УДК 519.686.2

С.В. Потиенко

МЕТОДЫ ПРЯМОГО И ОБРАТНОГО СИМВОЛЬНОГО МОДЕЛИРОВАНИЯ СИСТЕМ, ЗАДАННЫХ БАЗОВЫМИ ПРОТОКОЛАМИ

В работе рассмотрены методы прямого и обратного символьного моделирования, используемые для решения задач верификации систем базовых протоколов. Помимо алгоритмов поиска в пространстве состояний системы, детально проработанных в области проверки на модели (model checking), основная задача символьного моделирования заключается в осуществлении перехода от одного класса состояний к следующему. Такая трансформация производится с помощью предикатных трансформеров. В данной работе построены алгоритмы прямого и обратного предикатных трансформеров для числовых, символьных и списочных структур данных. Трансформеры рассматриваются как функции, которые выводят из заданной формулы новую, определяющую класс состояний системы после перехода, совершенного под действием заданного базового протокола.

Вступление

Подходы к решению задачи верификации программных систем можно разделить на два типа – проверка на модели (model checking) и доказательство теорем (theorem proving). Символьное моделирование – один из методов комбинации проверки на модели с доказательством теорем. Во время проверки на модели осуществляется поиск в пространстве состояний, генерируемых поведением модели. В символьном моделировании роль состояний играют формулы, которые покрывают классы состояний. Во время поиска в таком пространстве классов состояний используются методы доказательств. Для простоты, классы состояний, заданные формулами, далее будем называть состояниями. Переход от заданного состояния к следующему осуществляется с помощью прямого предикатного трансформера, определяемого как функция вывода формулы – нового состояния модели. Для решения задачи проверки достижимости заданного состояния, как один из методов применяется обратное символьное моделирование, в котором переход от заданного состояния к предыдущему осуществляется с помощью обратного предикатного трансформера.

В данной работе рассматриваются методы символьного моделирования над системами, заданными набором базовых протоколов. Общая методика верификации систем базовых протоколов описана в [1]. Представление требований в форме базовых протоколов базируется на теории взаимодействия агентов и сред [2, 3]. Алгоритмы поиска в пространстве состояний описаны в [4–6]. Здесь раскрываются алгоритмы работы прямого и обратного предикатных трансформеров для систем с числовыми, символьными и списочными структурами данных.

Краткое описание языка базовых протоколов

Базовый протокол представляет собой выражение вида $\forall x(\alpha \rightarrow< u > \beta)$, где x – список (типизированных) параметров, α и β – формулы базового логического языка, u – процесс протокола (конечное поведение композиции нескольких агентов и среды, обычно задается с помощью MSC диаграмм). Формула α называется предусловием, а формула β – постусловием базового протокола. Сам базовый протокол может рассматриваться как формула темпоральной логики, выражающая тот факт, что, если состояние системы имеет разметку, удовлетворяющую условию α, то процесс u может быть инициирован, и после его завершения состояние системы будет удовлетворять условию β.

В качестве базового языка используется язык многосортного исчисления предикатов первого порядка. Формула базового языка может содержать переменные и константы следующих простых типов данных: числовые (целый и действительный) и символьные (булевский, перечислимый и произвольный символьный). Так же допустимы массивы элементов простых типов с целыми или перечислимыми индексами, функциональные типы (функции от аргументов простых типов, возвращающие значение простого типа), списки элементов простых типов. В роли переменных, меняющих свои значения в процессе функционирования системы, выступают атрибуты и атрибутные выражения. Атрибутными выражениями являются операторы доступа к элементу массива по индексам ($a[i,j]$), функции доступа к спискам (*get_from_head*(l), *get_from_tail*(l), *empty*(l)), выражения атрибутов функциональных типов ($a(x,y,z)$).

