ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆԻ ԳԻՏԱԿԱՆ ՏԵՂԵԿԱԳԻՐ
УЧЕНЫЕ ЗАПИСКИ ЕРЕВАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Բնական գիտություններ 3, 2008 Естественные науки

Информатика

УДК 519.682.1

Р. Ю. АКОПЯН

О ПРОЦЕДУРНЫХ СЕМАНТИКАХ СТРОГО ТИПИЗИРОВАННЫХ ФУНКЦИОНАЛЬНЫХ ПРОГРАММ

Статья посвящена процедурным семантикам строго типизированных функциональных программ, основанным на алгоритмах интерпретации, которые используют три операции: подстановку, одношаговую β-редукцию и одношаговую δ-редукцию. Доказывается, что процедурная семантика, основанная на любом из этих алгоритмов интерпретации, непротиворечива. Доказывается также несравнимость процедурных семантик, использующих некоторые алгоритмы интерпретации.

Используемые определения и результаты. Определения данного параграфа заимствованы нами из работ [1–3]. Пусть M есть частично упорядоченное множество, содержащее наименьший элемент $\perp$, и каждый элемент из M сравним только с $\perp$ и с самим собой.

Множество типов $Types$ определим следующим образом:

1. $M \in Types$;

2. если $\beta, \alpha_1, ..., \alpha_k \in Types\ (k > 0)$, то множество всех монотонных отображений из $\alpha_1 \times ... \times \alpha_k$ в β (обозначим $[\alpha_1 \times ... \times \alpha_k \to \beta]$) принадлежит $Types$;

3. других типов, кроме определенных пунктами 1–2, нет.

Пусть $\alpha \in Types$. Порядком типа α будет натуральное число (обозначим $ord(\alpha)$), определяемое следующим образом:

1. если $\alpha = M$, то $ord(\alpha) = 0$;

2. если $\alpha = [\alpha_1 \times ... \times \alpha_k \to \beta]$, где $\beta, \alpha_1, ..., \alpha_k \in Types\ (k > 0)$, то $ord(\alpha) = \max(ord(\alpha_1), ..., ord(\alpha_k), ord(\beta)) + 1$.

Будем считать, что для каждого $\alpha \in Types$ мы имеем счетное множество V_α переменных типа α. Если переменная $x \in V_\alpha$ и константа $c \in \alpha$, то $ord(x) = ord(c) = ord(\alpha)$. Пусть $V = \bigcup_{\alpha \in Types} V_\alpha$. Определим множество термов $\Lambda = \bigcup_{\alpha \in Types} \Lambda_\alpha$, где Λ_α – множество термов типа α, следующим образом:

1. если $c \in \alpha,\ \alpha \in Types,$ то $c \in \Lambda_\alpha$;

2. если $x \in V_\alpha,\ \alpha \in Types,$ то $x \in \Lambda_\alpha$;

3. если $t \in \Lambda_{[\alpha_1 \times ... \times \alpha_k \to \beta]},\ t_i \in \Lambda_{\alpha_i},\ \beta, \alpha_i \in Types,\ i = 1,...,k\ (k > 0)$, то $t(t_1,...,t_k) \in \Lambda_\beta$ ($t_1,...,t_k$ назовем областью действия аппликатора t);

4. если $t \in \Lambda_\beta,\ x_i \in V_{\alpha_i},\ \beta, \alpha_i \in Types,\ i \neq j \Rightarrow x_i \neq x_j,\ i, j = 1,...,k\ (k > 0)$, то $\lambda x_1 ... x_k[t] \in \Lambda_{[\alpha_1 \times ... \times \alpha_k \to \beta]}$ (t назовем областью действия абстрактора $\lambda x_1 ... x_k$);

5. никаких других термов, кроме определенных пунктами 1– 4, нет.

