УДК: 004.05

Название: Предметно-ориентированные системы переходов: объектная модель и язык

Автор(ы):

Ануреев И.С. (Институт систем информатики СО РАН)

Аннотация: В статье представлены объектная модель и язык предметно-ориентированных систем переходов — нового формализма, предназначенного для спецификации и апробации формальных методов обеспечения надежности программного обеспечения.

Ключевые слова: предметно-ориентированные системы переходов, семантика, верификация, онтология

1. Введение. Обеспечение надежности программного обеспечения (ПО) — актуальная задача теории и практики программирования. Важную роль в этом играют формальные методы. В настоящее время имеется довольно много инструментов разработки надежного ПО, базирующихся на формальных методах. Они покрывают многие аспекты его разработки, от проектирования и прототипирования программных систем (ПС) до их формальной спецификации и верификации.

Однако, если в Semantic Web прослеживается тенденция к интеграции разнородных данных и сервисов, в разработке надежного ПО мы по-прежнему имеем дело с разрозненным набором отдельных инструментов, каждый из которых покрывает лишь отдельный специфический аспект разработки и, как правило, расчитан на использование лишь с небольшим числом компьютерных языков. Также заметен разрыв между огромным потенциалом развитых формальных методов и, за редким исключением, игрушечными примерами их применения [10]. Среди трудностей, препятствующих широкому распространению формальных методов, отметим сложность их изучения, высокую цену внедрения и неверие в них у представителей программной индустрии. Недостаточное внимание также уделяется технологическим аспектам разработки формальной семантики компьютерных языков, которая играет важную роль в обеспечении надежности ПО.

В [2, 3, 6] предложен унифицированный подход к обеспечению надежности программного обеспечения, который охватывает такие этапы разработки ПО, как прототипирование, проектирование, спецификация и верификация ПС. Этот подход также был использован для разработки формальной операционной семантики и логики безопасности (варианта аксиоматической семантики) компьютерных языков [5]. Он базируется на языке описания специализированных систем переходов. С помощью этих систем описываются модели ПС (в случае разработки семантики компьютерного языка описывается модель абстрактной машины этого языка.). Модель представляет собой пару — предметно-ориентированный язык, описывающий логику функционирования ПС, и его выполнимую семантику, описываемую специализированной системой переходов. Поэтому мы называем эти системы переходов предметно-ориентированными системами переходов (ПОСП). На языке ПОСП специфицируются формальные методы обеспечения надежности ПО, применяемые к моделям ПС, что позволяет достаточно быстро внедрять их в процесс разработки ПО. ПОСП можно также рассматривать как "технологические" машины абстрактных состояний [8], в которых формализованы язык и объектная модель состояний и правил переходов. При этом, в отличие от языков реализации машин абстрактных состояний ASML [7] и XASM [9], обеспечивается более высокий уровень абстракции при спецификации ПС.

В [1, 4, 6] был представлен язык описания ПОСП Atoment и подъязыки для описания отдельных видов ПОСП, ориентированных на решение тех или иных задач разработки надежного ПО. В этой статье мы представляем общую объектную модель всех видов ПОСП и новую более полную версию языка Atoment, базирующуюся на этой модели.

Работа выполнена при финансовой поддержке РФФИ, проект № 11-01-0028-а «Интегрированный мультиязыковый подход к верификации императивных программ» и междисциплинарного интеграционного проекта СО РАН № 3 «Принципы построения онтологии на основе концептуализаций средствами логических дескриптивных языков».

2. Предварительные понятия и обозначения.

Пусть Int, Nat, Nat^0 и $Bool$ обозначают множество целых чисел, множество натуральных чисел, $Nat \cup \{0\}$ и $\{true, false\}$, соответственно.

Пусть A^* (A^+) — множество всех конечных (непустых) последовательностей, состоящих из элементов множества A. Пусть seq_{emp} обозначает пустую последовательность. Пусть a_1 ... a_n обозначает последовательность из элементов a_1, ..., a_n. Пусть $len(a)$ и $a.i$ обозначают длину и i-й элемент последовательности или кортежа a, соответственно.

Пусть $a_{\in X}$ обозначает элемент a множества X, а $a_{\notin X}$ — элемент, не принадлежащий множеству X. Пусть $a_{\subseteq X}$ обозначает подмножество a множества X. Пусть $pset(X)$ — множество всех подмножеств множества X.

Пусть $X \to Y$ ($X \to_t Y$) обозначает множество всех (тотальных) функций из X в Y. Пусть $dom(f)$ обозначает область определения функции f. Будем считать, что $f(x) = und$, если $x \notin dom(f)$. Пусть $dom(f) \cap dom(g) = \emptyset$. Объединение $f \cup g$ функций f и g определяется как функция h такая, что $dom(h) = dom(f) \cup dom(g)$, $h(x) = f(x)$ для $x \in dom(f)$ и $h(x) = g(x)$ для $x \in dom(g)$. Пусть $im(f)$ обозначает образ функции f, т. е. множество $\{f(x) \mid x \in dom(f)\}$. Пусть $im(f, X)$ обозначает образ функции f относительно множества X, т. е. множество $\{f(x) \mid x \in X\}$.

Логическая функция $oDif$ определяется следующим образом: $oDif(f, g, M) = true$ тогда и только тогда, когда $f(x) = g(x)$ для $x \notin M$. Таким образом, значения функций f и g могут отличаться только на элементах множества M.

Говорят, что множество A определяется конструкторами f_1, ..., f_n, если выполнены следующие свойства:

- для любого $a \in A$ существуют $1 \leq i \leq n$ и $b \in dom(f_i)$ такие, что $a = f_i(b)$;
- для любых $1 \leq i \leq n$, $1 \leq j \leq n$, $a \in dom(f_i)$, и $b \in dom(f_j)$, если $f_i(a) = f_j(b)$, то $i = j$, и $a = b$.

Говорят, что множество A определяется конструкторами f_1, ..., f_n с ограничением $P \in A \to Bool$, если A определяется f_1, ..., f_n, и для любого $a \in A$ выполнено $P(a) = true$. Говорят, что a содержит b, если $a = b$, или существуют конструктор f и элементы x_1, ..., x_n, y_1, ..., y_m, c такие, что $a = f(x_1, ..., x_n, c, y_1, ..., y_m)$, и c содержит b.

Пусть $a \in A$ и $a = f_i(b)$. Расширим функции len и . на множества, определяемые конструкторами, следующим образом: если b — кортеж или последовательность, то $len(a) = len(b)$, и $a.k = b.k$. Например, $len(f_1(a, b)) = 2$, и $f_1(a, b).1 = a$ для кортежа (a, b), и $len(f_2(a)) = 3$, и $f_2(a).1 = b$ для последовательности $a = b\ c\ d$.

Система переходов определяется как пара $(Conf, tr)$, где $Conf$ — множество, $tr \in Conf \times Conf \to Bool$. Элементы $Conf$ называются конфигурациями. Функция tr называется отношением перехода.

3. Базовые понятия теории предметно-ориентированных систем переходов. Опишем виды объектов, из которых строится объектная модель ПОСП.

3.1. Атомы, символы, элементы. Пусть At — множество объектов, называемых атомами. Пусть $Int \subseteq At$ и $true, und \in At$.

Пусть $+v, \text{-}v \in At$. Атомы $+v$ и $\text{-}v$ называются спецификаторами аргументов. Пусть $ArgSpec = \{+v, \text{-}v\}$. Элементы множества $\{x \in At^+ \mid$ если $len(x) \neq 0$, то существует $1 \leq i \leq len(x)$ такое, что $x.i \notin ArgSpec\}$ называются символами. Пусть Sym обозначает множество символов. Число вхождений спецификаторов аргументов в символ f называется местностью f и обозначается $arity(f)$.

Множество El элементов определяется конструкторами $El_{at} \in At \to El$, $El_{seq} \in El^* \to El$, и $El_{fun} \in \cup_{n \in Nat_0}(El^n \to El) \to El$. Элементы $e_{\in im(El_{at})}$, $e_{\in im(El_{seq})}$ и $e_{\in im(El_{fun})}$ называется атомарным элементом, символьным вызовом и функциональным элементом, соответственно.

Далее, если не оговорено противное, буквы e и p (возможно, с индексами и штрихами) обозначают элементы и последовательности элементов, соответственно.

Пусть $a \in At$, $M \subseteq At$ и $b = b_1\ ...\ b_n \in At^+$. Тогда a_{el}, M_{el} и b_{el} обозначают $El_{at}(a)$, $\{a_{el} \mid a \in M\}$ и $(b_1)_{el}\ ...\ (b_n)_{el}$, соответственно. Для $e = i_{el}$, где $i \in Int$, пусть e_{int} обозначает i. Пусть $[p_1\ ...\ p_n]$ и $/[b_1\ ...\ b_n]/$ обозначают $El_{seq}(p_1\ ...\ p_n)$ и $[(b_1)_{el}\ ...\ (b_n)_{el}]$, соответственно. Пусть $/k[a_1\ ...\ a_k\ p_1\ ...\ p_n]$, $[p_1\ ...\ p_n\ b_1\ ...\ b_m]m/$ и $/k[a_1\ ...\ a_k\ p_1\ ...\ p_n\ b_1\ ...\ b_m]m/$ обозначают

$[(a_1)_{el} \ ... \ (a_k)_{el} \ p_1 \ ... \ p_n]$, $[p_1 \ ... \ p_n \ (b_1)_{el} \ ... \ (b_m)_{el}]$ и $[(a_1)_{el} \ ... \ (a_k)_{el} \ p_1 \ ... \ p_n \ (b_1)_{el} \ ... \ (b_m)_{el}]$, соответственно.

Пусть f — функция, $dom(f) \subseteq El$, и $M \in dom(f)$. Пусть $nar(f, M)$ обозначает функцию g такую, что $g(x) = f(x)$ для $x \in M$, и $g(x) = und_{el}$ для $x \in dom(f) \setminus M$.

3.2. Контексты вычисления. Пусть $EvCont$ — множество объектов, называемых контекстами вычисления. Определение контекстов вычисления может различаться в различных ситуациях. Далее буква q (возможно, с индексами и штрихами) обозначает контекст вычисления.

3.3. Состояния. Пусть $StSym \subseteq Sym$. Состояние s относительно базиса $(At, StSym)$ определяется как функция из $StSym \times Int \to_t \cup_{n \in Nat_0}(El^n \to_t El)$ такая, что $s(f, i) \in El^{arity(f)} \to_t El$, и выполнено свойство конечности состояния: $\{p \mid s(f,i)(p) \neq und_{el}, f \in StSym, i \in Int$ и $p \in El^{arity(f)}\}$ конечно. Пусть St — множество всех состояний относительно базиса $(At, StSym)$. Пусть $EvCont = St \times Int$. Элемент $s(f, i)$ называется интерпретацией (значением) символа f в контексте (s, i).

3.4. Контексты состояний. Функция $c \in StSym \to_t Int$ называется контекстом состояния. Пусть $StCont$ — множество всех контекстов состояний.

Элемент e называется представлением контекста состояния c и обозначается $contRep(c)$, если $e = El_{fun}(g)$, $g \in El \to El$, $g(El_{seq}(f_{el})) = (c(f))_{el}$ для $f \in StSym$, и $g(e) = und_{el}$ для $e \in El \setminus \{El_{seq}(f_{el}) \mid f \in StSym\}$. Пусть $n > 0$ и $f_1, ..., f_n \in StSym$.

Далее, если не оговорено противное, буквы s и c (возможно, с индексами и штрихами) обозначают состояния и контексты состояния, соответственно.

3.5. Интерпретации символов. Пусть $PreSym \subseteq Sym$. Интерпретация символов τ относительно базиса $(At, PreSym)$ определяется как функция из $PreSym \times EvCont \to_t \cup_{n \in Nat_0}(El^n \to_t El)$ такая, что $\tau(f, q) \in El^{arity(f)} \to_t El$ для любых $f \in PreSym$, и $q \in EvCont$. Элемент $\tau(f, q)$ называется значением символа f при интерпретации τ в контексте q.