Предусловие базового протокола содержит формулу базового языка, в постусловии используются присваивания, операторы обновления списков, а так же формула базового языка. Левыми частями присваиваний могут выступать атрибуты простых типов и атрибутные выражения кроме функций доступа к спискам. Присваивания и операторы обновления списков рассматриваются как равенства, связывающие старые и новые значения атрибутов простых типов и атрибутных выражений.

Прямой предикатный трансформер

На каждом шаге моделирования известно текущее состояние среды (на первом шаге известно начальное состояние). Применимые в этом состоянии протоколы модифицируют состояние согласно своим постусловиям. Каждый из одновременно применимых протоколов создает новую ветвь в дереве поведения системы.

Пред- и постусловие каждого базового протокола представляется в таком виде:

- предусловие: $A(r,l,s,z)$;
- постусловие: $B(r,l,s,z) = (r := t_0(r,l,s,z)) \wedge$ $\wedge\ U(l,r,s,z) \wedge C(r,l,s)$.

Здесь:

- l – вектор списковых атрибутов;
- r,s,z – вектора атрибутов и атрибутных выражений числовых и символьных типов;
- $A(r,l,s,z)$ и $C(r,l,s)$ – формулы базового языка;
- $U(l,r,s,z)$ – конъюнкция операторов обновления списков l;
- $r := t_0(r,l,s,z)$ – конъюнкция присваиваний новых значений атрибутам r: $(r_1 := t_1(r,l,s,z)) \wedge (r_2 := t_2(r,l,s,z)) \wedge \ldots$; $r := t(r,s,z)$ – присваивания после подстановки значений операторов доступа к спискам (*get_from_head*(l), *get_from_tail*(l));
- z – вектор переменных, присутствующих в операторах присваивания и обновления списков, но отсутствующих в формуле $C(r,l,s)$ постусловия.

Таким образом, r – это атрибутные выражения, значения которых меняются присваиваниями, а значения s могут измениться недетерминированным образом, поскольку входят в условие C. Значения выражений из z не меняются.

Базовый протокол применим на состоянии E, если формула $E \wedge A(r,l,s,z)$ не ложна. Применимый базовый протокол осуществляет переход:

$$E \rightarrow E'.$$

Здесь E и E' – формулы, определяющие состояния среды. Они представляются в таком виде:

- $E = F(r,s,z) \wedge L(r,l,s,z)$,
- $E' = F'(r,s,z) \wedge L'(r,l,s,z)$,

где

- $F(r,s,z)$, $F'(r,s,z)$ – формулы базового языка;
- $L(r,l,s,z)$, $L'(r,l,s,z)$ – списковые равенства вида:

$(l_1 = list(head_1(r,s,z), \ldots, tail_1(r,s,z))) \wedge$ $\wedge\ (l_2 = list(head_2(r,s,z), \ldots, tail_2(r,s,z))) \wedge \ldots$,

где

- $head_i(r,s,z)$ и $tail_i(r,s,z)$ – последовательности выражений, могут быть пустыми;
- … – абстрактная (неизвестная) часть списка; она отсутствует в списках конкретной длины.

Во время прямого моделирования известно состояние E, с помощью прямого предикатного трансформера необходимо найти состояние E':

$$E' = pt(\ E \wedge A(r,l,s,z),\ B(r,l,s,z)\ ).$$

Прямой предикатный трансформер определяется с учетом следующего предположения: все атрибуты и атрибутные выражения, встречающиеся на верхнем уровне в формуле $C(r,l,s)$ постусловия, и только они могут изменить свои значения после применения базового протокола. Считается, что атрибут находится на верхнем уровне, если он не расположен внутри какого-либо атрибутного выражения. Например, если атрибут встречается как индекс массива или параметр атрибута функционального типа, это не считается верхним уровнем.