Некоторое вхождение переменной x в терме t называется связанным, если оно либо принадлежит некоторому абстрактору, либо находится в области действия абстрактора, использующего x. Несвязанное вхождение переменной x в терме t назовем свободным. Будем говорить, что переменная x свободна в терме t, если x имеет хотя бы одно свободное вхождение в терме t. Множество всех свободных переменных терма t обозначим $FV(t)$. Термы t_1, t_2 назовем конгруэнтными ($t_1 \equiv t_2$), если терм t_2 получается из терма t_1 серией замен связанных переменных.

Пусть $t \in \Lambda,\ t_i \in \Lambda_{\alpha_i},\ x_i \in V_{\alpha_i},\ \alpha_i \in Types,\ i \neq j \Rightarrow x_i \neq x_j,\ i, j = 1,...,k\ (k > 0)$. Тогда одновременную подстановку термов $t_1,...,t_k$ в терм t вместо некоторых свободных вхождений переменных $x_1,...,x_k$ соответственно назовем допустимой, если ни одно из рассматриваемых свободных вхождений переменной x_i не находится в области действия абстрактора, использующего некоторую свободную переменную терма $t_i, i = 1,...,k$. Терм t, в котором зафиксированы некоторые свободные вхождения переменных $x_1,...,x_k$ слева направо, обозначим $t < x_{i_1},\ldots,x_{i_s} >,\ x_{i_j} \in \{x_1,\ldots,x_k\},\ j = 1,\ldots,s.$ Терм, полученный в результате допустимой подстановки термов $t_{i_1},...,t_{i_s}$ в терм $t < x_{i_1},\ldots,x_{i_s} >$ вместо фиксированных свободных вхождений $x_{i_1},\ldots,x_{i_s}$ соответственно, обозначим $t < t_{i_1},\ldots,t_{i_s} >$. Терм, полученный в результате допустимой подстановки термов $t_1,...,t_k$ в терм t вместо всех свободных вхождений переменных $x_1,...,x_k$ соответственно, условимся обозначить $t\{t_1 / x_1,\ldots,t_k / x_k\}$ (коротко $t\{\overline{t} / \overline{x}\}$).

Каждому терму $t \in \Lambda_\alpha$ сопоставим константу $Val_{\overline{y}_0}(t) \in \alpha, \alpha \in Types,$ где $FV(t) \subset \{y_1,\ldots,y_n\},\ y_i \in V_{\alpha_i},\ \overline{y}_0 =< y_1^0,\ldots,y_n^0 >,\ y_i^0 \in \alpha_i,\ \alpha_i \in Types,$ $i = 1,\ldots,n\ (n \geq 0),$ следующим образом:

1. если $t \equiv c,\ c \in \alpha$, то $Val_{\overline{y}_0}(t) = c$;

2. если $t \equiv x,\ x \in V_\alpha$, то $Val_{\overline{y}_0}(t) = y_i^0$, где $x \equiv y_i,\ i = 1,...,n\ (n > 0)$;

3. если $t \equiv \tau(t_1,...,t_k) \in \Lambda_\alpha,\ \tau \in \Lambda_{[\alpha_1 \times ... \times \alpha_k \to \alpha]},\ t_i \in \Lambda_{\alpha_i},\ \alpha_i \in Types,\ i = 1,...,k$ $(k \geq 1),$ то $Val_{\overline{y}_0}(\tau(t_1,...,t_k)) = Val_{\overline{y}_0}(\tau)(Val_{\overline{y}_0}(t_1),\ldots,Val_{\overline{y}_0}(t_k))$;

