаг 2. Построить множество атрибутов  $AT$ , синтаксически входящих в  $PRE[t]$ .

Шаг 3. Для каждого атрибута  $a \in AT$ , построить множество предикатов  $PA$  формулы  $PRE[t]$  синтаксически его содержащих и выполнить шаги 4–6.

Шаг 4. Построить новую сокращенную формулу  $newPRE$  предусловия путем удаления всех предикатов  $PA$  из  $PRE[t]$ .

Шаг 5. Построить формулу  $FS$  из таких  $SUFFICIENT[]$ , присваивания которых встречаются в  $t$ :  $FS := \bigvee_{assign \in \beta_t} SUFFICIENT[assign]$ .

Шаг 6. Проверить выполнимость  $FS \rightarrow newPRE$ . Если выполнимо, то обновить формулу предусловия:  $PRE[t] := newPRE$ .

**Определение 9.** Формула  $PRE[t]$ , полученная в результате работы алгоритмов 1 и 2, называется достаточным частным условием для выполнения перехода  $t$ .

Необходимо отметить, что, при осуществлении перехода, алгоритм порождения достижимых состояний для проверки выполнимости этого перехода должен использовать исходную формулу предусловия, а для порождения графа информационных связей – соответствующее достаточное частное условие. Таким образом, множество дуг результирующего  $D$ -графа будет подмножеством дуг исходного графа. Назовем такой граф  $Reduced-1-D$ -графом.

**Теорема 1.**  $Reduced-1-D$ -граф сохраняет трассовую эквивалентность относительно исходного  $D$ -графа.

*Пример.* В результате применения алгоритма 1 была сокращена дуга (см. рис. 4), соединяющая вершины  $a\_state=1$  и  $Cf=fb$ . Этим выражен факт независимости значения потока управления от значения датчика. На рис. 5 представлен соответствующий  $Reduced-1-D$ -граф:

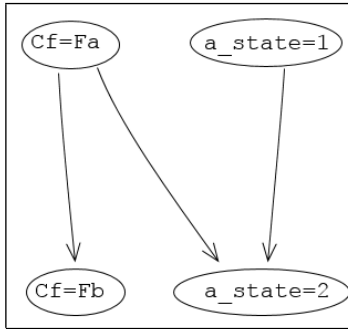


Рис. 4. Сокращенный подграф для перехода  $Ta_1$ .

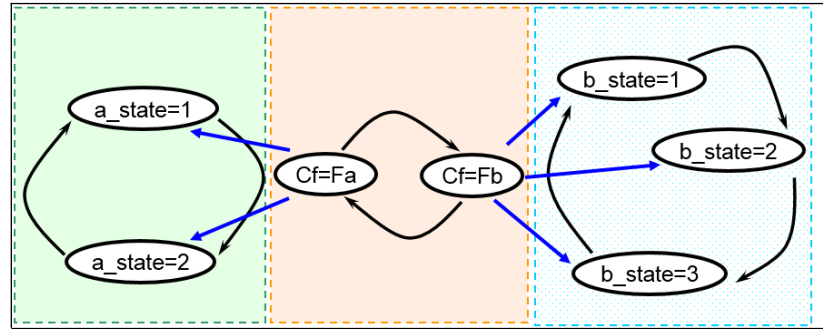


Рис. 5. Reduced-1-D-граф для примера 1.

Заметим, что в Reduced-1-D-графе нет связей, ведущих от значений датчиков к потоку управления; более того, значения датчиков группы  $a$  и  $b$  теперь не зависимы (нет пути на графе, который бы их соединял), а значит, их можно рассматривать как независимые функциональности со всеми вытекающими следствиями (отладка, тестирование и проверка достижимости упрощается – достаточно рассмотреть всего  $2 * (\max(2,3)) = 6$  состояний; слайсинг по одному из датчиков не будет включать другого; возможно оптимизирующее распараллеливание и т.д.). При этом множество возможных трасс не изменилось.

#### Алгоритм определения сокращенного подмножества информационных связей

Следующий алгоритм делает еще большие сокращения D-графа, прибегая к некоторым закруглениям в информационных связях, потенциально приводящим к потере тупиков. Формально, потеря тупиков нарушает трассовую эквивалентность, однако не влияет на доказательство достижимости. Идея этого алгоритма состоит в использовании подмножества состояний, обладающих свойством неполноты [13] для подмножества переходов, содержащих присваивания очередному рассматриваемому атрибуту. Введем следующие определения:

**Определение 10.** Достаточным общим условием для невыполнения присваивания называется дизъюнкция предусловий всех переходов модели, содержащих синтаксически иное присваивание тому же атрибуту, что и в данном присваивании.

**Определение 11.** Сокращенным общим условием для выполнения присваивания называется дизъюнкция достаточного общего условия для его выполнения и отрицания достаточного общего условия для его невыполнения.