Прямой предикатный трансформер применяется к формуле

$$F(r,l,s,z) \wedge L(l,r,s,z) \wedge A(r,l,s,z)$$

и выводит новую формулу E':

$$E' = E_1 \vee E_2 \vee \ldots.$$

$$E_i = \exists(u,v)\,(F(u,v,\xi_i) \wedge A(u,l,v,\xi_i) \wedge$$
$$\wedge\, T(r,u,v,\xi_i) \wedge L'(l,u,v,\xi_i) \wedge P_i(u,v,r,s)) \wedge$$
$$\wedge\, C(r,l,s).$$

$$T(r,u,v,\xi_i) =$$
$$= (r_1 = t_1(u,v,\xi_i)) \wedge (r_2 = t_2(u,v,\xi_i)) \wedge \ldots.$$

$$L'(l,u,v,\xi_i) =$$
$$= (l_1 = list(head_1(u,v,\xi_i),\ldots,tail_1(u,v,\xi_i))) \wedge$$
$$\wedge\, (l_2 = list(head_2(u,v,\xi_i),\ldots,tail_2(u,v,\xi_i))) \wedge$$
$$\wedge \ldots.$$

Здесь u, v – вектора связанных переменных, введенных для обозначения старых значений атрибутов r, s. $L'(l,u,v,\xi_i)$ содержит обновленные списки после применения операторов $U(l,r,s,\xi_i)$. Если атрибут функционального типа встречается среди атрибутных выражений вектора r или s, а также в векторе z, должны быть рассмотрены все комбинации возможных отождествлений его аргументов. Формула $P_i(u,v,r,s)$ определяет одну из таких возможностей. Вектора ξ_i получаются заменой выражений из z с отождествленными аргументами переменными из списков u или v.

Например, пусть выражение $x(i)$ встречается среди левых частей присваивания, а $x(j)$ встречается в z, где x – атрибут функционального типа. Тогда рассмотрим два случая: $i = j$ и $i \neq j$. Для случая $i = j$, выражение $x(j)$ заменяется на переменную из списка u, соответствующую выражению $x(i)$.

Обратный предикатный трансформер

Обратное моделирование позволяет из заданного состояния среды построить трассу в начальное состояние. Для этого используются алгоритмы прямого моделирования с обратным предикатным трансформером, который определяется далее.

Во время обратного моделирования известно состояние E', с помощью обратного предикатного трансформера необходимо найти состояние E:

$$E = pt^{-1}(\,E', A(r,l,s,z), B(r,l,s,z)\,).$$

Определяющим свойством обратного трансформера является следующее условие:

$$pt(\,pt^{-1}(E', A(r,l,s,z), B(r,l,s,z)) \wedge$$
$$\wedge\, A(r,l,s,z), B(r,l,s,z)) \rightarrow E'.$$

Рассмотрим два случая.

1. $C(r,l,s) = 1$.

В данном случае постусловие содержит только операторы присваивания и обновления списков. Рассмотрим оба типа операторов.

Операторы присваивания изменяют формулу в состоянии среды как описано в прямом предикатном трансформере. Положим:

$$F(r,s,z) = F'(t(r,s,z),s,z) \wedge A(r,l,s,z) \wedge P(r,s,z)$$

или

$$pt^{-1}(\,F'(r,s,z)\,,A(r,l,s,z), B(r,l,s,z)\,) =$$
$$= F'(t(r,s,z),s,z) \wedge A(r,l,s,z) \wedge P(r,s,z).$$

Как упоминалось выше, r, s, z – вектора атрибутов не списковых типов, т.е. простых типов – числовые (целые и действительные), перечислимые, символьные, а так же функциональных типов. Массивы могут рассматриваться как функциональные типы с одним целочисленным или перечислимым параметром, ограниченным размерностью массива. Если один атрибут функционального типа встречается в постусловии более одного раза, должны быть рассмотрены все комбинации возможных отождествлений его аргументов. Формула $P(r,s,z)$ определяет одну из таких возможностей.