4. если $t \equiv \lambda x_1 ... x_k[\tau] \in \Lambda_{[\alpha_1 \times ... \times \alpha_k \to \beta]}$, $\tau \in \Lambda_\beta$, $x_i \in V_{\alpha_i}$, $\beta, \alpha_i \in Types$, $i = 1,...,k$ $(k \geq 1)$, то $Val_{\bar{y}_0}(\lambda x_1 ... x_k[\tau]) : \alpha_1 \times \ldots \times \alpha_k \to \beta$, и для всяких $\bar{x}_0 =< x_1^0, \ldots, x_k^0 >$, $x_i^0 \in \alpha_i$, $i = 1, \ldots k$, имеем $Val_{\bar{y}_0}(\lambda x_1 ... x_k[\tau])(\bar{x}_0) = Val_{\bar{x}_0 \bar{y}_0'}(\tau)$, где $FV(\lambda x_1 ... x_k[\tau]) = \{y_{i_1}, \ldots y_{i_m}\}$, $y_0' =< y_{i_1}^0, \ldots, y_{i_m}^0 >$.

Пусть t_1, t_2 есть термы, и $FV(t_1) \cup FV(t_2) = \{y_1, \ldots, y_n\}$, $y_i \in V_{\alpha_i}$, $\alpha_i \in Types$, $i = 1, \ldots n$ $(n \geq 0)$. Термы t_1, t_2 назовем эквивалентными (обозначим $t_1 \sim t_2$), если для любого $\bar{y}_0 =< y_1^0, \ldots, y_n^0 >$, где $y_i^0 \in \alpha_i, i = 1, \ldots, n$, имеем $Val_{\bar{y}_0}(t_1) = Val_{\bar{y}_0}(t_2)$.

Пусть $t \in \Lambda_\alpha$, $\alpha \in Types, FV(t) \subseteq \{y_1, \ldots, y_n\}$, $y_i \in V_{\alpha_i}$, $\alpha_i \in Types$, $i = 1,...,n$ $(n \geq 0)$. Терм t назовем константным термом со значением $b \in \alpha$, если для любого $\bar{y}_0 =< y_1^0, \ldots, y_n^0 >$, $y_i^0 \in \alpha_i, i = 1, \ldots, n$, имеем $Val_{\bar{y}_0}(t) = b$.

Рассмотрим систему уравнений P:

$$\begin{cases} F_1 = \tau_1, \\ \quad \ldots \\ F_n = \tau_n, \end{cases} \qquad (1)$$

где $F_i \in V_{\alpha_i}$, $i \neq j \Rightarrow F_i \not\equiv F_j$, $\tau_i \in \Lambda_{\alpha_i}$, $FV(\tau_i) \subset \{F_1, \ldots, F_n\}$, $\alpha_i \in Types$, $i, j = 1, \ldots, n$ $(n \geq 1)$. Отображение $\Psi_P : \alpha_1 \times \ldots \times \alpha_n \to \alpha_1 \times \ldots \times \alpha_n$, где $\Psi_P(\bar{g}) =< Val_{\bar{g}}(\tau_1), \ldots, Val_{\bar{g}}(\tau_n) >$ и $\bar{g} \in \alpha_1 \times \ldots \times \alpha_n$, назовем отображением, соответствующим системе уравнений (1). Через $(\bar{g})_i$, $i = 1, \ldots, n$, обозначим i-ю компоненту вектора $\bar{g}$. Будем говорить, что $\bar{f} \in \alpha_1 \times \ldots \times \alpha_n$ является решением системы уравнений (1), если $\Psi_P(\bar{f}) = \bar{f}$. В работе [1] доказывается, что каждая система уравнений (1) имеет наименьшее решение. Система уравнений (1) называется программой, если все используемые ею константы имеют порядок ≤ 1; константы порядка 1 являются вычислимыми функциями и $F_1 \in V_{[M^k \to M]}$ $(k \geq 1)$. Уравнение $F_1 = \tau_1$ называется главным уравнением программы P, а функция $f_P = (\bar{f})_1$, где $\bar{f}$ – наименьшее решение программы P, – семантикой неподвижной точки программы P.

