УДК 004.415.28 + 004.415.52

С.В. Потиенко

Организация базы знаний о переходах системы с атрибутами перечислимых типов

Предложен метод построения базы знаний о переходах моделируемой системы для решения проблемы определения допустимости переходов в символьном моделировании.

A method for building of a knowledge base about transitions of a system under modeling is suggested for solving a problem of detection acceptable transitions in symbolic model checking.

Запропоновано метод побудови бази знань про переходи системи, що моделюється для вирішення проблеми визначення допустимості переходів у символьному моделюванні.

Введение. Рассмотрим модели многокомпонентных систем, описанные в языке базовых протоколов с использованием только перечислимых типов. Опыт верификации множества индустриальных проектов показал, что большинство из них обходится перечислимыми типами, а остальные ограничиваются добавлением примитивных выражений над целочисленными переменными для организации таймеров и счетчиков. Они тоже могут быть приведены к перечислимым типам с помощью несложных абстракций (однако отметим, что это не точные абстракции, которые могут порождать ложные поведения). Ограничение по типам исключает из рассмотрения смешанные формулы, что позволяет упростить алгоритмы доказательства выполнимости, уменьшить алгоритмическую сложность и увеличить производительность реализации за счет дополнительных оптимизаций. Под реализацией подразумевается система верификации требований *VRS*, использующая методику инсерционного моделирования [1–5].

Модель системы базовых протоколов с атрибутами перечислимых типов

Модель, описанная в языке базовых протоколов, состоит из среды и погруженных в нее агентов, работающих параллельно, асинхронно и взаимодействующих между собой посредством чтения и изменения атрибутов. Состояние среды состоит из набора значений ее атрибутов и значений атрибутов всех действующих агентов, а переходы задаются базовыми протоколами. В символьном моделировании под состоянием среды подразумевается множество состояний, заданное логической формулой. Атрибуты, для которых формула не задает одного конкретного значения, называются символьными.

Базовый протокол представляет собой формулу темпоральной логики (тройку Хоара) $\forall x\,(\alpha\,(r, x) \rightarrow <P(r, x)> \beta(r, x)$ и описывает тот факт, что если в какой-то момент времени истинно предусловие α, то выполняется процесс P и состояние среды изменяется согласно постусловию β (здесь r – список атрибутных выражений, которые будут рассмотрены ниже). Пред- и постусловия есть формулами логики первого порядка над атрибутами и переменными перечислимых типов. Постусловие так же может содержать операторы присваивания.

Считается, что каждый базовый протокол выполняется каким-либо одним конкретным агентом, называемым ключевым для этого протокола. Для организации потока управления ключевого агента в пред- и постусловиях введен специальный атрибут *state*(k) (здесь k – имя ключевого агента, которое может быть константой или переменной из списка x) со следующими ограничениями:

– выделен синтаксически;

– обязательно встречается в пред- и постусловии с одним и тем же аргументом и не более одного раза;

– всегда имеет конкретное значение.

В системе *VRS* агенты разделены по типам. Агентным типом базового протокола называется тип его ключевого агента.

Для осуществления перехода системы из состояния s в состояние s' необходимо выбрать базовый протокол b_i. Так как рассматривается асинхронная система, а базовый протокол выполняется одним агентом, то в первую очередь произвольным образом выбирается агент k_j, будучи ключевым для применяемого протокола.

Далее необходимо выбрать применимый базовый протокол соответствующего агентного типа. Условием применимости есть выполнимость формулы:

$$\exists y\,(s\,(r, y)) \wedge \exists x((x_0 = k_j) \wedge \alpha_i(r, x)).$$

Здесь x_0 – переменная из списка x, обозначающая имя ключевого агента в протоколе b_i. Специальный атрибут *state* выбранного агента k_j имеет конкретное значение как в $s(r,y)$, так и в $\alpha_i(r,x)$. Очевидно, что множество базовых протоколов каждого типа агентов можно разбить на подмножества по значениям атрибута *state*, таким образом сократив количество проверок применимости до подмножества протоколов со значением атрибута $state(x_0)$ в предусловии, равным значению $state(k_j)$ в состоянии среды.