Например, пусть выражения $x(i)$ и $x(j)$ встречаются $B(r,l,s,z)$, где x – атрибут функционального типа. Тогда рассмотрим два случая: $i = j$ и $i \neq j$.

Если формула $F'(r,l,s,z)$ ложна, то данный базовый протокол не мог быть применен и соответствующая ветвь поведения не рассматривается.

Докажем определяющее свойство такого обратного трансформера:

$$pt(\ pt^{-1}(F'(r,s,z), A(r,l,s,z), B(r,l,s,z)) \wedge$$
$$\wedge A(r,l,s,z), B(r,l,s,z)\ ) =$$
$$= pt(\ F(r,s,z) \wedge A(r,l,s,z), B(r,l,s,z)\ ) =$$
$$= \exists u\ (\ F(u,s,\xi_1) \wedge A(u,l,s,\xi_1) \wedge (r = t(u,s,\xi_1)) \vee$$
$$\vee F(u,s,\xi_2) \wedge A(u,l,s,\xi_2) \wedge$$
$$\wedge (r = t(u,s,\xi_2)) \vee ...\ ) =$$
$$= \exists u\ (\ F'(t(u,s,\xi_1),s,\xi_1) \wedge A(u,l,s,\xi_1) \wedge$$
$$\wedge P(u,s,\xi_1) \wedge (r = t(u,s,\xi_1)) \vee ...\ ) =$$
$$= F'(r,s,\xi_1) \wedge \exists u\ (\ A(u,l,s,\xi_1) \wedge P(u,s,\xi_1)\ ) \vee$$
$$\vee ... \rightarrow F'(r,s,z).$$

Операторы обновления списков $U(r,l,s,z)$ изменяют списковые равенства в состоянии среды:

$$pt(L(r,l,s,z), U(r,l,s,z)) = L'(r,l,s,z),$$

$$pt^{-1}(L'(r,l,s,z), A(r,l,s,z), U(r,l,s,z)) = L(r,l,s,z).$$

$U(r,l,s,z)$ содержит операторы add_to_tail(l, f(r,s,z)), add_to_head(l, f(r,s,z)), remove_from_tail(l, f(r,s,z)), remove_from_head(l, f(r,s,z)).

$L(r,l,s,z)$ содержит равенства вида:

$$l = list(h_1(r,s,z), head(r,s,z), \ldots, tail(r,s,z), q_n(r,s,z)).$$

Здесь

- $h_1(r,s,z)$, $head(r,s,z)$ – рекурсивное представление непустой последовательности; $h_1(r,s,z)$ – арифметическое выражение или символьная константа;
- $tail(r,s,z)$, $q_n(r,s,z)$ – рекурсивное представление непустой последовательности $tail(r,s,z)$; $q_n(r,s,z)$ – арифметическое выражение или символьная константа.

Если $head(r,s,z)$ или $tail(r,s,z)$ пусты, то $l = list(\ldots, tail(r,s,z), q_n(r,s,z))$ или $l = list(h_1(r,s,z), head(r,s,z), \ldots)$ соответственно. Полностью абстрактный список содержит только неизвестную часть: $l = list(\ldots)$.

Восстановление списков и генерация списковых равенств $L(l,r,s,z)$ производится по правилам, описанным далее.

Для оператора $add\_to\_tail(l,f(r,s,z))$:

- если $L'(l,r,s,z)$ содержит равенство $l = list(head(r,s,z), \ldots, tail(r,s,z), q_n(r,s,z))$ → применяем remove_from_tail(l) → тогда $L(l,r,s,z)$ содержит равенство $l = list(head(t(r,s,z),s,z), \ldots, tail(t(r,s,z),s,z))$; в формулу F необходимо добавить: $\exists m(\ (get\_from\_tail(m) = f(r,s,z)) \wedge (l = list(head(r,s,z), \ldots, tail(r,s,z), q_n(r,s,z))\ ) \rightarrow f(r,s,z) = q_n(r,s,z)$;
- если $L'(l,r,s,z)$ содержит равенство $l = list(head(r,s,z), \ldots)$ → неизвестная часть остается неизвестной → тогда $L(l,r,s,z)$ содержит то же равенство $l = list(head(t(r,s,z),s,z), \ldots)$.