Множество $Fix(P)$, соответствующее семантике неподвижной точки программы P, определятся следующим образом:

$$Fix(P) = \{< \bar{m}, m > | \, f_P(\bar{m}) = m, \ \bar{m} \in M^k, \ m \in M\}.$$

Напомним понятие β-редукции. $\beta = \{< \lambda x_1 ... x_k[t](t_1, \ldots t_k), \ t\{t_1 / x_1, \ldots, t_k / x_k\} > | \, x_i \in V_{\alpha_i}, \ t_i \in \Lambda_{\alpha_i}, \ \alpha_i \in Types, \ i = 1, \ldots, k \ (k \geq 1)\}$. Терм $\lambda x_1 ... x_k[t](t_1, \ldots t_k)$ есть β-редекс, а $t\{t_1 / x_1, \ldots, t_k / x_k\}$ – его свертка. Скажем, что терм t' получен из терма t посредством одношаговой β-редукции $(t \to_\beta t')$, если $t \equiv t_\tau, t' \equiv t_{\tau'}$ и τ есть β-редекс, а τ' – его свертка, где через t_τ обозначен

терм t с некоторым фиксированным вхождением подтерма τ, а через $t_{\tau'}$ – терм, полученный в результате замены данного вхождения подтерма τ на терм τ'. Скажем, что терм t' получен из терма t посредством β-редукции $(t \to\to_\beta t')$, если либо $t \equiv t'$, либо существует конечная последовательность термов $t_1,\ldots,t_k$ $(k \geq 0)$ такая, что $t \to_\beta t_1 \to_\beta \ldots \to_\beta t_k \to_\beta t'$. Далее будем рассматривать термы, которые не используют константы, порядок которых ≥ 2.

Определим понятие δ-редукции. Пусть $\Delta = \{< f(t_1,\ldots t_k), \tau > | f$ – константа, $\tau, t_1,\ldots t_k$ $(k > 0)$ – термы и либо τ – константа и $f(t_1,\ldots t_k)$ является константным термом со значением τ, либо τ – собственный подтерм терма $f(t_1,\ldots t_k)$ и $f(t_1,\ldots t_k) \sim \tau\}$.

Любое рекурсивное подмножество δ множества Δ назовем понятием δ-редукции. Если $<\tau,\tau'> \in \delta$, то τ называется δ-редексом, а τ' – его сверткой. Скажем, что терм t' получен из терма t посредством одношаговой δ-редукции $(t \to_\delta t')$, если $t \equiv t_\tau, t' \equiv t_{\tau'}$ и τ есть δ-редекс, а τ' – его свертка. Скажем, что терм t' получен из терма t посредством δ-редукции $(t \to\to_\delta t')$, если либо $t \equiv t'$, либо существует конечная последовательность термов $t_1,\ldots,t_k$ $(k \geq 0)$ такая, что $t \to_\delta t_1 \to_\delta \ldots \to_\delta t_k \to_\delta t'$.

Понятие $\beta \cup \delta$-редукции, обозначим через $\beta\delta$. Условимся $\beta\delta$-редукцию называть просто редукцией, одношаговую $\beta\delta$-редукцию обозначать $\to$, а $\beta\delta$-редукцию – $\to\to$.

Скажем что терм t есть нормальная форма, если в нем нет подтерма, который является $\beta\delta$-редексом. Множество нормальных форм обозначим NF.

Обозначим через Δ_0 следующее подмножество множества Δ:
$\Delta_0 = \{< f(t_1,\ldots t_k), \tau > | < f(t_1,\ldots t_k), \tau > \in \Delta$, где τ либо константа, либо $\tau \equiv t_i \ (1 \leq i \leq k)\}$.

Понятие δ-редукции назовем естественным, если:

1. $\delta \subset \Delta_0$;
2. δ – однозначное отношение, т.е. если $<t,\tau_1> \in \delta$ и $<t,\tau_2> \in \delta$, то $\tau_1 \equiv \tau_2$, где $t, \tau_1, \tau_2 \in \Lambda$;
3. для любого константного терма $f(t_1,\ldots t_k)$ со значением $m \in M$ $f(t_1,\ldots t_k) \to\to m$, где f – константа, $t_1,\ldots t_k \in \Lambda$.