В последнее время пользователи системы *VRS* столкнулись с проблемой большого количества базовых протоколов, в частности, полученных трансляцией диаграмм *UCM*. Такие модели сокращают с помощью введения дополнительных атрибутов управления потоком и их комбинирования в пред- и постусловиях. Этот метод позволяет легко организовать синхронизацию различных агентов, обработку прерываний и т.д., но при этом не представляется возможным использование специального атрибута *state* в силу действующих ограничений. Соответственно, количество проверок условия применимости становится максимальным, что очень плохо влияет на производительность.

База знаний о базовых протоколах

Анализ пользовательских моделей показал, что атрибуты управления потоком практически всегда имеют конкретные значения в состоянии среды. Такая же ситуация наблюдается и в предусловиях, но если рассмотреть каждый из таких атрибутов по отдельности, то он встречается не везде. Так же предусловия не ограничены использованием лишь одного атрибута.

Для таких моделей можно задать разбиение базовых протоколов, основываясь на конкретных значениях атрибутов управления потоком.

Как уже упоминалось, формулы предусловий, а также формулы состояний среды, полученные в результате применения базовых протоколов, есть формулами логики первого порядка вида $\exists x\,(F\,(r, x))$, где x – список переменных перечислимых типов, r – список атрибутных выражений. Атрибутным выражением является имя r_i атрибута среды перечислимого типа либо функциональное выражение $r_j(z_1, \ldots, z_n)$, где r_j – атрибут среды функционального типа или любой атрибут какого-либо агента, $z_1, \ldots, z_n$ – аргументы: константы, переменные или атрибутные выражения. Любой атрибут агента представляется как функциональное выражение, первым аргументом которого является имя агента, а остальные аргументы, если они есть, сдвигаются на одну позицию вправо. Например, в языке *VRS* атрибуты агентов записываются как $T\ a.r_j$ или $T\ a.r_k(z_1, \ldots, z_n)$ (здесь T – имя типа агента, a – имя агента) и будут представлены как $r_j(a)$ и $r_k(a, z_1, \ldots, z_n)$ соответственно.

Будем опираться на то, что все формулы приводятся к некоторому нормальному виду, где известны области допустимых значений (ОДЗ) каждого атрибутного выражения и связанной переменной, а аргументы функциональных выражений по возможности вычисляются. Поэтому любую формулу можно рассматривать как двойку вида:

$$\begin{aligned} &< \exists x(F(r,x)), V(r,x) >, \\ &V(r,x) = (r_0 \in O_0, \ldots, r_k(z_{k1}, \ldots, z_{kn}) \in \\ &\qquad \in O_k, \ldots, x_0 \in O'_0, \ldots), \\ &O_i = \{v_{i0}, \ldots, v_{in}\},\ O'_i = \{v'_{i0}, \ldots, v'_{in}\}. \end{aligned} \qquad (*)$$

Здесь F – формула, V – список таких пар: атрибутное выражение $r_i \in r$ или переменная $x_i \in x$ и его (ее) ОДЗ O_i или O'_i. Остальные атрибуты, не входящие в F, могут принимать любые значения из своих перечислимых типов, поэтому нет смысла включать их в список V.

Для построения базы знаний о базовых протоколах рассмотрим их предусловия как тройки:

$$p = < \exists x(P\,(r, x)),\ V(r,x),\ H(h, x_0) >,$$

где P – формула предусловия, V – список пар как в (*), x_0 – переменная из списка x, обозначающая имя ключевого агента данного базового протокола, h – список, содержащий атрибутные выражения из r нулевой арности, а также функциональные выражения из r, все аргументы которых являются либо константами (вычислены в процессе нормализации), либо переменной x_0. Первым в списке h стоит

специальный атрибут $state(x_0)$. H – список таких пар из V, которые содержат ОДЗ только для h. Если базовый протокол имеет константное имя ключевого агента, то переменная x_0 в аргументах атрибутных выражений из h отсутствует. Фактически $h \subset r$, $H \subset V$.

Представим базу знаний базовых протоколов в виде дерева, узлами которого являются следующие тройки:

$$n =< r_i, \overline{B}, \overline{N} >,$$

где r_i – атрибутное выражение из списка h некоторого предусловия; $\overline{B} = (B_1, \ldots, B_n)$, $B_i = \{b_{i1}, b_{i2} \ldots\}$ – вектор множеств имен базовых протоколов; $\overline{N} = (N_1, \ldots, N_n)$, $N_i = (n_{i1}, n_{i2} \ldots)$ – вектор списков следующих узлов дерева, данный узел для них будем называть родительским.