Для оператора $add\_to\_head(l,f(r,s,z))$:

- если $L'(l,r,s,z)$ содержит равенство $l = list(h_1(r,s,z), head(r,s,z), \ldots, tail(r,s,z))$ → применяем $remove\_from\_head(l)$ → тогда $L(l,r,s,z)$ содержит равенство $l = list(head(t(r,s,z),s,z), \ldots, tail(t(r,s,z),s,z))$; в формулу F необходимо добавить равенство $f(r,s,z) = h_1(r,s,z)$;
- если $L'(l,r,s,z)$ содержит равенство $l = list(\ldots, tail(r,s,z))$ → неизвестная часть остается неизвестной → тогда $L(l,r,s,z)$ содержит то же равенство $l = list(\ldots, tail(t(r,s,z),s,z))$.

Для оператора $remove\_from\_tail(l)$:

- пусть $L'(l,r,s,z)$ содержит равенство $l = list(head(r,s,z), \ldots, tail(r,s,z))$ → вводим новую переменную v и применяем $add\_to\_tail(l,v)$ → тогда $L(l,r,s,z)$ содержит равенство $\exists v\ (\ l = list(head(t(r,s,z),s,z), \ldots, \ldots, tail(t(r,s,z),s,z), v)\ )$.

Для оператора $remove\_from\_head(l)$:

- пусть $L'(l,r,s,z)$ содержит равенство $l = list(head(r,s,z), \ldots, tail(r,s,z))$ →

вводим новую переменную v и применяем *add_to_head*(l,v) →
тогда $L(l,r,s,z)$ содержит равенство
$\exists v\ (\ l = list(v, head(t(r,s,z),s,z), \ldots,$
$\ldots, tail(t(r,s,z),s,z))\ )$.

У списков конкретной длины отсутствует неизвестная часть …, правила обновления применяются те же. Необходимо лишь отдельно рассмотреть случай, когда состояние E' содержит пустой список $l = list()$, а постусловие базового протокола оператор *add_to_tail*(l) или *add_to_head*(l). Тогда этот протокол не мог быть применен и соответствующая ветвь поведения не рассматривается.

$$pt^{-1}(F'(r,s,z) \wedge L'(l,r,s,z), A(r,l,s,z),$$
$$B(r,l,s,z)) =$$
$$= F'(t(r,s,z),s,z) \wedge L(l,t(r,s,z),s,z) \wedge$$
$$\wedge M(r,s,z) \wedge A(r,l,s,z) \wedge P(r,s,z).$$

Здесь $M(r,s,z)$ – конъюнкция равенств, добавленных во время восстановления списков для операторов *add_to_tail* и *add_to_head*.

Докажем определяющее свойство обратного трансформера с учетом обновления списков:

$$pt(\ pt^{-1}(F'(r,s,z) \wedge L'(l,r,s,z),\ A(r,l,s,z),$$
$$B(r,l,s,z)) \wedge A(r,l,s,z),\ B(r,l,s,z)\ ) =$$
$$=\ pt(\ F(r,s,z) \wedge L(l,t(r,s,z),s,z) \wedge$$
$$\wedge M(r,s,z) \wedge A(r,l,s,z),\ B(r,l,s,z)\ ) =$$
$$= \exists u\ (\ F(u,s,\xi_1) \wedge L'(l,r,s,\xi_1) \wedge M(u,s,\xi_1) \wedge$$
$$\wedge A(u,l,s,\xi_1) \wedge (r = t(u,s,\xi_1)) \vee \ldots\ ) =$$
$$= \exists u\ (\ F'(t(u,s,\xi_1),s,\xi_1) \wedge L'(l,r,s,\xi_1) \wedge$$
$$\wedge M(u,s,\xi_1) \wedge A(u,l,s,\xi_1) \wedge P(u,s,\xi_1) \wedge$$
$$\wedge (r = t(u,s,\xi_1)) \vee \ldots\ ) =$$
$$= F'(r,s,\xi_1) \wedge L'(l,r,s,\xi_1) \wedge \exists u\ (\ M(u,s,\xi_1) \wedge$$
$$\wedge A(u,l,s,\xi_1) \wedge P(u,s,\xi_1)\ ) \vee \ldots \rightarrow$$
$$\rightarrow F'(r,s,z) \wedge L'(l,r,s,z).$$