Будем говорить, что понятие δ-редукции обладает свойством подстановочности, если из условия $< f(t_1,\ldots t_k), t_j > \in \delta \ (1 \leq j \leq k)$, где f – константа, $t_1,\ldots t_k \in \Lambda$ и $f(t_1,\ldots t_k)$ не является константным термом, следует, что для любой допустимой подстановки $\{\tau_1 / x_1,\ldots,\tau_k / x_k\}$, где $\tau_i \in \Lambda_{\alpha_i}$, $x_i \in V_{\alpha_i}$, $\alpha_i \in Types$, $i \neq j \Rightarrow x_i \not\equiv x_j$, $i, j = 1,\ldots,m$ $(m > 0)$, существуют термы $t'_1,\ldots t'_k$ такие, что $t_1\{\overline{\tau} / \overline{x}\} \to\to t'_1, \ldots, t_k\{\overline{\tau} / \overline{x}\} \to\to t'_k$ и $< f(t'_1,\ldots t'_k), t'_j > \in \delta$. Будем

говорить, что понятие δ-редукции обладает свойством наследуемости, если из того, что $< f(t_1,\dots t_k), t_j > \in \delta\ (1 \le j \le k)$, где f – константа, $t_1,\dots t_k \in \Lambda$ и $f(t_1,\dots t_k)$ не является константным термом, $t_i \equiv \tau_r$ для некоторого $i\ (1 \le i \le k)$, где r – редекс, следует, что существуют термы $t_1',\dots t_k'$ такие, что $t_1 \to\to t_1', \dots, \tau_{r'} \to\to t_i', \dots, t_k \to\to t_k'$ и $< f(t_1',\dots t_k'), t_j' > \in \delta$, где r' – свертка редекса r. Если понятие δ-редукции обладает свойством подстановочности и наследуемости, то будем говорить, что оно обладает ПН-свойством. В работе [3] доказывается, что, если δ – некоторое естественное понятие δ-редукции, для любого терма t имеет место следующее: $t \to\to t', t \to\to t'',\ t', t'' \in NF \Rightarrow t' \equiv t''$ в том и только том случае, если понятие δ-редукции обладает ПН-свойством. Далее будем рассматривать только такие понятия δ-редукции, которые являются естественными и обладают ПН-свойством.

Алгоритмы интерпретации. Зафиксируем некоторое понятие δ-редукции. Введем понятие алгоритма интерпретации A. Алгоритм интерпретации A, получив на вход программу P вида (1) и терм $F_1(\bar{m})$, где $\bar{m} \in M^k$, либо останавливается с результатом $m \in M$, либо функционирует бесконечно. Алгоритм интерпретации A использует 3 вида операций:

1. подстановка термов $\tau_1,\dots,\tau_n$ вместо некоторых свободных вхождений переменных $F_1,\dots,F_n$ соответственно;

2. одношаговая β-редукция;

3. одношаговая δ-редукция.

Пусть A – некоторый алгоритм интерпретации и P – программа. Множество $Proc_A(P)$, соответствующее процедурной семантике, использующей алгоритм интерпретации A, определим следующим образом: $Proc_A(P) = \{< \bar{m}, m > |$ алгоритм A на P и $F_1(\bar{m})$ останавливается с результатом $m \neq \perp$, где $\bar{m} \in M^k$, $m \in M\}$. Скажем, что процедурная семантика, использующая алгоритм интерпретации A, непротиворечива, если $Proc_A(P) \subset Fix(P)$ для любой программы P.

Теорема 1 (о непротиворечивости). Процедурная семантика, использующая любой алгоритм интерпретации A, непротиворечива.

Перед тем как перейти к непосредственному доказательству теоремы 1, докажем лемму 1.