3.6. Элементы как примеры символов. Множество $TypArg$ типизированных аргументов определяется конструктором $TypArg \in (El \times ArgSpec)^* \to TypArg$. Элемент e — пример $f_{\in Sym}$ относительно $a_{\in TypArg}$ тогда и только тогда, когда выполнено первое подходящее свойство:

- если $e = [e']$, $f = u_{\in ArgSpec}$, и $a = TypArg((e', u))$, то $true$;
- если $e = [e' \ p_{\in El^+}]$, $f = u_{\in ArgSpec} \ v_{\in At^*}$, и $a = TypArg((e', u) \ d)$, то $[p]$ — пример v относительно $TypArg(d)$;
- если $e = /[f_{\in At}]/$, то $true$;

- если $e = /1[b_{\in At}\ \ p_{\in El^+}]$, и $f = b\ w$, то $[p]$ — пример w относительно a;
- $false$.

Один и тот же элемент может быть примером для разных символов, поэтому возможен конфликт при выборе символа, соответствующего этому элементу. Мы считаем, что определена функция $match_{sym} \in pset(Sym) \times El \to StSym \times TypArg$, разрешающая этот конфликт, такая, что, если $match_{sym}(Sy, e) = (f, a)$, то e — пример f относительно a. Эта функция зависит от множества допустимых символов Sy. Элемент e называется примером f, если $match_{sym}(Sy, e) = (f, a)$. Элемент e называется вызовом символа f, если e — пример f. Элемент e называется вызовом символа f с аргументами a, если $match_{sym}(Sy, e) = (f, a)$.

3.7. Семантика элементов. Функция означивания элементов $val \in El \times EvCont \to El$ относительно базиса $(PreSym, \tau)$, где $EvCont = St \times StCont$, определяется следующим образом (применяется первое подходящее правило):

- $val(e_{\in im(El_{at}) \cup im(El_{fun})}, q) = e$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in StSym \setminus PreSym$, то $val(e, q) = q.1(f, q.2(f))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, $f \in StSym \cap PreSym$, и $q.1(f, q.2(f))(argVal(a, q)) = und_{el}$, то $val(e, q) = \tau(f, q)(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in ContSym \cap PreSym$, то $val(e, q) = q.1(f, q.2(f))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in PreSym$, то $val(e, q) = \tau(f, q)(argVal(a, q))$;
- $val(e, q) = und_{el}$.

Элемент $val(e, q)$ называется значением элемента e в контексте q. Функция $argVal \in TypArg \times EvCont \to El^*$ возвращает значения аргументов символа f и определяется следующим образом (применяется первое подходящее правило):

- $argVal(TypArg(seq_{emp}), q) = seq_{emp}$;
- $argVal(TypArg((/1[+v\ e], w)\ d), q) = val(e, q)\ argVal(TypArg(d), q)$;
- $argVal(TypArg((/[\text{-}v\ e], w)\ d), q) = e\ argVal(TypArg(d), q)$;
- $argVal(TypArg((e, +v)\ d), q) = val(e, q)\ argVal(TypArg(d), q)$;

- $argVal(TypArg((e,\text{-}v)\ d),q) = e\ argVal(TypArg(d),q)$.

Исходя из определений функций val и $argVal$, можно отметить следующее. Если элемент соответствует аргументу символа f со спецификатором $+v$, то функция $q.1(f,q.2(f))$ получает на вход значение этого элемента. Если элемент соответствует аргументу символа f со спецификатором -v, то функция $q.1(f,q.2(f))$ получает на вход сам этот элемент.

3.8. Подстановки. Функция $\sigma \in At \to El^*$ называется подстановкой. Пусть Sub — множество всех подстановок. Если $dom(\sigma) = \{x_1,...,x_n\}$, то σ может записываться как $((x_1,\sigma(x_1)),...,(x_n,\sigma(x_n)))$. Функция подстановки $sub \in El^* \times Sub \to El^*$ относительно σ определяется следующим образом (применяется первое подходящее правило):

- $sub(seq_{emp},\sigma) = seq_{emp}$;
- $sub(El_{at}(u_{\in dom(\sigma)}),\sigma) = \sigma(u)$;
- $sub(e_{\in im(El_{at}) \cup im(El_{fun})},\sigma) = e$;
- $sub([p],\sigma) = [sub(p,\sigma)]$;
- $sub(e\ p,\sigma) = sub(e,\sigma)\ sub(p,\sigma)$.

3.9. Конфигурации. Конфигурация t относительно базиса $(At, StSym, PreSym, \tau)$ определяется как тройка (p,s,c). Компоненты p, s и c конфигурации специфицируют, что выполняется, над чем и в каком контексте, соответственно.

Далее, если не оговорено противное, буква t (возможно, с индексами и штрихами) обозначает конфигурацию.

3.10. Правила перехода. Пусть $+s \in At$. Элементы вида u_{el} и $/[+s\ u]/$, где $u \in At$, называются спецификаторами переменных образца. Говорят, что эти элементы специфицируют переменную u. Пусть $VarSpec$ — множество всех спецификаторов переменных образца. Значение переменной u, определяемой спецификатором $El_{at}(u)$, принадлежит El. Значение переменной u, определяемой спецификатором $/[+s\ u]/$, принадлежит El^*.

Множество Rul правил перехода определяется конструктором $Rul \in El \times VarSpec^* \times El^* \to Rul$ со следующим ограничением: если $r \in Rul$, $1 \leq i,j \leq len(r.2)$, $r.2.i$ и $r.2.j$ специфицируют одну и ту же переменную, то $i = j$.

Пусть $r \in Rul$. Элемент $r.1$, переменные, входящие в $r.2$, и последовательность $r.3$ называются образцом, переменными образца и телом правила r, соответственно.

Говорят, что r применимо к e относительно $\sigma_{\in Sub}$, если $sub(r.1,\sigma) = e$, $dom(\sigma)$ — множество переменных образца правила r, $\sigma(u) \in El$ для $u_{el} \in r.2$, и $\sigma(u) \in El^*$ для $/[+s\ u]/ \in r.2$.

Правило может быть применимо к одному и тому же элементу относительно разных подстановок, поэтому возможен конфликт при выборе подстановки, соответствующей этому элементу. Мы считаем, что определена функция $match_{sub} \in El \times Rul \to Sub$, разрешающая этот конфликт, такая, что, если $match_{sub}(e, r) \neq und$, то r применимо к e относительно $match_{sub}(e, r)$.

На практике встречается случай, когда требуется подставить в качестве значения переменной образца не сопоставленный элемент, а его значение. Пусть $++v \in At$. Когда требуется сопоставить переменной образца элемент e и вычислить его значение, этот элемент заменяется в сопоставляемом выражении на $/1[++v\ e]$. Расширим определение функции $match_{sym}$ на этот случай. Соответствующая функция $match_{sub} \in El \times Rul \times Conf \to Sub$ определяется следующим образом:

- если $match_{sub}(e, r) = und$, то $match_{sub}(e, r, t) = und$;
- $dom(match_{sub}(e, r, t)) = dom(match_{sub}(e, r))$;
- для любого $u \in dom(match_{sub}(e, r))$ выполнено первое подходящее свойство:
  - если $u \in r.2$ и $match_{sub}(e, r)(u) = /1[++v\ e]$, то $match_{sub}(e, r, t)(u) = val(e, (t.2, t.3))$;
  - если $u \in r.2$, то $match_{sub}(e, r, t)(u) = match_{sub}(e, r)(u)$;
  - если $/[+s\ u]/ \in r.2$, то $len(match_{sub}(e, r, t)(u)) = len(match_{sub}(e, r)(u))$, и для любого $1 \leq i \leq len(match_{sub}(e, r)(u))$ выполнено первое подходящее свойство:
    - если $match_{sub}(e, r)(u).i = /1[++v\ e]$, то $match_{sub}(e, r, t)(u).i = val(e, (t.2, t.3))$;
    - $match_{sub}(e, r, t)(u).i = match_{sub}(e, r)(u).i$.

Говорят, что r применимо к e в конфигурации t, если $match_{sub}(e, r, t) \neq und$.

3.11. Определение ПОСП. Пусть $tr_{interp} \in Conf \times Conf \to Bool$ и $PredEl \subseteq El$. Система переходов $\rho = (Conf, tr)$ называется предметно-ориентированной относительно базиса $(At, StSym, PreSym, \tau, RulSet_{\subseteq Rul}, match_{sym}, match_{sub}, PredEl, tr_{interp})$, если выполнены следующие свойства:

- $Conf$ — множество всех конфигураций относительно $(At, StSym, PreSym, \tau)$;
- $tr(t, t') = true$ тогда и только тогда, когда $t.1 = e\ p$, и выполнено одно из двух условий: $e \in PredEl$ и $tr_{interp}(t, t') = true$, или $e \notin PredEl$, и существует $r_{\in RulSet}$ такое, что $tr(t, t', r) = true$. Функция tr с дополнительным аргументом r определяется следующим образом: $tr(t, t', r) = true$ тогда и только тогда, когда r применимо к e в t, $t'.1 = sub(r.3, match_{sub}(e, r, t))\ p$, $t'.2 = t.2$ и $t'.3 = t.3$.

Предметная ориентированность системы переходов ρ определяется ее базисом. Пусть $base(\rho)$ обозначает базис ρ. Функция tr_{interp} называется интерпретированным отношением перехода, а элементы $PredEl$ — предопределенными элементами. Пусть $tr(t, t') = true$. Тогда $t.1$ ($t.2$, $t.3$) и $t'.1$ ($t'.2$, $t'.3$) называются входной и выходной управляющими последовательностями (входным и выходным состояниями, входным и выходным контекстами) перехода, соответственно.

3.12. Понятия, связанные с переходом в ПОСП. Конфигурация t называется заключительной, если не существует t' такой, что $tr(t, t') = true$. Конфигурация t называется заключительной относительно $r \in RulSet$, если не существует t' такой, что $tr(t, t', r) = true$. Трассой называется конечная последовательность конфигураций t_1 ... t_n такая, что $tr(t_i, t_{i+1}) = true$ для $1 \leq i \leq n-1$. Пусть $Trace$ — множество всех трасс.

В следующих разделах, если не оговорено противное, используются следующие сокращения. Пусть $f \in At$. Пусть A_f обозначает $(f, t.3(f))$ для символов вида f. Пусть A_f обозначает $(f \text{ -}v, t.3(f \text{ -}v))$ для символов вида f -v. Пусть v_f и v'_f обозначают $t.2(A_f)$ и $t'.2(A_f)$, соответственно.

4. Предопределенные символы. В этом разделе описываются общезначимые предопределенные символы из $PreSym$.

4.1. Символ -*vv* -*v*. Пусть -$vv \in At$. Символ -vv -v имеет следующую семантику: $\tau(\text{-}vv \text{ -}v, q)(e) = e$.

4.2. Символ $+vv$ $+v$. Пусть $+vv \in At$. Символ $+vv$ $+v$ имеет следующую семантику: $\tau(+vv \ +v, q)(e) = val(e, q)$.

4.3. Нумерованные элементы. Пусть $el \in At$. Символ el $+v$ имеет следующую семантику:

- если $e \in Int_{el}$, то $\tau(el \ +v, q)(e) = /1[el \ e]$;
- в противном случае $\tau(el \ +v, q)(e) = und_{el}$.

Элементы вида $/[el \ i_{\in Int}]/$ называются нумерованными элементами.

Пусть $(x)^n$ обозначает последовательность из n вхождений атома x.

4.4. Контекстные символы. Пусть $cont \in At$. Семейство символов $\{cont \ +v \ (\text{-}v)^n\}_{n \in Nat}$ имеет следующую семантику:

- если $n \geq 1$ и $q \in St \times StSym$, то $\tau(cont \ +v \ (\text{-}v)^n, q)((i_{\in Int})_{el}, e_1, ..., e_n) = val([e_1 \ .... \ e_n], (q.1, q.2, i))$;
- в противном случае $\tau(cont \ +v \ (\text{-}v)^n, q)(e, e_1, ..., e_n) = und_{el}$.

Элементы вида $/1[cont \ p]/$ называются контекстными элементами.