2. $C(r,l,s) \neq 1$.

В этом случае постусловие кроме операторов присваивания и обновления списков содержит формулу базового языка. Обновление списков производится по вышеописанным правилам. Необходимо лишь отдельно рассмотреть случай, когда состояние E' содержит пустой список $l = list()$, а формула $C(r,l,s)$ содержит операторы доступа *get_from_tail*(l) или *get_from_head*(l). Тогда этот протокол не мог быть применен и соответствующая ветвь поведения не рассматривается. Рассмотрим влияние формулы $C(r,l,s)$ на формулу F.

Пусть $(\ r = t(u,s,z) \wedge C(r,l,s)\ ) \neq 0$ (условие валидности постусловия).

Операторы присваивания и формула $C(r,l,s)$ изменяют формулу в состоянии среды:

$$F'(r,s,z) \Leftrightarrow \exists u\ (\ F(u,s,z) \wedge A(u,l,s,z) \wedge$$
$$\wedge (r = t(u,s,z)) \wedge C(r,l,s)\ ).$$

Здесь u – вектор связанных переменных, введенных для обозначения старых значений атрибутов r. Отсюда следует:

$$F(r,s,z) =$$
$$= \exists v\ (\ F'(t(r,v,z),v,z)\ ) \wedge A(r,l,s,z) \wedge P(r,s,z),$$

или

$$pt^{-1}(\ F'(r,s,z)\ , A(r,l,s,z), B(r,l,s,z)\ ) =$$
$$= \exists v\ (\ F'(t(r,v,z),v,z)\ ) \wedge A(r,l,s,z) \wedge P(r,s,z).$$

Определяющее свойство такого обратного трансформера, в том числе и с учетом списков, доказывается аналогично вышеприведенному.

Примеры

Пример 1:
- предусловие: 1;
- постусловие: (x := x + 1) ∧ (y := z);
- формула F': (x > 0) ∧ (y > 5);
- полученная формула F:

(x + 1 > 0) ∧ (z > 5).

Пример 2:
- предусловие: 1;
- постусловие:

(x := x + 1) ∧ (y := z) ∧ add_to_tail(lst, 10);
- состояние E':

(x > 0) ∧ (y > 5) ∧ lst = list(x,y);
- полученное состояние E: (x + 1 > 0) ∧

∧ (z > 5) ∧ (z = 10) ∧ lst = list(x+1) →
(x + 1 > 0) ∧ (z = 10) ∧ lst = list(x+1).

Пример 3:
- предусловие: 1;
- постусловие: $(i := x[i]) \wedge (j := x[j])$;
- формула F': $i = j$;
- полученная формула F:

$(x[i] = x[j]) \wedge (i = j \vee i \neq j) \rightarrow$
$(i = j) \vee (x[i] = x[j] \wedge i \neq j)$.

Пример 4:
- предусловие: $x < 0$;
- постусловие: $y < 0$;
- формула F': $a < y \wedge y \leq x$;
- полученная формула F:

$\exists u (a < u \wedge u \leq x) \wedge x < 0 \rightarrow a < x \wedge x < 0$.