Лемма 1. Пусть A – алгоритм интерпретации, P – программа, термы t, t' такие, что терм t' получен из терма t в результате применения одной из трех операций, используемых алгоритмом интерпретации A. Тогда $Val_{\bar{f}}(t) = Val_{\bar{f}}(t')$, где $\bar{f}$ – наименьшее решение программы P.

Доказательство.

1. Если терм t' получен в результате замены некоторых свободных вхождений переменных $F_1,\dots,F_n$ в терме t на термы $\tau_1,\dots,\tau_n$, то доказательство леммы 1 следует из леммы 3.2.1 работы [3].

2. Если $t \to_\beta t'$, то доказательство леммы 1 следует из теоремы 2.1.1 (о редукции) работы [3].

3. Если $t \to_\delta t'$, то доказательство леммы 1 следует из теоремы 2.1.1 (о редукции) работы [3].

Лемма 1 доказана.

Доказательство теоремы 1. Пусть $<\bar{m}, m> \in Proc_A(P)$. Тогда существуют термы $t_0,...,t_k \in \Lambda$ такие, что $t_0 \equiv F_1(\bar{m})$, $t_k \equiv m$, и терм t_{i+1} получен из терма t_i, $i = 0,...,k-1\,(k \geq 1)$, посредством применения одной из трех операций, используемых алгоритмом интерпретации A. Покажем что $<\bar{m}, m> \in Fix(P)$. Согластно лемме 1, имеем $Val_{\bar{f}}(t_0) = Val_{\bar{f}}(t_1) = ... = Val_{\bar{f}}(t_k) = m$, и так как $Val_{\bar{f}}(t_0) = Val_{\bar{f}}(F_1(\bar{m})) = f_P(\bar{m})$, то $f_P(\bar{m}) = m$, следовательно, $<\bar{m}, m> \in Fix(P)$.

Теорема 1 доказана.

Далее рассмотрим следующие алгоритмы интерпретации.

Алгоритм ACT (активный алгоритм).

Вход: программа P, терм t.

Выход: терм ACT(P, t), если ACT определен для P и t.

1. если $t \in NF$ и $FV(t) \cap \{F_1,\ldots,F_n\} = \emptyset$, то t, иначе перейти к шагу 2;

2. если $t \equiv t < F_i >$, где F_i – самое левое вхождение переменой из множества $\{F_1,\ldots,F_n\}$ в терм t и данное вхождение переменной F_i находится левее самого левого редекса терма t, то ACT$(P, t < \tau_i >)$, иначе перейти к шагу 3;

3. если $t \equiv t_{\lambda x_1 \ldots x_k [\tau](t_1,\ldots,t_k)}$ и $\lambda x_1 \ldots x_k [\tau](t_1,\ldots,t_k)$ – самый левый редекс терма t, то ACT$(P, t_{\tau\{ACT(P,t_1)/x_1,\ldots,ACT(P,t_k)/x_k\}})$; иначе перейти к шагу 4;

4. если $t \equiv t_\tau$ и τ – самый левый δ-редекс, то ACT$(P, t_{\tau'})$, где τ' – свертка редекса τ.

Алгоритм PAS (пассивный алгоритм).

Вход: программа P, терм t.

Выход: терм PAS(P, t), если PAS определен для P и t.

1. если $t \in NF$ и $FV(t) \cap \{F_1,\ldots,F_n\} = \emptyset$, то t, иначе перейти к шагу 2;

2. если $t \equiv t < F_i >$, где F_i – самое левое вхождение переменой из множества $\{F_1,\ldots,F_n\}$ в терм t и данное вхождение переменной F_i находится левее самого левого редекса терма t, то PAS$(P, t < \tau_i >)$, иначе перейти к шагу 3;

3. если $t \equiv t_{\lambda x_1 \ldots x_k [\tau](t_1,\ldots,t_k)}$ и $\lambda x_1 \ldots x_k [\tau](t_1,\ldots,t_k)$ – самый левый редекс терма t, то PAS$(P, t_{\tau\{t_1/x_1,\ldots,t_k/x_k\}})$, иначе перейти к шагу 4;

4. если $t \equiv t_\tau$ и τ – самый левый δ-редекс, то PAS$(P, t_{\tau'})$, где τ' – свертка редекса τ.