Пусть $EvCont = St \times StCont \times Int$. Пусть $q' = (q.1, q.2)$. Функция $val \in El \times EvCont \to El$ относительно базиса $(PreSym, \tau)$ определяется следующим образом (применяется первое подходящее правило):

- $val(e_{\in im(El_{at}) \cup im(El_{fun})}, q) = e$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in StSym \setminus PreSym$, то $val(e, q) = q.1(f, q.3)(argVal(a, q'))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, $f \in StSym \cap PreSym$, и $s(f, q.3)(argVal(a, q')) = und_{el}$, то $val(e, q) = \tau(f, q)(argVal(a, q'))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in ContSym \cap PreSym$, то $val(e, q) = q.1(f, q.3)(argVal(a, q'))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in PreSym$, то $val(e, q) = \tau(f, q)(argVal(a, q'))$;
- $val(e, q) = und_{el}$.

4.5. Функциональные элементы. Пусть $sym \in At$. Семейство символов $\{sym\ (\text{-}v)^n\}_{n \in Nat}$ имеет следующую семантику:

- если $n \geq 1$, $f_{\in StSym} = e_1\ ...\ e_n$, и $q \in St \times StSym$, то $\tau(sym\ (\text{-}v)^n, q)(e_1, ..., e_n) = El_{fun}(q.1(f, q.2(f)))$;
- если $n \geq 1$, $f_{\in StSym} = e_1\ ...\ e_n$, и $q \in St \times StSym \times Int$, то $\tau(sym\ (\text{-}v)^n, q)(e_1, ..., e_n) = El_{fun}(q.1(f, q.3))$;
- в противном случае $\tau(sym\ (\text{-}v)^n, q)(e_1, ..., e_n) = und_{el}$.

Элементы вида $/[sym\ f_{\in Sym}]/$ называются функциональными элементами.

4.6. Атомы и символы. Пусть $is, atom, sym \in At$. Символы -v is $atom$ и -v is sym описывают атомы и символы, соответственно. Они имеют следующую семантику:

- $s(\text{-}v\ is\ atom)(x) = true$ тогда и только тогда, когда $x \in At$;
- $s(\text{-}v\ is\ sym)(x) = true$ тогда и только тогда, когда $x \in Sym$.

4.7. Символы $oDif$ $+v$ $+v$ $+v$ и $oDifSeq$ $+v$ $+v$ $+v$. Символ $oDif$ $+v$ $+v$ $+v$ имеет следующую семантику: $\tau(oDif\ +v\ +v\ +v, q)(x, y, z) = true$ тогда и только тогда, когда выполнено первое подходящее свойство:

- если $x = El_{fun}(x')$, $x' \in El^n \to_t El$, $y = El_{fun}(y')$, $y' \in El^n \to_t El$, то $oDif(x', y', \{z\}) = true$;
- $false$.

Символ $oDifSeq$ $+v$ $+v$ $+v$ имеет следующую семантику: $\tau(oDifSeq\ +v\ +v\ +v, q)(x, y, z) = true$ тогда и только тогда, когда выполнено первое подходящее свойство:

- если $x = El_{fun}(x')$, $x' \in El^n \to_t El$, $y = El_{fun}(y')$, $y' \in El^n \to_t El$ и $z = [z_1\ ...\ z_n]$, то $oDif(x', y', \{z_1, ..., z_n\}) = true$;
- $false$.

4.8. Ветвление по неопределенности. Пусть $else \in At$. Символ $+v\ else$ -v называется ветвлением по неопределенности и имеет следующую семантику:

- если $e = und_{el}$, то $\tau(+v\ else\ \text{-}v, q)(e, e') = val(e', q)$;
- В противном случае $\tau(+v\ else\ \text{-}v, q)(e, e') = e$.

Вариант $+v\ elsev\ +v$ этого символа имеет семантику

- если $e = und_{el}$, то $\tau(+v\ elsev\ +v, q)(e, e') = e'$;
- в противном случае $\tau(+v\ elsev\ +v, q)(e, e') = e$.

4.9. Условный элемент. Пусть $if, then, else \in At$. Символ $if\ +v\ then\ \text{-}v\ else\ \text{-}v$ называется условным элементом и имеет следующую семантику:

- если $e = true_{el}$, то $\tau(if\ +v\ then\ \text{-}v\ else\ \text{-}v, q)(e, e', e'') = val(e', q)$;
- в противном случае $\tau(if\ +v\ then\ \text{-}v\ else\ \text{-}v, q)(e, e', e'') = val(e'', q)$.

4.10. Нормализация постусловия. Пусть $ver \in At$. Символ $postNorm\ \text{-}v\ +v$ называется нормализацией постусловия и используется в дедуктивных ПОСП для нормализации постусловия при проверке корректности тела функции. Пусть $len(p) = n$. Этот символ имеет следующую семантику (применяется первый подходящий случай):

- $\tau(postNorm\ \text{-}v, q)(e, i_{\notin Int}) = und_{el}$;
- $\tau(postNorm\ \text{-}v, q)(e_{\in im(El_{at}) \cup im(El_{fun})}, i) = e$;
- $\tau(postNorm\ \text{-}v, q)([nextSt\ p], i) = [\tau(postNorm\ \text{-}v, q)(p_1, i)\ ...\ \tau(postNorm\text{-}v, q)(p_n, i)]$;
- $\tau(postNorm\ \text{-}v, q)([p], i) = [ver\ i\ i\ \tau(postNorm\ \text{-}v, q)(p_1, i)\ ...\ \tau(postNorm\text{-}v, q)(p_n, i)]$.

4.11. Операции над целыми числами. Целые числа задаются элементами вида i_{el}, где $i \in \{..., -2, -1, 0, 1, 2, ...\}$.

Пусть $+, \text{-}, *, \text{div}, \text{mod} \in At$. Арифметические операции сложения, вычитания, умножения, нахождения целой части от деления и нахождения остатка от деления определяются соответствующими символами $+v$ + $+v$, $+v$ - $+v$, $+v$ * $+v$, $+v$ div $+v$ и $+v$ mod $+v$.

Пусть $=, !=, <, <=, >, >= \ \in \ At$. Арифметические отношения над целыми числами определяются соответствующими символами $+v = +v$, $+v$!= $+v$, $+v < +v$, $+v <= +v$, $+v > +v$ и $+v >= +v$.

4.12. Логические операции. Пусть $false \in At$. Логические константы задаются элементами $true_{el}$ и $false_{el}$.

Пусть $and, or, not, =>, <=> \in At$. Логические связки конъюнкция, дизъюнкция, отрицание, импликация и эквивалентность определяются соответствующими символами $+v$ and $+v$, $+v$ or $+v$, not $+v$, $+v$ => $+v$ и $+v$ <=> $+v$. В качестве ложного значения выступает любой элемент, не равный $true_{el}$. Выражение $(e_1 \ and \ ... \ and \ e_n)$ является сокращением для $((\ ... \ (e_1 \ and \ e_2) \ and \ ... \ e_{n-1}) \ and \ e_n)$. Аналогичное сокращение используется для or.

Пусть $forall, exists \in At$. Кванторы всеобщности и существования $\forall x A$ и $\exists x A$ определяются символами forall -v -v и exist -v -v, соответственно. Выражение $forall \ x_1 \ ... \ x_n \ e$ является сокращением для $(forall \ x_1 \ ... \ (forall \ x_{n-1} \ (forall \ x_n \ e)))$. Аналогичное сокращение используется для $exist$.

5. Базовые виды предметно-ориентированных систем переходов. В этом разделе рассматриваются базовые виды ПОСП. Конкретные, используемые на практике ПОСП обычно определяются как комбинации базовых видов ПОСП.

5.1. ПОСП со счетчиком. Пусть $count \in At$. ПОСП ρ называется ПОСП со счетчиком, если $count \in StSym \setminus PreSym$, и $v_{count} \in \emptyset \to Int_{el}$ для любой $t \in Conf$. Символ $count$ называется счетчиком;

5.2. ПОСП с возвращаемым значением. Пусть $val \in At$. ПОСП ρ называется ПОСП с возвращаемым значением, если $val \in StSym \setminus PreSym$. Символ val называется спецификатором возвращаемого значения. Говорят, что управляющая последовательность p возвращает значение $v_{\in El}$ в состоянии s в контексте c, если $tr((p, s, c), t') = true$ для некоторой конфигурации t' и $t'.2(val, t'.3(val))() = v$.

5.3. ПОСП с инкрементом счетчика. Пусть $count{+}{+} \in At$. ПОСП ρ со счетчиком и возвращаемым значением называется ПОСП с инкрементом счетчика, если $PredEl$ включает элемент $/[count{+}{+}]/$, называемый инкрементом счетчика, со следующей семантикой: $tr_{interp}(t, t') = true$, где $t.1 = /[count{+}{+}]/ \ p$, тогда и только тогда, когда $t'.1 = p$, $oDif(t'.2, t.2, \{A_{count}, A_{val}\}) = true$, $v'_{count}() = ((v_{count}())_{int} + 1)_{el}$, $v'_{val}() = v'_{count}()$ и $t'.3 = t.3$.

5.4. ПОСП с генерацией нумерованных элементов. Пусть $newEl \in At$. ПОСП ρ со счетчиком и возвращаемым значением называется ПОСП с генерацией нумерованных элементов, если $PredEl$ включает элемент $/[newEl]/$, называемый генератором нумерованного элемента, со следующей семантикой: $tr_{interp}(t, t') = true$, где $t.1 = /[newEl]/\ p$, тогда и только тогда, когда $t'.1 = p$, $oDif(t'.2, t.2, \{A_{count}, A_{val}\}) = true$, $v'_{count}() = ((v_{count}())_{int} + 1)_{el}$, $v'_{val}() = /1[el\ v'_{count}()]$ и $t'.3 = t.3$.

5.5. Условные ПОСП. ПОСП ρ называется условной ПОСП, если выполнены следующие свойства:

- Rul определяется конструктором $Rul_{cond} \in El \times VarSpec^* \times El^* \times El \to Rul$. Дополнительная компонента $r.4$ называется условием r, а правило r — условным правилом;
- отношение применимости правила переопределяется следующим образом: r применимо к e в t, если $match_{sub}(e, r, t) \neq und$, и $val(sub(r.4, match_{sub}(e, r, t)), (t.2, t.3)) = true_{el}$.

Таким образом, условное правило применимо только в том случае, если выполнено его условие.

5.6. ПОСП с переменными истории. Пусть $hvar \in At$. ПОСП ρ со счетчиком называется ПОСП с переменными истории, если выполнены следующие свойства:

- для любого $i \in Int$ выполнено свойство $hvar\ i \in StSym \setminus PreSym$. Элемент вида $/[hvar\ i]/$ называется переменной истории ПОСП ρ, а $i_{\in Int}$ — индексом этой переменной. В переменных истории хранятся промежуточные результаты (история) функционирования ρ;
- Rul определяется конструктором $Rul_{hvar} \in El \times VarSpec^* \times El^* \times At^* \to Rul$ со следующим ограничением: множества переменных образца и элементов $r.4$ не пересекаются для любого $r_{\in RulSet}$. Дополнительная компонента $r.4$ называется декларацией переменных истории и используется для добавления переменных истории $/[hvar\ i]/$ с индексами $i \in Int$ такими, что $(v_{count}())_{int} < i \leq (v_{count}())_{int} + len(r.4)$, в управляющие последовательности. Элементы $r.4$ называются переменными истории правила r;
- пусть функция $hvarSub \in Rul \times St \to Sub$ определяется следующим образом: $dom(hvarSub)$ — множество переменных истории r и $hvarSub(r, s)(r.4.j) = /[hvar\ (v_{count}())_{int} + j]/$ для $1 \leq j \leq len(r.4)$. Отношение tr переопределяется следующим образом: $tr(t, t', r) = true$ тогда и только тогда, когда $t.1 = e\ p$, r применимо к e в t, $t'.1 = sub(r.3, match_{sub}(e, r, t) \cup hvarSub(r, t.2))\ p$, $oDif(t'.2, t.2, \{A_{count}\}) = true$, $v'_{count}() = ((v_{count}())_{int} + len(r.4))_{el}$, и $t'.3 = t.3$.