Вектора $\overline{B}$ и $\overline{N}$ имеют размер перечислимого типа, значения которого принимает атрибутное выражение r_i, и их элементы взаимно однозначно соответствуют значениям этого типа. При этом множества b_{ij} и списки n_{ij} могут быть пустыми. Обозначим $T(n) = (n_0, n_1, \ldots, n)$ путь от верхнего узла n_0 до узла n, $T_r(n) = (r_0, r_1, \ldots, r_i)$ – список атрибутных выражений, каждое из которых – первый элемент тройки каждого узла на пути $T(n)$.

Построим дерево таким образом, чтобы оно обладало следующими свойствами:

• Для каждого предусловия $p =< \exists x(P(r,x)), V(r,x), H(h,x_0) >$ существует путь $T(n)$, содержащий все атрибутные выражения из h, и $T_r(n) = h$.

• Каждый из элементов b_{ij} и n_{ij} векторов $\overline{B}$ и $\overline{N}$ любого узла $n =< r_i, \overline{B}, \overline{N} >$ соответствует одному значению v_j перечислимого типа атрибутного выражения r_i следующим образом:

– если есть такое предусловие p, что $r_i \in h$ и $v_j \in O_i$ этого предусловия, а также $T_r(n) = h$, то множество b_{ij} содержит имя соответствующего предусловию p базового протокола;

– если есть такое предусловие p, что $r_i \in h$ и $v_j \in O_i$ этого предусловия, $T_r(n) \neq h$, но $T_r(n)$ включается в список h с первой позиции, то список n_{ij} содержит узел, соответствующий следующему за r_i атрибутному выражению в списке h.

Оценим размер построенного дерева. Из представления узлов (в виде троек) и первого свойства следует, что минимальная оценка количества узлов в дереве равна количеству различных атрибутных выражений из списков h всех предусловий. В худшем случае пути $T(n)$ для всех предусловий не будут пересекаться, поэтому максимальная оценка равна сумме размеров списков h всех предусловий.

Заметим, что описанные свойства не задают однозначного построения дерева. Так как дерево строится статически, поэтому не будем детально останавливаться на алгоритме построения и его эффективности. Рассмотрим поиск по базе знаний.

На каждом шаге моделирования имеется нормализованная формула $< \exists x(F(r,x)), V(r,x) >$ состояния среды. Обозначим множество протоколов, являющихся кандидатами на применение, как Q. Вначале это множество пустое. Перед поиском применимых протоколов выбирается ключевой агент k_j, как уже сказано. Далее, начиная с верхнего узла дерева базы знаний, выполняются следующие действия:

• Обозначим текущий узел $n =< r_i, \overline{B}, \overline{N} >$. Если в аргументах атрибутного выражения r_i встречается переменная, обозначающая имя ключевого агента, вместо нее подставляется выбранное ранее имя k_j. Обозначим новое атрибутное выражение как r'_i.

• Для r'_i найдем ОДЗ O_i в списке V. Если такая пара отсутствует, то будем считать, что O_i содержит все значения соответствующего перечислимого типа.

• Для каждого значения $v_j \in O_i$:

– добавим все протоколы из множества b_{ij}, являющегося элементом списка $\overline{B}$, который соответствует значению v_j, в множество кандидатов на применение Q;

– для каждого узла из списка n_{ij}, являющегося элементом списка $\overline{N}$, который соответствует значению v_j, продолжим выполнять алгоритм, начиная с шага 1.

На шаге 3*b* возникает рекурсия, но так как дерево не имеет циклов, алгоритм закончит свою работу после обхода всех подходящих узлов.

После этого для каждого базового протокола из множества кандидатов Q необходимо проверить условие его применимости.

Очевидно, что данный алгоритм поиска по базе знаний имеет линейную сложность. Эксперименты над пользовательскими проектами показали, что время такого поиска ничтожно мало, что подтвердило неактуальность исследования эффективности построения базы знаний.

Пример. В качестве примера построим базу знаний о базовых протоколах с предусловиями.