Рассмотрим три алгоритма интерпретации, которые основаны на подстановках и редукции к нормальной форме. Свободное вхождение перемен-

ной F в терме t назовем внутренним, если оно не входит в аппликатор τ, область действия которого содержит свободное вхождение некоторой переменной. Свободное вхождение переменной F в терме t назовем внешним, если оно не входит в область действия аппликатора, содержащего свободное вхождение некоторой переменной.

Алгоритм LIS (основанный на самой левой внутренней замене).

Вход: программа P, терм t.

Выход: терм $\mathrm{LIS}(P,t)$, если LIS определен для P и t.

1. если $t \in NF$ и $\mathrm{FV}(t) \cap \{F_1,\ldots,F_n\} = \emptyset$, то t, иначе, если $t \notin NF$, перейти к шагу 2, иначе перейти к шагу 3;

2. пусть $t \equiv t_\tau$ и τ – самый левый $\beta\delta$-редекс, тогда $\mathrm{LIS}(P, t_{\tau'})$, где τ' – свертка редекса τ;

3. пусть $t \equiv t < F_i >$, где F_i – самое левое внутреннее вхождение переменой из множества $\{F_1,\ldots,F_n\}$ в терм t, тогда $\mathrm{LIS}(P, t < \tau_i >)$.

Алгоритм PIS (основанный на параллельной внутренней замене).

Вход: программа P, терм t.

Выход: терм $\mathrm{PIS}(P,t)$, если PIS определен для P и t.

1. если $t \in NF$ и $\mathrm{FV}(t) \cap \{F_1,\ldots,F_n\} = \emptyset$, то t, иначе,
   если $t \notin NF$, перейти к шагу 2, иначе перейти к шагу 3;

2. пусть $t \equiv t_\tau$ и τ – самый левый $\beta\delta$-редекс, тогда $\mathrm{PIS}(P, t_{\tau'})$, где τ' – свертка редекса τ;

3. пусть $t \equiv t < F_{i_1},\ldots,F_{i_s} >$, где $F_{i_1},\ldots,F_{i_s}$ – все внутренние вхождения переменных из множества $\{F_1,\ldots,F_n\}, j = 1,\ldots,s$, в терм t, тогда $\mathrm{PIS}(P, t < \tau_{i_1},\ldots,\tau_{i_s} >)$.

Алгоритм LES (основанный на самой левой внешней замене).

Вход: программа P, терм t.

Выход: терм $\mathrm{LES}(P,t)$, если LES определен для P и t.

1. если $t \in NF$ и $\mathrm{FV}(t) \cap \{F_1,\ldots,F_n\} = \emptyset$, то t, иначе,
   если $t \notin NF$, перейти к шагу 2, иначе перейти к шагу 3;

2. пусть $t \equiv t_\tau$ и τ – самый левый $\beta\delta$-редекс, тогда $\mathrm{LES}(P, t_{\tau'})$, где τ' – свертка редекса τ;

3. пусть $t \equiv t < F_i >$, где F_i – самое левое внешнее вхождение переменной из множества $\{F_1,\ldots,F_n\}$ в терм t, тогда $\mathrm{LES}(P, t < \tau_i >)$.

Скажем, что процедурные семантики, использующие алгоритмы интерпретации A и B, несравнимы, если существует такая программа P, что

а) $\mathrm{Proc}_A(P) \not\subset \mathrm{Proc}_B(P)$,

б) $\mathrm{Proc}_B(P) \not\subset \mathrm{Proc}_A(P)$.