5.7. ПОСП с частично интерпретированными телами правил. ПОСП ρ называется ПОСП с частично интерпретированными телами правил, если tr переопределяется следующим образом: $tr(t, t', r) = true$ тогда и только тогда, когда $t.1 = e\ p$, и существует $r_{\in RulSet}$ такое, что r применимо к e в t, $t'.1 = partVal(sub(r.3, match_{sub}(e, r, t)), (t.2, t.3))\ p$, $t'.2 = t.2$, и $t'.3 = t.3$;

Пусть $EvCont = St \times StCont$. Функция $partVal \in El^* \times EvCont \to El^*$ вычисляет элементы вида $/1[interp\ e]$ в теле правила r и определяется следующим образом (применяется первое подходящее правило):

- $partVal(seq_{emp}, q) = seq_{emp}$;
- $partVal(e_{\in im(El_{at}) \cup im(El_{fun})}, q) = e$;
- $partVal(/1[interp\ e], q) = val(partVal(e, q), q)$;
- $partVal([p], q) = [partVal(p, q)]$;
- $partVal(e\ p, q) = partVal(e, s, c)\ partVal(p, q)$.

5.8. ПОСП с определениями символов. Эти ПОСП используются для спецификации предопределенных символов. Значение символа, сопоставленного с образцом, определяется как значение, возвращаемое телом правила. ПОСП ρ называется ПОСП с определениями символов, если выполнены следующие свойства:

- наряду с обычными правилами используются правила, в которых множество спецификаторов переменных $VarSpec$ переопределяется следующим образом. Элементы вида $/[\text{-}v\ u]/$ и u, где $u \in At$, называются спецификаторами переменных образца. Пусть $r \in RulSet$. Определяемый правилом r символ f является результатом замены в образце правила r переменных со спецификаторами $/[\text{-}v\ u]/$ и u на спецификаторы аргументов -v и $+v$, соответственно.

- значение $val(e, q)$ вызова функции f в контексте $q_{\in St \times StCont}$ определяется следующим образом (применяется первое подходящее правило):
  - если $tr((e, q.1, q.2), t') = true$, и $tr((e, q.1, q.2), t'') = true$, где t', t'' — заключительные конфигурации, и $val((val), (t'.2, t'.3)) \neq val((val), (t''.2, t''.3))$, то $val(e, q) = und_{el}$;
  - если не существует заключительной конфигурации t' такой, что $tr((e, q.1, q.2), t') = true$, то $val(e, q) = und_{el}$;
  - если $tr((e, q.1, q.2), t') = true$, где t' — заключительная конфигурация, то $val(e, q) = val((val), (t'.2, t'.3))$.

Заметим, что при вычислении значения вызова символа f, определяемого правилом ПОСП ρ, после вычисления, следующие переходы выполняются в контексте, в котором вычислялся f, т. е. происходит откат из конфигурации, полученной в результате вычисления f, в исходную конфигурацию.

5.9. Бэктрекинг в ПОСП. Использование бэктрекинга в ПОСП расширяет выразительные возможности этих систем. Формально он определяется как отношение перехода на последовательностях расширенных конфигураций.

Пусть $EConf$ — множество четверок $(p, s, c, R_{\subseteq RulSet})$, называемых расширенными конфигурациями. Четвертая компонента определяет, какие правила перехода уже применялись при переходе из конфигурации (p, s, c). Пусть $BackSym$ — конечное множество символов. Оно специфицирует символы, значения которых сохраняются при возвратах. Функция $ifSt(s, s')$ возвращает состояние и определяется следующим образом:

- если $f \in BackSym$, то $ifSt(s, s')(f) = s'(f)$;
- если $f \in StSym \setminus BackSym$, то $ifSt(s, s')(f) = s(f)$.

Выделяется два вида бэктрекинга — управляемый бэктрекинг и полный перебор с возвратом.

5.10. ПОСП с управляемым бэктрекингом. Пусть $backtrack, branch \in At$. Пусть $ba(s, c)$ и $fi(s, c)$ обозначают $(backtrack_{el}, s, c, \emptyset)$ и $(seq_{emp}, s, c, \emptyset)$, соответственно. Пусть $w \in im(El_{seq})^*$, и $t, t' \in EConf$. Отношение перехода $tr \in EConf^* \times EConf^* \to Bool$ называется управляемым бэктрекингом, если $tr(u, u') = true$ тогда и только тогда, когда выполнено первое подходящее свойство:

- $u = u''\ (backtrack_{el}\ p, s, c, R)$, где $p \neq seq_{emp}$, или $R \neq \emptyset$, и $u' = u''\ ba(s, c)$;
- $u = u''\ (/1[branch\ [p']\ w]\ p, s, c, R)\ ba(s'', c'')$, и $u' = u''\ (/1[branch\ w]\ p, s, c, R)$ $(p'\ p, ifSt(s, s''), c, \emptyset)$;
- $u = u''\ t\ (/[branch]/\ p, s, c, R)\ ba(s'', c'')$, и $u' = u''\ t\ ba(s'', c'')$;
- $u = (/[branch]/\ p, s, c, R)\ ba(s'', c'')$, и $u' = ba(ifSt(s, s''), c)$;
- $u = u''\ (/1[branch\ [p']\ w]\ p, s, c, R)$, и $u' = u''\ (/1[branch\ w]\ p, s, c, R)\ (p'\ p, s, c, \emptyset)$;
- $u = u''\ (/[branch]/\ p, s, c, R)$, и $u' = u''\ fi(s, c)$;
- $u = u''\ (e\ \ p, s, c, R)\ \ ba(s'', c'')$, $e \notin PredEl$, $r \in RulSet \setminus R$, $tr((e\ p, ifSt(s, s''), c), (p', s', c'), r) = true$, и $u' = u''\ (e\ p, s, c, R \cup \{r\})\ (p', s', c', \emptyset)$;
- $u = u''\ t\ t'\ ba(s'', c'')$, и $u' = u''\ t\ ba(s'', c'')$;

- $u = (p, s, c, R)\ ba(s'', c'')$, и $u' = ba(ifSt(s, s''), c)$;

- $u = ba(s, c)$, и $u' = fi(s, c)$;

- $u = u''\ (e\ p, s, c, R)$, $e \notin PredEl$, $r \in RulSet \setminus R$, $tr((e\ p, s, c), (p', s', c'), r) = true$, и $u' = u''\ (e\ p, s, c, R \cup \{r\})\ (p', s', c', \emptyset)$;

- если $u = u''\ (e\ p, s, c, R)$, $e \in PredEl$, $tr_{interp}((e\ p, s, c), (p', s', c')) = true$, и $u' = u''\ (ep, s, c, R)\ (p', s', c', \emptyset)$;

- $false$.

Элемент $backtrack_{el}$, называемый условием возврата, инициирует возврат в ПОСП. Элемент $/1[branch\ p]$ называется элементом ветвления и используется для выполнения перебора вариантов с возвратом.

ПОСП называется ПОСП с управляемым бэктрекингом, если tr на расширенных конфигурациях является управляемым бэктрекингом.

5.11. ПОСП с полным перебором. Отношение перехода $tr \in EConf^* \times EConf^* \to Bool$ называется полным перебором с возвратом, если $tr(u, u') = true$ тогда и только тогда, когда выполнено первое подходящее свойство:

- $u = u''\ (p, s, c, R)$, где $p \neq seq_{emp}$, или $R \neq \emptyset$, (p, s, c, R) — заключительная конфигурация и $u' = u''\ fi(s, c)$;

- $u = u''\ ([/1[cases\ p']\ p, s, c, R)\ u'''$, где $p \neq seq_{emp}$, или $R \neq \emptyset$, (p, s, c, R) — заключительная конфигурация, и $u' = u''\ ([/1[casesRest\ ()\ p']\ p, s, c, R)\ u'''$;

- $u = u''\ ([/1[casesRest\ (w_1\ ...\ w_n)1[if\ e\ then_{el}\ p']\ p'']\ p, s, c, R)\ fi(s'', c'')$, и
  $u' = u''\ (/1[casesRest\ (w_1\ ...\ w_n\ e)\ p'']\ p, s, c, R)$
  $(/1[assume\ ((not(w_1\ and\ ...\ and\ w_n))\ and\ e)]\ p'\ p, ifSt(s, s''), c, \emptyset)$;

- $u = u''\ ([/1[casesRest\ (w_1\ ...\ w_n)1[else\ p']\ p'']\ p, s, c, R)\ fi(s'', c'')$, и
  $u' = u''\ (/1[assume\ (not(w_1\ and\ ...\ and\ w_n))]\ p'\ p, ifSt(s, s''), c, \emptyset)$;

- $u = (/[casesRest]/\ p, s, c, R)\ fi(s'', c'')$, и $u' = fi(ifSt(s, s''), c)$;

- $u = u''\ (/[casesRest]/\ p, s, c, R)\ fi(s'', c'')$, и $u' = u''\ fi(s'', c'')$;

- $u = u''\ ([/1[casesRest\ (w_1\ ...\ w_n)\ /1[if\ e\ then_{el}\ p']\ p'']\ p, s, c, R)$, и
  $u' = u''\ (/1[casesRest\ (w_1\ ...\ w_n\ e)\ p'']\ p, s, c, R)$
  $(/1[assume\ ((not(w_1\ and\ ...\ and\ w_n))\ and\ e)]\ p'\ p, s, c, \emptyset)$;

- $u = u''$ $([/1[casesRest\ (w_1\ ...\ w_n)\ /1[else\ p']\ p'']\ p, s, c, R)$, и
  $u' = u''$ $(/1[assume\ (not(w_1\ and\ ...\ and\ w_n))]\ p'\ p, s, c, \emptyset)$;
- $u = u''$ $(/[casesRest]/\ p, s, R)$, и $u' = u''\ fi(s, c)$;
- $u = u''$ $(/1[branch\ [p']\ w]\ p, s, c, R)\ fi(s'', c'')$, и
  $u' = u''$ $(/1[branch\ w]\ p, s, c, R)\ (p'\ p, ifSt(s, s''), c, \emptyset)$;
- $u = (/[branch]/\ p, s, c, R)\ fi(s'', c'')$, и $u' = fi(ifSt(s, s''), c)$;
- $u = u''$ $(/[branch]/\ p, s, c, R)\ fi(s'', c'')$, и $u' = u''\ fi(s'', c'')$;
- $u = u''$ $(/1[branch\ [p']\ w]\ p, s, c, R)$, и $u' = u''$ $(/1[branch\ w]\ p, s, c, R)\ (p'\ p, s, c, \emptyset)$;
- $u = u''$ $(/[branch]/\ p, s, R)$, и $u' = u''\ fi(s, c)$;
- $u = u''$ $(e\ \ p, s, c, R)\ fi(s'', c'')$, $e \notin PredEl$, $r \in RulSet \setminus R$, $tr((e\ p, ifSt(s, s''), c), (p', s', c'), r) = true$, и $u' = u''$ $(e\ p, s, c, R \cup \{r\})\ (p', s', c', \emptyset)$;
- $u = u''\ t_{\in EConf}\ t'_{\in EConf}\ fi(s'', c'')$, и $u' = u''\ t\ fi(s'', c'')$;
- $u = (p, s, c, R)\ fi(s', c')$, и $u' = fi(ifSt(s, s'), c)$;
- $u = u''$ $(e\ p, s, c, R)$, $e \notin PredEl$, $r \in RulSet \setminus R$, $tr((e\ p, s, c), (p', s', c'), r) = true$, и $u' = u''$ $(e\ p, s, c, R \cup \{r\})\ (p', s', c', \emptyset)$;
- если $u = tr(u''$ $(e\ p, s, c, R)$, $e \in PredEl$, $tr_{interp}((e\ p, s, c), (p', s', c')) = true$, и $u' = u''$ $(e\ p, s, c, R)\ (p', s', c', \emptyset)$;
- $false$.