Базовый протокол	Предусловие
bp1	∀x state(x)=idle; gcf = = gcf0 & cf(x,e1) = cf0
bp2	∀x state(x)=idle; gcf = = gcf0 & cf(x,e2) = cf0
bp3	∀x state(x)=idle; gcf = = gcf1 & cf(x,e1) = cf1
bp4	∀x state(x)=idle; gcf = = gcf0 & cf(x,e1) = cf1
bp5	∀(x,y) state(x)=idle; cf(x,y) = cf0
Типы атрибутов и переменных:	
Gcf ∈ {gcf0, gcf1} cf(x,y) ∈ {cf0, cf1}, x ∈ A (множество агентов), y ∈ {e1,e2,e3}	

База знаний будет представлена в виде проиллюстрированного дерева:

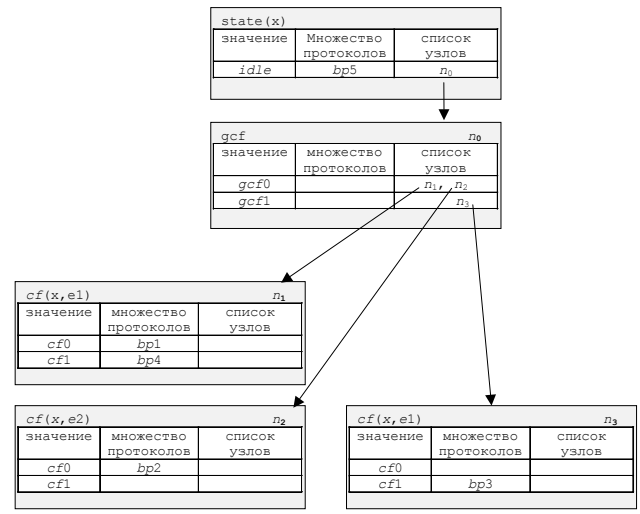

Рис.

На примере видно, что все базовые протоколы имеют одно и то же значение *idle* специального атрибута *state*(x). Без такой базы знаний на каждом шаге моделирования осуществлялась бы проверка условия применимости всех протоколов. На рисунке видно, что при наличии конкретных значений у попавших в базу атрибутных выражений каждый путь дерева ведет лишь к одному протоколу, и только *bp*5 должен проверяться всегда.

Заключение. База знаний о базовых протоколах позволила практически полностью избежать проверок условия применимости для неприменимых протоколов в пользовательских моделях, использующих множество атрибутов управления потоком вместо специального атрибута *state*. Это сократило количество проверок в среднем на 95%, а общее время работы процедуры применения базового протокола – на 70%. Результаты экспериментов с некоторыми *UCM*-моделями приведены в таблице.

Модель	Кол-во базовых протоколов (и типов агентов)	Кол-во проверок условия применимости			
		Успешных	Общее с базой знаний	Общее без базы знаний	Сокращение проверок
#1 (*co*)	55 (2)	7 817	7 817	204 160	96%
#2 (*ps*)	215 (1)	51 944	51 944	1 900 815	97%
#3 (*ci*)	212 (1)	10 644	11 433	141 192	92%
#4 (*em*)	188 (1)	60 494	60 602	1 083 068	94%

Время поиска по базе знаний в таблице не отражено, так как оно ничтожно мало в сравнении с проверками применимости базовых протоколов.

Работа выполнялась при поддержке ДФФД по проекту Ф40.1/004.

1. *Спецификация* систем с помощью базовых протоколов / А.А. Летичевский, Ю.В. Капитонова, В.А. Волков и др. // Кибернетика и системный анализ. – 2005. – № 4. – С. 3–21.
2. *Amir Pnueli*, *Ofer Strichman*. Reduced Functional Consistency of Uninterpreted Functions // Elect. Notes in Theor. Comp. Sci. (ENTCS). – 2006. – **144**. – Issue 2. – P. 53–65.
3. *Insertion* modeling in distributed system design / A.A. Letichevsky, J.V. Kapitonova J., V.P. Kotlyarov et al. // Проблеми програмування. – 2008. – № 4. – С. 13–38.
4. *Потиенко С.В.* Методы прямого и обратного символьного моделирования систем, заданных базовыми протоколами // Там же. – С. 39–45.
5. *Свойства* предикатного трансформера системы *VRS* / А.А. Летичевский, А.Б. Годлевский, А.А. Летичевский (мл.) и др. / Кибернетика и системный анализ. – 2010. – № 4. – С. 3–16.

Тел. для справок: +38 044 200-8423 (Киев)
E-mail: *stepan.mail@gmail.com*