Пример 5:
- предусловие: $x < 0$;
- постусловие: $y := x + b \wedge y < b – 5$;
- формула F': $a < y \wedge y \leq x \wedge a > b – 10$;
- полученная формула F:

$\exists u (a < x + u \wedge x + u \leq x \wedge a > u –10) \wedge$
$\wedge x < 0 \rightarrow \exists u (u – 10 < a \wedge a < x + u \wedge$
$\wedge u \leq 0 \wedge x < 0) \rightarrow a < x \wedge x < 0$.

Пример 6 (неприменимый протокол):
- предусловие: $x < 0$;
- постусловие: $y := x + b \wedge y < b – 5$;
- формула F': $a < y \wedge y \leq x \wedge a > b$;
- полученная формула F:

$\exists u (a < x + u \wedge x + u \leq x \wedge a > u) \wedge x < 0 \rightarrow$
$\exists u (a < x + u \wedge u \leq 0 \wedge a > u \wedge x < 0) \rightarrow 0$.

Выводы

В данной работе рассмотрены методы прямого и обратного символьного моделирования, используемые для решения задач верификации систем базовых протоколов. Приведено решение задачи осуществления перехода системы от одного класса состояний к следующему. Такая трансформация производится с помощью предикатных трансформеров. Каждый известный класс состояний задается формулой базового языка. Построены алгоритмы прямого и обратного предикатных трансформеров для числовых, символьных и списочных структур данных. Трансформеры рассматриваются как функции, которые выводят из заданной формулы новую, определяющую класс состояний системы после перехода, совершенного под действием заданного базового протокола.

1. *Летичевский А.А., Капитонова Ю.В., Волков В.А. и др.* Спецификация систем с помощью базовых протоколов // Кибернетика и системный анализ. – 2005. – 4. – С. 3–21.
2. *Letichevsky and D. Gilbert.* A Model for Interaction of Agents and Environments // In D. Bert, C. Choppy, P. Moses, editors. Recent Trends in Algebraic Development Techniques. Lecture Notes in Computer Science 1827, Springer. – 1999. – P. 311–328.
3. *Летичевский А.А., Капитонова Ю.В., Волков В.А. и др.* Инсерционное программирование // Кибернетика и системный анализ. – 2003. – 1. – С. 19–32.
4. *Летичевский А.А.* Об одном классе базовых протоколов // Проблеми програмування. – 2005. – № 4. – С. 3–19.
5. *Летичевський О.О., Левченко І.А.* Символьна трасова генерація // Тези доп. Міжнар. конф. «Теоретичні та прикладні аспекти побудови програмних систем TAAPSD'2005». – К.: НаУКМА, Національний ун-т ім. Т.Г. Шевченка, Інститут програмних систем НАН України. – 2005. – С. 144–147.
6. *Колчин А.В.* Направленный поиск в верификации формальных моделей // Тези доп. Міжнар. конф. «Теоретичні та прикладні аспекти побудови програмних систем TAAPSD'2007». – Бердянськ. НаУКМА, Національний ун-т ім. Т.Г. Шевченка, Інститут програмних систем НАН України. – 2007. – С. 256–258.
7. *Letichevsky A., Kapitonova J., Letichevsky A. Jr., Volkov V., Baranov S., Kotlyarov V., Weigert T.* Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications // Proc. International Workshop, WITUL’04. Rennes (France). – 2004. – P. 30–38.
8. *Kapitonova J., Letichevsky A., Volkov V., and Weigert T.* Validation of Embedded Systems // In R. Zurawski, editor. The Embedded Systems Handbook. CRC Press, Miami. – 2005. – 51 p.

Получено 04.06.2008

Об авторе:
Потиенко Степан Валериевич,
ведущий математик.

Место работы автора:
Институт кибернетики им. В.М. Глушкова НАН Украины.
Тел.: (044) 200 8423.
e-mail: stepan@iss.org.ua