При полном переборе с возвратом семантика выражений $/1[branch\ p]$ и $/1[cases\ p]$ изменяется по сравнению с управляемым бэктрекингом. В этом случае элемент $/1[branch\ p]$ называется элементом полного ветвления и используется для выполнения полного перебора вариантов с возвратом. Элемент

$$/1[cases\ /1[if\ e_1\ then_{el}\ p_1]\ ...\ /1[if\ e_n\ then_{el}\ p_n]\ /1[else\ p]]$$

называется дедуктивным условным ветвлением и используется для перебора вариантов $p_1, ..., p_n, p$ с условиями $p_1, ..., p_n, (not\ (p_1\ and\ ...\ and\ p_n))$. Такое ветвление называется дедуктивным, т.к. определяется через выражение $/1[assume\ x]$, которое в этом случае имеет семантику дедуктивного условия продолжения.

ПОСП называется ПОСП с полным перебором, если tr на расширенных конфигурациях является полным перебором с возвратом.

5.12. Версионные ПОСП. Пусть $st, cont \in At$. ПОСП ρ называется версионной ПОСП, если выполнены следующие свойства:

- $base(\rho)$ включает дополнительно конечные множества $VerSym_{\subseteq Sym}$ и $ContFreePreSym_{\subseteq PreSym}$. Элементы множеств $VerSym$ и $ContFreePreSym$ называются версионными символами и свободными от контекста предопределенными символами, соответственно. Если $f \in ContFreePreSym$, то $\tau(f,q) = \tau(f,q')$ для любых q и q', т. е. интерпретация свободного от контекста символа не зависит от контекста вычисления.

- $st\text{-}v \in StSym \setminus PreSym$, и $v_{st}(/[f]/) \in Int_{el}$ для любого $f \in VerSym$, и любой $t \in Conf$;

- $cont\text{-}v \in StSym \setminus PreSym$, и $v_{count}(/[f]/) \in Int_{el}$ для любого $f \in VerSym$, и любой $t \in Conf$.

Версионные ПОСП используются для моделирования выполнения других ПОСП. В частности, они используются для декларативного описания изменений состояния и контекста в виде формулы, называемой предусловием, в дедуктивных системах с прямым прослеживанием (раздел 5.13). Множество символов состояния моделируемой системы специфицируется множеством версионных символов. Конфигурации трассы моделируемой ПОСП, в которых изменилось состояние или контекст по сравнению с предыдущей конфигурацией, нумеруются последовательными целыми числами. Символ $st\text{-}v$ специфицирует наибольший номер конфигурации, в которой, возможно, было модифицировано значение версионного символа. Символ $cont\text{-}v$ специфицирует текущий контекст моделируемой ПОСП и называется версионным контекстом.

5.13. Дедуктивные ПОСП с прямым прослеживанием. Дедуктивные ПОСП используются в реализации аксиоматической семантики и логики безопасности программ. Пусть $pre, verCond \in At$. Версионная ПОСП ρ с полным перебором называется дедуктивной ПОСП с прямым прослеживанием, если выполнены следующие свойства:

- $pre \in StSym \setminus PreSym$;

- $verCond \in StSym \setminus PreSym$, и $v_{verCond}() \in im(El_{seq})$ для любой $t \in Conf$;

- $verCond \in BackSym$.

Символ pre, называемый предусловием, специфицирует в виде формулы изменения состояния и контекста моделируемой системы. Символ $verCond$ хранит последовательность порожденных условий корректности.

5.14. ПОСП с обратным проходом. ПОСП ρ называется ПОСП с обратным проходом, если tr переопределяется следующим образом: $tr(t,t')=true$ тогда и только тогда, когда $t.1=p\ e$ и выполнено любое из двух условий: $e\in PredEl$, и $tr_{interp}(t,t')=true$, или $e\notin PredEl$, и существует $r_{\in RulSet}$ такое, что $tr(t,t',r)=true$. Функция tr с дополнительным аргументом r определяется следующим образом: $tr(t,t',r)=true$ тогда и только тогда, когда r применимо к e в t, $t'.1=p\ sub(r.3, match_{sub}(e,r,t))$, $t'.2=t.2$, и $t'.3=t.3$.

5.15. Онтологические ПОСП. Пусть $EvCont = St\times Int$. Пусть $is, ontSym$, $instSym\in At$. ПОСП ρ называется онтологической ПОСП, если выполнены следующие свойства:

- $\text{-}v\ is\ ontSym\in StSym\setminus PreSym\neq\emptyset$. Символ $f_{\in StSym}$ называется онтологическим символом в q, если $q.1(\text{-}v\ is\ ontSym, q.2)(/[f]/)=true_{el}$. Онтологические символы специфицируют элементы (понятия, атрибуты, отношения и т. п.) онтологии системы, описываемой с помощью ρ;

- $\text{-}v\ is\ instSym\in StSym\setminus PreSym$. Символ $f\in StSym$ называется символом экземпляризации в q, если $s(\text{-}v\ is\ instSym, q)(/[f]/)=true_{el}$. Символы экземпляризации специфицируют связи элементов онтологии с экземплярами;

- $q.1(\text{-}v\ is\ instSym, q.2)(/[+v\ is\ \text{-}v]/)=true_{el}$. Свойство $q.1(+v\ is\ \text{-}v, q.2)(x,y)=true_{el}$ означает, что x — экземпляр понятия y в q;

- Rul определяется конструктором $Rul_{ont}\in El\times El^*\to Rul$ со следующим ограничением: $r.1$ имеет вид $/[x\ is\ y]/$, где $x,y\in At$, для каждого r вида $Rul_{ont}(...)$. Семантика правила $Rul_{ont}(/[x\ is\ y]/, z)$ совпадает с семантикой условного правила $Rul_{cond}(x_{el}, x, z, /[x\ is\ y]/)$.

Рассмотрим примеры онтологических символов:

- $q.1(\text{-}v\ isConcept, q.2)(x)=true_{el}$ означает, что x — понятие в q;

- $q.1(\text{-}v\ isAttributeOf\ \text{-}v, q)(x,y)=true_{el}$ означает, что x — атрибут понятия y в q;

- $q.1(\text{-}v\ isAttributeOf\ \text{-}v\ ofType\ \text{-}v, q.2)(x,y,z)=true_{el}$ означает, что x — атрибут понятия y типа z в (s,i). Типом является некоторое понятие. Например, $q.1(+v\ isAttributeOf\ y\ ofType\ z, q.2)(x,y,int_{el}])=true_{el}$ означает, что x — целочисленный атрибут понятия y в q.

Рассмотрим примеры символов экземпляризации:

- $q.1(\text{-}v\ of\ +v, q.2)(x,y)$ возвращает значение атрибута x экземпляра y некоторого понятия в q;

- $q.1(\text{-}v\ of\ +v\ of\ \text{-}v, q.2)(x, y, z)$ возвращает значение атрибута x экземпляра y понятия z в q. Этот символ используется, чтобы разрешить конфликт в случае, когда y является экземпляром нескольких понятий, имеющих атрибут x.

5.16. ПОСП с сопоставлением с образцом на последовательностях. ПОСП ρ называется ПОСП с сопоставлением с образцом на последовательностях, если выполнены следующие свойства:

- конструктор Rul переопределяется по первому аргументу: $Rul \in El^+ \times VarSpec^* \times El^* \to Rul$. Таким образом, образцы правил таких ПОСП — непустые последовательности элементов;

- $r_{\in Rul}$ применимо к $p_{\in El^{len(r.1)}}$ относительно $\sigma_{\in Sub}$, если $sub(r.1, \sigma) = p$, $dom(\sigma)$ — множество переменных образца правила r, $\sigma(u) \in El$ для $u_{el} \in r.2$, и $\sigma(u) \in El^*$ для $/[+s\ u]/ \in r.2$;

- функция $match_{sub}$ переопределяется таким образом, что $match_{sub} \in El^+ \times Rul \to Sub$, и, если $match_{sub}(p, r) \neq und$, то $len(p) = len(r.1)$, и r применимо к p относительно $match_{sub}(p, r)$. Функция $match_{sub}$ расширяется на $El^+ \times Rul \times Conf \to Sub$ так же, как и в случае обычных правил ПОСП. Говорят, что r применимо к p в t, если $match_{sub}(p, r, t) \neq und$.

- Функция tr определяется следующим образом: $tr(t, t', r) = true$ тогда и только тогда, когда $t.1 = p''_{\in El^+}\ p$, r применимо к p'', $t'.1 = sub(r.3, match_{sub}(p'', r, t))\ p$, $t.2 = t'.2$, и $t.3 = t'.3$.

6. Комбинированные виды предметно-ориентированных систем переходов. На практике, как правило, используются комбинации базовых видов ПОСП. Рассмотрим некоторые из них.

6.1. Условные ПОСП с переменными истории. Эти ПОСП обладают следующим дополнительным свойством: вместо конструкторов Rul_{cond} и Rul_{hvar} используется комбинированный конструктор $Rul_{cond+hvar} \in El \times VarSpec^* \times El^* \times El \times At^* \to Rul$, где дополнительные компоненты $r.4$ и $r.5$ являются условием и декларацией переменных истории правила r, соответственно.

6.2. Условные ПОСП с частично интерпретированными телами правил. Эти ПОСП обладают следующими дополнительными свойствами:

- отношение применимости правила переопределяется следующим образом: r применимо к e в t, если $match_{sub}(e, r, t) \neq und$, и $val(partVal(sub(r.4, match_{sub}(e, r, t)), (t.2, t.3)), (t.2, t.3)) = true_{el}$;

- tr переопределяется следующим образом: $tr(t, t', r) = true$ тогда и только тогда, когда $t.1 = e\ p$, и существует $r_{\in RulSet}$ такое, что r применимо к e в t, $t'.1 = partVal(sub(r.3, match_{sub}(e, r, t)), (t.2, t.3))\ p$, $t'.2 = t.2$, и $t'.3 = t.3$.

6.3. ПОСП с переменными истории с частично интерпретированными телами правил. В таких ПОСП сначала выполняется подстановка переменных истории, а затем выполняется частичная интерпретация тел правил.

6.4. ПОСП с полным управляемым перебором. Эти ПОСП комбинируют ПОСП с управляемым бэктрекингом и ПОСП с полным перебором. Семантика выражений $/1[branch\ p]$ и $/1[cases\ p]$ определяется как в ПОСП с полным перебором. Если требуется их интерпретация в ПОСП с управляемым перебором, вместо этих выражений используются выражения $/1[branch*\ p]$ и $/1[cases*\ p]$.

6.5. ПОСП с управляемым бэктрекингом и со счетчиком. Эти ПОСП обладают следующим дополнительным свойством: $count \in BackSym$.

6.6. ПОСП с управляемым бэктрекингом и с переменными истории. Эти ПОСП обладают следующим дополнительным свойством: $hvar\ i \in BackSym$ для любого $i \in Int$.

6.7. ПОСП с полным перебором и со счетчиком. Эти ПОСП обладают следующим дополнительным свойством: $count \in BackSym$.

6.8. ПОСП с полным перебором и с переменными истории. Эти ПОСП обладают следующим дополнительным свойством: $hvar\ i \in BackSym$ для любого $i \in Int$.

6.9. Версионные ПОСП с переменными истории. Для этих ПОСП выполнены следующие свойства:

- $hvar\ i \in VerSym$ для любого $i \in Int$;

- свойство конечности состояния заменяется на свойство частичной конечности состояния: $\{p \mid s(f, i)(p) \neq und_{el}, f \in StSym, i \in Int$ и $p \in El^{arity(f)}\} \setminus \{/[hvar\ i]/ \mid i \in Int\}$ конечно.

7. Предопределенные элементы. В этом разделе определены часто встречающиеся предопределенные элементы.

Будем считать, если не оговорено противное, что $x, y, z \in El$.

7.1. Останов. Пусть $stop \in At$. Элемент $/[stop]/$ называется остановом и имеет следующую семантику: $tr_{interp}(t, t'') = true$, где $t.1.1 = /[stop]/$, тогда и только тогда, когда $t'.1 = seq_{emp}$, $t'.2 = t.2$ и $t'.3 = t.3$.

7.2. Небезопасное завершение. Пусть $fail \in At$. Элемент $/[fail]/$ называется называется небезопасным завершением. Он специфицирует некорректное завершение ПОСП.