**Алгоритм 3.** Построение условий для порождения сокращенного подмножества информационных связей.

*Вход.* Формальная модель.

*Выход.* Массив SUFFICIENT[] сокращенных общих условий для каждого присваивания.

*Описание.*

Шаг 1. Построить множество различных присваиваний ASSIGNS из постусловий всех переходов модели.

Для каждого присваивания  $assign \in ASSIGNS$  выполнить шаги 2–7:

Шаг 2. Построить множество TA переходов, постуловия которых включают присваивание  $assign: TA := \{t : t \in T \wedge assign \in \beta_t\}$ .

Шаг 3. Построить множество присваиваний UNASSIGNS, которые выполняют иное присваивание тому же атрибуту, что и в assign.

Шаг 4. Построить множество TNA переходов на основании UNASSIGNS:  $TNA := \{t : t \in T \wedge unassign \in \beta_t\}$ .

Шаг 5. Построить формулу, выражающую достаточное общее условие для выполнения присваивания:  $SA := \bigvee_{t \in TA} \alpha_t$ .

Шаг 6. Построить формулу, выражающую достаточное общее условие для невыполнения присваивания:  $SNA := \bigvee_{t \in TNA} \alpha_t$ .

Шаг 7. Построить формулу, выражающую сокращенное общее условие для выполнения присваивания:  $SUFFICIENT[assign] := simplify(SA \vee \neg SNA)$ .

Далее требуется построить массив сокращенных предусловий PRE[] используя алгоритм 2.

**Определение 12.** Формула  $PRE[t]$ , полученная в результате работы алгоритмов 3 и 2, называется сокращенным частным условием для выполнения перехода  $t$ .

Аналогично предыдущему случаю, при осуществлении перехода, для его выполнимости алгоритму порождения достижимых состояний следует использовать исходную формулу предусловия, а для построения графа информационных связей (при условии выполнимости) – соответствующую формулу сокращенного частного условия. Назовем такой граф Reduced-2-D-графом.

**Теорема 2.** Множество достижимых состояний, порождаемых Reduced-2-D-графом, включает множество достижимых состояний исходного D-графа.

*Следствие.* Если некоторое состояние недостижимо в Reduced-2-D-графе, то оно недостижимо и в исходном D-графе.

Это делает Reduced-2-D-граф привлекательным для повышения эффективности проверки достижимости – на его основании можно строить достоверные выводы о недостижимости (например, переходов) и отсекал исследуемые ветви поведения.

## Выводы

Предложен новый метод отсечения избыточных, по отношению к проверяемым свойствам, информационных связей в формальных моделях. Факт избыточности устанавливается путем символьного анализа формул предусловий переходов модели. Предложенные алгоритмы не порождают пространство состояний модели; их вычислительная сложность на исследуемых практических примерах линейна относительно количества присваиваний в переходах модели, что позволяет добиться высокой производительности. Результат работы метода – данные об избыточности связей – позволяют не рассматривать некоторые (независимые с точки зрения проверяемых свойств) перестановки действий, таким образом, сократить эффект «комбинаторного взрыва», и как следствие, повысить эффективность анализа свойств модели (в частности, проверки достижимости), отладки, а также тестирования, слайсинга, оптимизаций и т.п.



**Литература**

1. Jhala R., Majumdar R. Software model checking // ACM Comput. Surv. – Vol.41(4). – 2009. – 54 P.
2. Peled D. Partial order reduction: linear and branching temporal logics and process algebras // In proc. of the workshop on Partial order methods in verification. NY, USA: AMS Press, Inc. – 1997. – P. 233–257.
3. Godefroid P. Partial-order methods for the verification of concurrent systems – an approach to the state–explosion problem // LNCS, Springer-Verlag. – 1996. – Vol. 1032. –143 P.
4. Holzmann G. The model checker SPIN // IEEE transactions on software engineering. – 1997. – Vol. 23. – N 5. – P. 279–295.
5. Podgurski A., Clarke A. A. Formal model of program dependencies and its implications for software testing, debugging and maintenance // IEEE Trans. Software Eng. – 1990. –Vol.16(9). –P. 965–979.
6. Cortesi A., Halder R. Dependence Condition Graph for Semantics-based Abstract Program Slicing // In Proc. of the 10th Workshop on Language Descriptions, Tools and Applications. – 2010. – №4. – 9 P.
7. Halder R. Cortesi A., Abstract program slicing on dependence condition graphs Science of Computer Programming Volume 78, Issue 9, 1 September 2013, Pages 1240–1263.
8. Ilan Beer, Shoham Ben-David and oth. Explaining Counterexamples Using Causality // Formal Methods in System Design. – 2012. – №40. – P. 20–40.
9. Колчин А.В. Автоматический метод динамического построения абстракций состояний формальной модели // Кибернетика и системный анализ. – 2010. – № 4. – С. 70–90.
10. Колчин А.В., Четвертак Р.В. Интерактивная система для анализа поведения формальных моделей программных систем // Искусственный интеллект. –2012. – №4. – С. 330–341.
11. Колчин А.В. Метод редукции анализируемого пространства поведения при верификации формальных моделей распределенных программных систем // Искусственный интеллект. –2013. – №4. –С. 113–126.
12. Dijkstra E. Guarded commands, nondeterminacy and formal derivation of programs // Communications of the ACM. – 1975. – Vol.18. – N 8. – P. 453–457.
13. Колчин А.В., Летичевский А.А., Потиеенко С.В. Метод статической проверки полноты и непротиворечивости в формальных моделях распределенных программных систем // Проблемы программирования. – 2014. – № 2–3. – С. 146–150.