Конфигурация t называется небезопасной, если $t.1.1 = /[fail]/$. В противном случае t называется безопасной. Отношение tr должно удовлетворять следующему свойству: если t — небезопасная конфигурация, то t — заключительная конфигурация. Трасса, последний элемент которой — небезопасная конфигурация, называется небезопасной.

7.3. Условие продолжения. Пусть $assume \in At$. Элемент $/1[assume\ x]$ называется условием продолжения и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[assume\ x]\ p$, тогда и только тогда, когда $t'.2 = t.2$, $t'.3 = t.3$, и выполнено первое подходящее свойство:

- если $val(x, (t.2, t.3)) = true_{el}$, то $t'.1 = p$;
- $t'.1 = backtrack_{el}\ p$.

Условие продолжения базируется на элементе $backtrack_{el}$ и используется в ПОСП с управляемым бэктрекингом.

Пусть A^f обозначает $(f, t.3(f))$.

7.4. Условие модификации. Пусть $modify, nextSt \in At$. Элемент $/1[modify\ x]$ называется условием модификации и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[modify\ x]\ p$, тогда и только тогда, когда $t'.3 = t.3$ и выполнено первое подходящее свойство:

- если $val(x, (t.2, t.3, t'.2)) = true_{el}$, и $M = \{A^f \mid f \in StSym, match_{sym}(StSym, e) = (f, a)$, и x содержит $/1[nextSt\ \ e]$ для некоторых e и $a\} \neq \emptyset$, то $t'.1 = p$, и $oDif(t'.2, t.2, M) = true$;
- $t'.1 = backtrack_{el}\ p$, и $t'.2 = t.2$.

Вхождение выражения $/1[nextSt\ p]$ в x указывает на то, что значение выражения $[p]$ определяется в выходном состоянии $t'.2$.

Пусть $EvCont = St \times StCont \times St$. Функция $val \in El \times EvCont \to El$ определяется следующим образом (применяется первое подходящее правило):

- $val(e_{\in im(El_{at}) \cup im(El_{fun})}, q) = e$;
- если $match_{sym}(StSym \cup PreSym, [p]) = (f, a)$, и $f \in StSym \setminus PreSym$, то $val(/1[nextSt\ p], q) = q.3(f, q.2(f))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, [p]) = (f, a)$, $f \in StSym \cap PreSym$, и $q.3(f, q.2(f))(argVal(a, q)) = und_{el}$, то $val(/1[nextSt\, p], q) = \tau(f, (q.3, q))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, [p]) = (f, a)$, и $f \in ContSym \cap PreSym$, то $val(/1[nextSt\ p], q) = q.3(f, q.2(f))(argVal(a, q))$;

- если $match_{sym}(StSym \cup PreSym, [p]) = (f, a)$, и $f \in PreSym$, то $val(/1[nextSt\ p], q) = \tau(f, (q.3, q))(argVal(a, q))$;
- $val(/1[nextSt\ p], q) = und_{el}$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in StSym \setminus PreSym$, то $val(e, q) = q.1(f, q.2(f))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, $f \in StSym \cap PreSym$, и $q.1(f, q.2(f))(argVal(a, q)) = und_{el}$, то $val(e, q) = \tau(f, (q.1, q))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in ContSym \cap PreSym$, то $val(e, q) = q.1(f, q.2(f))(argVal(a, q))$;
- если $match_{sym}(StSym \cup PreSym, e) = (f, a)$, и $f \in PreSym$, то $val(e, q) = \tau(f, (q.1, q))(argVal(a, q))$;
- $val(e, q) = und_{el}$.

7.5. Модификация символа. Пусть $::= \in At$. Элемент $[x ::=_{el} y]$ называется модификацией символа и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = [x ::=_{el} y]\ p$, тогда и только тогда, когда $t'.3 = t.3$, и выполнено первое подходящее свойство:

- если $match_{sym}(StSym, x) = (f, a)$, $f \in StSym$, $a' = argVal(a, (t.2, t.3))$, то $oDif(t'.2, t.2, \{A^f\}) = true$, $oDif(t'.2(A^f), t.2(A^f), \{a'\}) = true$, и $t'.2(A^f)(a') = val(y, (t.2, t.3))$;
- $t'.1 = p$ и $t'.2 = t.2$.

Для специального распространенного случай $[x ::= und]2/$ используется сокращение $[x ::=]1/$.

7.6. Условие безопасности. Пусть $assert \in At$. Элемент $/1[assert\ x]$ называется условием безопасности и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[assert\ x]\ p$, тогда и только тогда, когда $t'.2 = t.2$, $t'.3 = t.3$, и выполнено первое подходящее свойство:

- если $val(x, (t.2, t.3)) = true_{el}$, то $t'.1 = p$;
- $t'.1 = /[fail]/$.

7.7. Перезапуск. Пусть $restart \in At$. Элемент $/1[restart\ [x]]$ называется перезапуском и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1.1 = /1[restart\ [x]]$, тогда и только тогда, когда $t'.1 = x$, $t'.2(f, i)(p) = und_{el}$ для любых $f \in StSym \setminus \{count\}$, $i \in Int$

и $p \in El^{arity(f)}$, $t'.2(count, 0)() = 0_{el}$, $t'.2(count, i)() = und_{el}$ для любого $i \in Int \setminus \{0\}$, и $t'.3(f) = 1$ для любого $f \in StSym$.

7.8. Условное ветвление. Пусть $cases \in At$. Элемент вида

$$/1[cases\ /1[if\ e_1\ then_{el}\ p_1]\ ...\ /1[if\ e_n\ then_{el}\ p_n]]$$

или

$$/1[cases\ /1[if\ e_1\ then_{el}\ p_1]\ ...\ /1[if\ e_n\ then_{el}\ p_n]\ /1[else\ p]]$$

называется условным ветвлением и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[cases\ p''']\ p$, тогда и только тогда, когда $t.2' = t.2$, $t.3' = t.3$, и выполнено первое подходящее свойство:

- если $t.1 = /1[cases\ [if\ e\ then\ p']\ p'']\ p$, и $val(e, (t.2, t.3)) = true_{el}$, то $t.1' = p'\ p$;
- если $t.1 = /1[cases\ [if\ e\ then\ p']\ p'']\ p$, и $val(e, (t.2, t.3)) \neq true_{el}$, то $t.1' = /1[cases\ p'']\ p$;
- если $t.1 = /1[cases\ [else\ p']]\ p$, то $t.1' = p'\ p$;
- если $t.1 = /[cases]/\ p$, то $t.1' = backtrack_{el}$.

7.9. Ветвление по образцу. Выражение b вида $/1[if\ e\ var\ w\ then_{el}\ p]$, где $w \in VarSpec^*$, называется ветвью сопоставления с образцом, выражение e — образцом ветви b, переменные образца из w — переменными образца ветви b, и p — телом ветви b. Пусть $body(b)$ обозначает тело ветви b. Пусть $MatchBranch$ — множество всех ветвей сопоставления с образцом. Говорят, что $b_{\in MatchBranch}$ применима к e относительно $\sigma_{\in Sub}$, если $sub(e, \sigma) = e$, $dom(\sigma)$ — множество переменных образца ветви b, $\sigma(u) \in El$ для $u_{el} \in w$, и $\sigma(u) \in El^*$ для $/[+s\ u]/ \in w$.

Правило может быть применимо к одному и тому же элементу относительно разных подстановок, поэтому возможен конфликт при выборе подстановки, соответствующей этому элементу. Мы считаем, что определена функция $match_{sub} \in El \times MatchBranch \to Sub$, разрешающая этот конфликт, такая, что, если $match_{sub}(e, b) \neq und$, то b применима к e относительно $match_{sub}(e, b)$.

Пусть $matchCases \in At$. Пусть $z' = val(z'', (t.2, t.3))$ для $z = /1[++v\ z'']$ и $z' = z$ в противном случае. Элемент вида $/1[matchCases\ z\ b_1\ ...\ b_n]$ или $/1[matchCases\ z\ b_1\ ...\ b_n\ /1[else\ p']]$, где $z \in El$ и $b_i \in MatchBranch$, называется ветвлением по образцу и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[matchCases\ z\ p''']\ p$, тогда и только тогда, когда $t.2' = t.2$, $t.3' = t.3$, и выполнено первое подходящее свойство:

- если $t.1 = /1[matchCases\ z\ b\ p''']\ p$, и $match_{sub}(z', b) \neq und$, то $t.1' = match_{sub}(z', b)(body(b))\ p$;
- если $t.1 = /1[matchCases\ z\ b\ p''']\ p$, и $match_{sub}(z', b) = und$, то $t.1' = /1[matchCases\ p''']\ p$;
- если $t.1 = /1[matchCases\ z\ [else\ p']]\ p$, то $t.1' = p'\ p$;
- если $t.1 = /[matchCases]\ z/\ p$, то $t.1' = backtrack_{el}$.

Элемент z называется спецификатором сопоставляемого выражения.

7.10. Элементы управления контекстом. Пусть $getCont, setCont, newCont \in At$. Пусть $n > 0$, $f_1, ..., f_n \in StSym$, и F обозначает $/[f_1]/\ ...\ /[f_n]/$.

Элемент $/[getCont]/$ возвращает представление текущего контекста и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /[getCont]/\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{val}\}) = true$, и $v'_{val} = contRep(t.3)$.

Элемент $/1[getCont\ F]$ возвращает представление текущего контекста, суженное на $\{/[f_1]/, ..., /[f_n]/\}$, и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[getCont\ F]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{val}\}) = true$, и $v'_{val} = nar(contRep(t.3), \{/[f_1]/, ..., /[f_n]/\})$.

Элемент $/1[setCont\ x]$ устанавливает текущий контекст равным $val(x, (t.2, t.3))$ и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[setCont\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.2 = t.2$, и выполнено первое подходящее свойство:

- если $val(x, (t.2, t3)) = El_{fun}(g)$, и $g(/[f]/) \in Int_{el} \cup \{und_{el}\}$ для любого $f \in StSym$, то для любого $f \in StSym$ выполнено первое подходящее свойство:
  - если $g(/[f]/) \neq und_{el}$, то $t'.3(f) = (g(/[f]/))_{int}$;
  - $t'.3(f) = t.3(f)$;
- $t'.3 = t.3$.

Элемент $/[newCont]/$ меняет контекст всех символов из $StSym$ и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /[newCont]/\ p$, тогда и только тогда, когда $t'.1 = p$, $oDif(t'.2, t.2, \{A_{count}\}) = true$, $v'_{count}() = ((v_{count})_{int} + 1)_{el}$, $oDif(t'.3, t.3, StSym) = true$, и $t'.3(f) = (v'_{count}())_{int}$ для любого $f \in StSym$.

Элемент $/1[newCont/F]$ меняет контекст символов $f_1, ..., f_n$ и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[newCont/F]\ p$, тогда и только тогда, когда $t'.1 = p$, $oDif(t'.2, t.2, \{A_{count}\}) = true$, $v'_{count}() = ((v_{count}())_{int} + 1)_{el}$, $oDif(t'.3, t.3, \{f_1, ..., f_n\}) = true$, и $t'.3(f_i) = (v'_{count}())_{int}$ для любого $1 \leq i \leq n$.

7.11. Вычисление элемента. Пусть $elVal \in At$. Элемент $/1[elVal\ x]$ возвращает значение элемента x и имеет следующую семантику: $tr_{interp}(t,t') = true$, где $t.1 = /1[elVal\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{val}\}) = true$, и $v'_{val}() = val(x, (t.2, t.3))$.

Следующая группа правил переопределяет элементы $/1[assume\ x]$, $/1[modify\ x]$, $[x\ ::=_{el}\ y]$, $/[fail]/$, $/1[assert\ x]$, $/[getCont]/$, $/1[getCont\ p]$, $/1[setCont\ p]$, $/[newCont]/$, $/1[newCont\ p]$, $/[count{+}{+}]/$, $/[newEl]/$ и $/1[elVal\ e]$ для дедуктивных ПОСП с прямым прослеживанием. Поэтому эти элементы при таком их определении называются дедуктивными. В случае, если в дедуктивных ПОСП требуется использовать элементы с исходной семантикой (будем называть их операционными элементами), то вместо них используются элементы $/1[assume{*}\ x]$, $/1[modify{*}\ x]$, $[x\ ::={*}_{el}\ y]$, $/[fail{*}]/$, $/1[assert{*}\ x]$, $/[getCont{*}]/$, $/1[getCont{*}\ p]$, $/1[setCont{*}\ p]$, $/[newCont{*}]/$, $/1[newCont{*}\ p]$, $/[count{+}{+}{*}]/$, $/[newEl{*}]/$ и $/1[elVal{*}\ e]$.

7.12. Дедуктивное условие продолжения. Элемент $/1[assume\ x]$ называется дедуктивным условием продолжения и имеет следующую семантику: $tr_{interp}(t,t') = true$, где $t.1 = /1[assume\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{pre}\}) = true$, и $v_{pre}() = [v'_{pre}()\ and_{el}\ add_{ver}(x, t.2)]$.

Пусть $EvCont = St$. Пусть $i_f = (q(A_{st})(/[f]/))_{int}$ и $j_f = (q(A_{cont})(/[f]/))_{int}$. Функция $add_{ver} \in El \times EvCont \to El$ расставляет в первом аргументе версии у всех символов, входящих в $VerSym$, и определяется следующим образом (применяется первое подходящее правило):

- $add_{ver}(e_{\in im(El_{at}) \cup im(El_{fun})}, q) = e$;
- если $i, j \in Int$, $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, и $f \in ContFreePreSym \setminus VerSym$, то $add_{ver}(/3[ver\ i\ j\ p], q) = /3[addSeq_{ver}(p, q, f)]$;
- если $i, j \in Int$, и $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, то $add_{ver}(/3[ver\ i\ j\ p], q) = /3[ver\ i\ j\ addSeq_{ver}(p, q, f)]$;
- если $i, j \in Int$, то $add_{ver}(/3[ver\ i\ j\ p], q) = und_{el}$;
- если $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, и $f \in ContFreePreSym \setminus VerSym$, то $add_{ver}([p], q) = [addSeq_{ver}(p, q, f)]$;
- если $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, то $add_{ver}([p], q) = /3[ver\ i_f\ j_f\ addSeq_{ver}(p, q, f)]$;
- $add_{ver}(e, q) = und_{el}$.

Функция $addSeq_{ver} \in El^* \times EvCont \to El^*$ определяется следующим образом (применяется первое подходящее правило):

- $addSeq_{ver}(seq_{emp}, q, f) = seq_{emp}$;
- $addSeq_{ver}(/1[\text{-}v\ e]\ p, q, a\ f) = e\ addSeq_{ver}(p, q, f)$;
- $addSeq_{ver}(/1[+v\ e]\ p, q, a\ f) = /1[+v\ add_{ver(e,q)}]\ addSeq_{ver}(p, q, f)$;
- $addSeq_{ver}(e\ p, q, +v\ f) = add_{ver(e,q)}\ addSeq_{ver}(p, q, f)$;
- $addSeq_{ver}(e\ p, q, a\ f) = e\ addSeq_{ver}(p, q, f)$.

Пусть Frm обозначает свойство: $oDif(t'.2, t.2, \{A_{count}, A_{st}, A_{pre}\}) = true$ и $v'_{count}() = ((v_{count}())_{int} + 1)_{el}$.

7.13. Дедуктивное условие модификации. Элемент $/1[modify\ x]$ называется дедуктивным условием модификации и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[modify\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, и выполнено первое подходящее свойство:

- если $M = \{/[f]/ \mid f \in VerSym, match_{sym}(VerSym, e) = (f, a)$, и x содержит $/1[nextSt\ e]$ для некоторых e и $a\} \neq \emptyset$, то Frm истинно, и выполнены следующие свойства:
  - $oDif(v'_{st}, v_{st}, M) = true$;
  - если $e \in M$, то $v'_{st}(e) = v'_{count}()$;
  - $v'_{pre}() = [v_{pre}()\ and_{el}\ add_{ver}(x, (t.2, t'.2))]$;
- $t'.2 = t.2$.

Пусть $EvCont = St \times St$. Пусть $i_f = (q.1(A_{st})(/[f]/))_{int}$, $i'_f = (q.2(A_{st})(/[f]/))_{int}$ и $j_f = (q.1(A_{cont})(/[f]/))_{int}$. Функция $add_{ver} \in El^* \times EvCont \to El^*$ расставляет в первом аргументе версии у всех символов, входящих в $VerSym$, и определяется следующим образом (применяется первое подходящее правило):

- $add_{ver}(e_{\in im(El_{at}) \cup im(El_{fun})}, q) = e$;
- если $i, j \in Int$, $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, и $f \in ContFreePreSym \setminus VerSym$, то $add_{ver}(/3[ver\ i\ j\ p], q) = /3[addSeq_{ver}(p, q, f)]$;
- если $i, j \in Int$, и $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, то $add_{ver}(/3[ver\ i\ j\ p], q) = /3[ver\ i\ j\ addSeq_{ver}(p, q, f)]$;

- если $i, j \in Int$, то $add_{ver}(/3[ver\ i\ j\ p], q) = und_{el}$;

- если $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, и $f \in ContFreePreSym \setminus VerSym$, то $add_{ver}([p], q) = [addSeq_{ver}(p, q, f)]$;

- если $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, то $add_{ver}(/1[nextSt\ p], q) = /3[ver\ i'_f\ j_f\ addSeq_{ver}(p, q, f)]$;

- если $match_{sym}(VerSym \cup PreSym, [p]) = (f, a)$, то $add_{ver}([p], q) = /3[ver\ i_f\ j_f\ addSeq_{ver}(p, q, f)]$;

- $add_{ver}(e, q) = und_{el}$.

Пусть n', i_f и j_f обозначают $(v'_{count}())_{int}$, $(v_{st}(/[f]/))_{int}$ и $(v_{cont}(/[f]/))_{int}$, соответственно.

7.14. Дедуктивная модификация символа. Пусть $x = [p']$. Элемент $[x ::=_{el} y]$ называется дедуктивной модификацией символа и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = [x ::=_{el} y]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, и выполнено первое подходящее свойство:

- если $f \in VerSym$ и $match_{sym}(VerSym, x) = (f, a)$, то Frm истинно, и выполнены следующие свойства:
  - $oDif(v'_{st}, v_{st}, \{/[f/]\}) = true$;
  - a' — последовательность аргументов вызова x;
  - $t'.2(A_{st})(/[f/]) = n_{el}$, и $t'.2(A_{st})(/[f/]) = n'_{el}$;
  - если $arity(f) > 0$, то
    $v'_{pre}() = [v_{pre}()\ and_{el}\ /1[oDif\ /[ver\ n\ j_f\ sym\ f]/\ /[ver\ n'\ j_f\ sym\ f]\ a']\ and_{el}$
    $[add_{ver}(/1[nextSt\ p'], t.2, t'.2)\ =_{el}\ add_{ver}(y, t.2)]]$;
  - если $arity(f) = 0$, то
    $v'_{pre}() = [v_{pre}()\ and_{el}\ [add_{ver}([nextSt\ p'], (t.2, t'.2))\ =_{el}\ add_{ver}(y, t.2)]]$;

- $t'.2 = t.2$.

7.15. Операционно-дедуктивная модификация символа. Пусть $x = [p']$. Элемент $[x ::=**_{el}\ y]$ называется операционно-дедуктивной (или смешанной) модификацией символа и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = [x ::=**_{el}\ y]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, и выполнено первое подходящее свойство:

- если $f \in VerSym$, и $match_{sym}(VerSym, x) = (f, a)$, то Frm истинно, и выполнены следующие свойства:

– $oDif(v'_{st}, v_{st}, \{/[f/]\}) = true$;

– a' — последовательность аргументов вызова x;

– $t'.2(A_{st})(/[f/]) = n_{el}$, и $t'.2(A_{st})(/[f/]) = n'_{el}$;

– если $arity(f) > 0$, то
$v'_{pre}() = [v_{pre}()\ and_{el}\ /1[oDif\ /[ver\ n\ j_f\ sym\ f]/\ /[ver\ n'\ j_f\ sym\ f]\ a']\ and_{el}$
$[add_{ver}(/1[nextSt\ p'], t.2, t'.2)\ =_{el}\ val(y, (t.2, t.3))]]$;

– если $arity(f) = 0$, то
$v'_{pre}() = [v_{pre}()\ and_{el}\ [add_{ver}([nextSt\ p'], (t.2, t'.2))\ =_{el}\ val(y, (t.2, t.3))]]$.

- $t'.2 = t.2$.

Этот элемент позволяет передавать информацию из операционной семантики при дедуктивной верификации и тем самым комбинировать операционную семантику с дедуктивным выводом.

7.16. Дедуктивное небезопасное завершение. Пусть $A_{verCond}$ обозначает $(verCond, t.3(verCond))$. Элемент $/[fail]/$ называется называется дедуктивным небезопасным завершением и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1.1 = /[fail]/$, тогда и только тогда, когда $t'.1 = seq_{emp}$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{verCond}\}) = true$, $v'_{verCond}() = [p'\ /1[not\ v_{pre}()]]$, где $v_{verCond}() = [p']$.

7.17. Дедуктивное условие безопасности. Элемент $/1[assert\ x]$ называется дедуктивным условием безопасности и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[assert\ x]\ p$, тогда и только тогда, когда

$$t'.1 = /1[cases\ /1[if\ x\ then_{el}\ p]\ /1[else\ /[fail]/]],$$

$t'.2 = t.2$ и $t'.3 = t.3$.

7.18. Дедуктивные элементы управления контекстом. Пусть $n > 0$, $f_1, ..., f_n \in VerSym$ и F обозначает $/[f_1]/\ ...\ /[f_n]/$.

Элемент $/[getCont]/$ возвращает значение версионного контекста и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /[getCont]/\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{val}\}) = true$, и $v'_{val} = El_{fun}(v_{cont})$.

Элемент $/1[getCont\ F]$ возвращает значение версионного контекста, суженное на $\{/[f_1]/, ..., /[f_n]/\}$, и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[getCont\ F]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{val}\}) = true$, и $v'_{val} = nar(El_{fun}(v_{cont}), \{/[f_1]/, ..., /[f_n]/\})$.

Элемент $/1[setCont\ x]$ устанавливает версионный контекст равным $val(x, (t.2, t.3))$ и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[setCont\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{cont}\}) = true$, и выполнено первое подходящее свойство:

- если $val(x, (t.2, t3)) = El_{fun}(g)$ и $g(/[f]/) \in Int_{el} \cup \{und_{el}\}$ для любого $f \in VerSym$, то для любого $f \in SVerSym$ выполнено первое подходящее свойство:
  - если $g(/[f]/) \neq und_{el}$, то $v'_{cont}(/[f]/) = g(/[f]/)$;
  - $v'_{cont}(/[f]/) = v_{cont}(/[f]/)$;
- $v'_{cont} = v_{cont}$.

Элемент $/[newCont]/$ меняет версионный контекст всех символов из $VerSym$ и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /[newCont]/\ p$, тогда и только тогда, когда $t'.1 = p$, $oDif(t'.2, t.2, \{A_{count}, A_{cont}\}) = true$, $v'_{count}() = ((v_{count})_{int} + 1)_{el}$, $oDif(v'_{cont}, v_{cont}, VerSym) = true$, и $v'_{cont}(/[f]/) = v'_{count}()$ для $f \in VerSym$.

Элемент $/1[newCont/F]$ меняет контекст символов $f_1, ..., f_n$ и имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /1[newCont/F]\ p$, тогда и только тогда, когда $t'.1 = p$, $oDif(t'.2, t.2, \{A_{count, A_{cont}}\}) = true$, $v'_{count}() = ((v_{count}())_{int} + 1)_{el}$, $oDif(v'_{cont}, v_{cont}, \{/[f_1]/, ..., /[f_n]/\}) = true$, и $v'_{cont}(/[f_i]/) = v'_{count}()$ для $1 \leq i \leq n$.