**Literatura**

1. Jhala R., Majumdar R. Software model checking // ACM Comput. Surv. – Vol.41(4). – 2009. – 54 P.
2. Peled D. Partial order reduction: linear and branching temporal logics and process algebras // In proc. of the workshop on Partial order methods in verification. NY, USA: AMS Press, Inc. – 1997. – P. 233–257.
3. Godefroid P. Partial-order methods for the verification of concurrent systems – an approach to the state–explosion problem // LNCS, Springer-Verlag. – 1996. – Vol. 1032. –143 P.
4. Holzmann G. The model checker SPIN // IEEE transactions on software engineering. – 1997. – Vol. 23. – N 5. – P. 279–295.
5. Podgurski A., Clarke A. A. Formal model of program dependencies and its implications for software testing, debugging and maintenance // IEEE Trans. Software Eng. – 1990. –Vol.16(9). –P. 965–979.
6. Cortesi A., Halder R. Dependence Condition Graph for Semantics-based Abstract Program Slicing // In Proc. of the 10th Workshop on Language Descriptions, Tools and Applications. – 2010. – №4. – 9 P.
7. Halder R. Cortesi A., Abstract program slicing on dependence condition graphs Science of Computer Programming Volume 78, Issue 9, 1 September 2013, Pages 1240–1263.
8. Ilan Beer, Shoham Ben-David and oth. Explaining Counterexamples Using Causality // Formal Methods in System Design. – 2012. – №40. – P. 20–40.
9. Kolchin A. V. An automatic method for the dynamic construction of abstractions of states of a formal model // Cybernetics and system analysis. – 2010. – № 4. – P. 70–90.
10. Kolchin A. V., Chetvertak R.V. An interactive system for analysis of formal models behavior // Artificial intelligence. –2012. – №4. – P. 330–341.
11. Kolchin A. V. A method for reduction of analyzed behavior space during verification of formal models of distributed software systems // Artificial intelligence. –2013. – №4. – P. 113–126.
12. Dijkstra E. Guarded commands, nondeterminacy and formal derivation of programs // Communications of the ACM. – 1975. – Vol.18. – N 8. – P. 453–457.
13. A. Kolchin, A. Letichevsky, S. Potiyenko. Static method of consistency and completeness checking in formal model of distributed software systems // Problems in programming. –2014. –N 2–3. –P.146–150.

**RESUME****A. Kolchin, A.Letychevskyy, S.Potiyenko****A static method for elimination of redundant dependencies in preconditions of transitions of formal models of transition systems**

A new method of elimination of redundant (with respect to properties under checking) information dependencies in formal models is proposed. In order to prove the redundancy fact, we use symbolic analysis of precondition formulas of model transitions. Proposed algorithms do not generate model state space, and their computational complexity is linear in number of assignments in model transitions on investigated practical examples, which reflects in high performance. The result of the method is data about redundancy of dependencies. It allows avoiding of considering of some (independent) shuffling of actions, and thus, to reduce combinatorial explosion effect, and, consequently, to improve efficiency of model properties analysis (checking reachability for example), debugging, testing, slicing, optimizations and so on.

**А.В. Колчин, А.А. Летичевский, С.В. Потиеенко****Статический метод устранения избыточных информационных связей в предусловиях переходов формальных моделей транзиторных систем**

Предложен новый метод отсекаания избыточных по отношению к проверяемым свойствам информационных связей в формальных моделях. Факт избыточности устанавливается путем символического анализа формул предусловий переходов модели. Предложенные алгоритмы не порождают пространство состояний модели; их вычислительная сложность на исследуемых практических примерах линейна относительно количества присваиваний в переходах модели, что позволяет добиться высокой производительности. Результат работы метода – данные об избыточности связей – позволяют не рассматривать некоторые (независимые) перестановки действий, таким образом, сократить эффект «комбинаторного взрыва», и, как следствие, повысить эффективность анализа свойств модели (в частности, проверки достижимости), отладки, а также тестирования, слайсинга, оптимизаций и т.п.

*Поступила в редакцию 01.09.2015*