7.19. Дедуктивный инкремент счетчика. Элемент $/[count{+}{+}]/$ имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /[count{+}{+}]/\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, Frm истинно, и выполнены следующие свойства:

- $oDif(v'_{st}, v_{st}, \{/[count]/\}) = true$;
- $v'_{count} = v'_{val} = n'_{el}$;
- $v'_{pre}() = [v_{pre}()\ and_{el}\ [/[ver\ n'\ j_{count}\ count]/\ =_{el}\ [/[ver\ i_{count}\ j_{count}\ count]/\ +\ 1]2/]\ and_{el}$ $[/[ver\ n'\ j_{val}\ val]/\ =_{el}\ /[ver\ n'\ j_{count}\ count]/]]$.

7.20. Дедуктивный генератор нумерованного элемента. Элемент $/[newEl]/$ имеет следующую семантику: $tr_{interp}(t, t') = true$, где $t.1 = /[newEl]/\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, Frm истинно, и выполнены следующие свойства:

- $oDif(v'_{st}, v_{st}, \{/[count]/, /[val]/\}) = true$;
- $v'_{count} = v'_{val} = n'_{el}$;
- $v'_{pre}() =$
  $[v_{pre}()\ and_{el}$
  $[/[ver\ n'\ j_{count}\ count]/\ =_{el}\ [/[ver\ i_{count}\ j_{count}\ count]/\ +\ 1]2/\ and_{el}$
  $[/[ver\ n'\ j_{val}\ val]/\ =_{el}\ /1[el\ /[ver\ n'\ j_{count}\ count]/]]]$.

7.21. Декларативное вычисление элемента. Элемент $/1[elVal\ x]$ имеет следующую семантику: $tr_{interp}(t,t') = true$, где $t.1 = /1[elVal\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, Frm истинно, и выполнены следующие свойства:

- $oDif(v'_{st}, v_{st}, \{/[val]/\}) = true$;
- $v'_{val} = n'_{el}$;
- $v'_{pre}() =\ [v_{pre}()\ and_{el}\ [/[ver\ n'\ j_{val}\ val]/\ =_{el}\ add_{ver}(x, t.2)]]$.

7.22. Элементы управления предусловием. Пусть $setPre \in At$. Элемент $/1[setPre\ x]$ имеет следующую семантику: $tr_{interp}(t,t') = true$, где $t.1 = /1[setPre\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, $oDif(t'.2, t.2, \{A_{pre}\}) = true$, и $v'_{pre}() = add_{ver}(x, t.2)$.

Пусть $newPre \in At$. Элемент $/1[newPre\ x]$ имеет следующую семантику: $tr_{interp}(t,t') = true$, где $t.1 = /1[newPre\ x]\ p$, тогда и только тогда, когда $t'.1 = p$, $t'.3 = t.3$, и выполнены следующие условия:

- $oDif(t'.2, t.2, \{A_{count}, A_{st}, A_{cont}, A_{pre}, A_{val}\}) = true$;
- $v'_{count}() = ((v_{count}())_{int} + 1)_{el}$;
- $oDif(v'_{st}, v_{st}, \{/[f]/ \mid f \in VerSym\}) = true$;
- $oDif(v'_{cont}, v_{cont}, \{/[f]/ \mid f \in VerSym\}) = true$;
- если $f \in VerSym$, то $v'_{st}(/[f]/) = v'_{cont}(/[f]/) = v'_{count}()$;
- $v'_{val}() = v'_{count}()$;
- $v'_{pre}() = add_{ver}(x, t'.2)$.

8. Язык предметно-ориентированных систем переходов Atoment. Язык Atoment предназначен для спецификации ПОСП. Его предыдущие версии рассматривались в [1, 4, 6]. Базовыми конструкциями языка Atoment, с помощью которых описываются элементы объектной системы ПОСП, являются атомы и выражения. Семантика конструкций языка Atoment определяется семантикой соответствующих элементов объектной модели ПОСП.

8.1. Служебные символы. К служебным символам языка Atoment относятся пробельные символы (пробел, переход на новую строку и т. п.), скобки `(` и `)`, символ `"`.

8.2. Атомы. Множество LAt атомов языка Atoment определяется следующим образом:

- если u — произвольная последовательность Unicode-символов, за исключением служебных символов, то $u \in LAt$;
- если u — произвольная последовательность Unicode-символов, в которой каждому вхождению символа `"` предшествует экранирующий символ `\` (если требуется вставить `\`, то он, как обычно, удваивается), то `"`u`"`$\in LAt$.

8.3. Выражения. Пусть t_i — последовательности пробельных символов. Множество Exp выражений языка Atoment определяется следующим образом:

- если $a \in LAt$, то $a \in Exp$;
- если $e_1, ..., e_n \in Exp$ — выражения, то $(t_0 e_1 t_1 ... e_n t_n) \in Exp$. Последовательность t_i может быть пустой только в случае, если она граничит хотя бы с одной скобкой. Выражение вида (t_0) называется пустым выражением.

Далее для простоты будем опускать t_i в определениях, считая, что пробел соответствует последовательности из одного или более пробельных символов. Так, $(t_0 e_1 t_1 ... e_n t_n)$ в упрощенном виде записывается как `(`e_1 ... e_n`)`.

Опишем соответствие между базовыми конструкциями языка Atoment и элементами объектной модели ПОСП. Пусть $e_1, ..., e_n \in Exp$ и $a, a_1, ..., a_n \in LAt$.

8.4. Соответствие атомов. Мы считаем, что множества атомов LAt и At языка Atoment и объектной модели ПОСП, соответственно, совпадают.

8.5. Соответствие выражений и элементов. Определим отображение $red \in Exp \to El$ выражений языка Atoment в элементы объектной модели ПОСП:

- $red(a) = El_{at}(a)$;
- если El_a — конструктор выражений, то $red($`(:`a e_1 ... e_n`)`$) = El_a(red(e_1)...red(e_n))$;
- $red($`(`e_1 ... e_n`)`$) = [e_1...e_n]$. В частности, $red($`()`$) = [\,]$.

Например, выражение `(assume (A and B))` соответствует элементу $/1[assume\ /[A\ and\ B]/]$, а выражение `(A (:seq B C) (:at D))` — элементу $/1[A\ /[B\ C]/\ D]/1$.

8.6. Правила переходов. Правило перехода $Rul(x, y, z)$ описывается выражением `(if` x' `var` y' `then` z'`)`, где последовательности x', y' и z' специфицируют x, y и z, соответственно. Признаком завершения последовательности x' является атом `var` (если правило содержит переменные образца) или атом `then` (если правило не содержит переменных образца). Признаком завершения последовательности y' является атом `then`.

8.7. Условные правила переходов. Условное правило перехода $Rul_{cond}(x, y, z, u)$ описывается выражением (`if` x' `var` y' `where` u' `then` z'), где последовательности x', y', u' и атом z' специфицируют x, y, u, z, соответственно. Признаком завершения последовательности y' является атом `where`.

8.8. Правила переходов с переменными истории. Правило перехода с переменными истории $Rul_{hvar}(x, y, z, u)$ описывается выражением (`if` x' `var` y' `hvar` u' `then` z'), где последовательности x', y', z' и u' специфицируют x, y, z и u, соответственно. Признаком завершения последовательности x' является атом `var` (если правило содержит переменные образца), атом `hvar` (если правило не содержит переменных образца и содержит переменные истории) или атом `then`. Признаком завершения последовательности y' является атом `hvar` (если правило содержит переменные истории) или атом `then` (если правило не содержит переменных истории). Признаком завершения последовательности u' является атом `then`.

8.9. Условные правила переходов с переменными истории. Условное правило перехода с переменными истории $Rul_{cond+hvar}(x, y, z, u, v)$ описывается выражением (`if` x' `var` y' `where` u' `hvar` v' `then` z'), где выражения x', y', z', u' и v' специфицируют x, y, z, u и v, соответственно. Признаком завершения последовательности y' является атом `where`.

8.10. Онтологические правила переходов. Онтологическое правило перехода $Rul_{ont}(x, y)$ описывается выражением (`if` x' `then` y'), где выражение x' и последовательность y' специфицируют x и y, соответственно.

8.11. ПОСП. ПОСП определяется как выражение вида (`dsts` a `kind` (b) `StSym` (c_1) `PreSym` (c_2) `ContFreePreSym` (c_3) `VerSym` (c_4) `OntSym` (c_5) `InstSym` (c_6) `rules` d), где a — атом, обозначающий имя ПОСП, b — последовательность атомов, описывающих вид ПОСП, $c_1, ..., c_6$ — последовательности выражений вида $/[f]/$, где $f \in Sym$, представляющих символы состояния, предопределенные символы, свободные от контекста предопределенные символы, версионные символы, онтологические символы и символы экземпляризации, соответственно; d — последовательность правил переходов. Последовательность b может включать атомы `backward` и `fullBacktrack`, обозначающие ПОСП с обратным проходом и ПОСП с полным перебором вариантов, соответственно.

ПОСП по умолчанию является условной, неонтологической, неверсионной ПОСП прямого прохода с частично интерпретированными телами правил с возвращаемым значением, с инкрементом счетчика, с генерацией нумерованных элементов, с переменными истории, с управляемым бэктрекингом и с сопоставлением с образцом на последовательностях. Множество предопределенных символов является общим для всех ПОСП, описываемых в Atoment.

9. Заключение. ПОСП — системы переходов специального вида, предназначенные для определения предметно-ориентированных языков, используемых для решения задач разработки семантики языков программирования и проектирования, спецификации, прототипирования и верификации программных систем. ПОСП составляют основу комплексного подхода к решению указанных задач.

В этой работе — первой из цикла работ, посвященных ПОСП, — описаны объектная модель и язык ПОСП. В следующих работах цикла будут рассмотрены примеры применения ПОСП к решению упомянутых выше задач.

Список литературы

1. Ануреев И.С. Типовые примеры использования языка Atoment // Моделирование и анализ информационных систем. 2011. Т. 18. №4. С. 7–20.
2. Непомнящий В.А., Ануреев И.С., Атучин М.М., Марьясов И.В., Петров А.А., Промский А.В. Верификация C-программ в мультиязыковой системе Спектр // Моделирование и анализ информационных систем. 2010. Т. 17. №4, С. 88–100.
3. Anureev I.S. Integrated approach to analysis and verification of imperative programs // Joint NCC&IIS Bulletin, Series Computer Science. 2011. Vol. 32. P. 1–18.
4. Anureev I.S. Introduction to the Atoment language // Joint NCC&IIS Bulletin, Series Computer Science. 2010. Vol. 31. P. 1–16.
5. Anureev I.S., Maryasov I.V., Nepomniaschy V.A. Two-level Mixed Verification Method of C-light Programs in Terms of Safety Logic. Joint NCC&IIS Bulletin, Series Computer Science. 2012. Vol. 34. P. 23–42.
6. Anureev I.S. Program Specific Transition Systems. Joint NCC&IIS Bulletin, Series Computer Science. 2012. Vol. 34. P. 1-21.
7. AsmL: The Abstract State Machine Language. Reference Manual, 2002. http://research.microsoft.com/en-us/projects/asml/
8. Gurevich Y. Abstract State Machines: An Overview of the Project. Foundations of Information and Knowledge Systems (FoIKS): Proc. Third Internat. Symp. Lect. Notes Comput. Sci. 2004. Vol. 2942. P. 6–13.
9. Matthias Anlauff. XasM — An Extensible, Component-Based Abstract State Machines Language. http://xasm.sourceforge.net/XasmAnl00/XasmAnl00.html
10. Parnas D.L. Really Rethinking Formal Methods. Computer. IEEE Computer Society. 2010. Vol. 43 (1). P. 28–34.

UDK: 004.05

Title: Domain-specific transition systems: object model and language

Author(s):

Igor S. Anureev (A.P. Ershov Institute of Informatics Systems)

Abstract: This paper presents the object model and the language of domain-specific transition systems, a new formalism designed for specification and approbation of formal methods which ensure software reliability.

Keywords: domain-specific transition systems, semantics, verification, ontology