Теорема 2 (о несравнимости процедурных семантик). Существуют такие множество M и понятие δ-редукции, что процедурные семантики,

использующие алгоритмы интерпретации ACT и PAS, ACT и LES, PAS и LIS, PAS и PIS, LIS и LES, PIS и LES, несравнимы.

Доказательство. Пусть Z – множество целых чисел, $M = Z \cup \{\perp\}$. $* \in [M^2 \to M]$, где

$$*(m_1, m_2) = \begin{cases} 0, & \text{åñëè } m_1 = 0 \text{ èëè } m_2 = 0, \\ n, & \text{åñëè } m_1, m_2 \in Z, \text{ è } n \text{ ÿâëÿåòñÿ ðåçóëüòàòîì óì í î æåí èÿ } m_1 \text{ í à } m_2, \\ \perp & \text{â îñòàëüíûõ ñëó÷àÿõ.} \end{cases}$$

Рассмотрим следующее понятие δ-редукции: для любых $t \in \Lambda_M$, $n, n_1, n_2 \in Z$ и $m \in M$ имеем:

$<*(n_1, n_2), n> \in \delta$, где n является результатом умножения n_1 на n_2,

$<*(t, 0), 0> \in \delta$,

$<*(0, t), 0> \in \delta$,

$<*(m, \perp), \perp> \in \delta$, где $m \neq 0$,

$<*(\perp, m), \perp> \in \delta$, где $m \neq 0$.

В работе [3] доказывается, что данное понятие δ-редукции является естественным и обладает ПН-свойством. Рассмотрим следующие программы P1 и P2:

$$\text{P1} \quad \begin{cases} F_1 = \lambda x[F_2(F_3(x))], \\ F_2 = \lambda x[*F_4(x), x)], \\ F_3 = \lambda x[0], \\ F_4 = \lambda x[F_4(x)], \end{cases} \qquad \text{P2} \quad \begin{cases} F_1 = \lambda x[F_2(F_3(x))], \\ F_2 = \lambda x[1], \\ F_3 = \lambda x[F_3(x)], \end{cases}$$

где $x \in V_M$, $F_1, F_2, F_3, F_4 \in V_{[M \to M]}$.

Рассмотрим некоторые применения алгоритмов интерпретации ACT, PAS, LES, LIS и PIS.

$$\text{ACT}(\text{P1}, \text{F}_1(1)) = \text{ACT}(\text{P1}, \lambda\text{x}[\text{F}_2(\text{F}_3(\text{x}))](1)) = \text{ACT}(\text{P1}, \text{F}_2(\text{F}_3(1))) = \text{ACT}(\text{P1}, \lambda\text{x}[*(\text{F}_4(\text{x}), \text{x})](\text{F}_3(1))).$$

Далее, согласно алгоритму интерпретации ACT, имеем

$$\text{ACT}(\text{P1}, \text{F}_3(1)) = \text{ACT}(\text{P1}, \lambda\text{x}[0](1)) = \text{ACT}(\text{P1}, 0) = 0.$$

Таким образом,

$$\text{ACT}(\text{P1}, \lambda\text{x}[*(\text{F}_4(\text{x}), \text{x})](\text{F}_3(1))) = \text{ACT}(\text{P1}, \lambda\text{x}[*(\text{F}_4(\text{x}), \text{x})](0)) = \text{ACT}(\text{P1}, *(\text{F}_4(0), 0)) = \text{ACT}(\text{P1}, 0) = 0.$$

Следовательно, $<1,0> \in \text{Proc}_{ACT}(\text{P1})$.

$$\text{ACT}(\text{P2}, \text{F}_1(1)) = \text{ACT}(\text{P2}, \lambda\text{x}[\text{F}_2(\text{F}_3(\text{x}))](1)) = \text{ACT}(\text{P2}, \text{F}_2(\text{F}_3(1))) = \text{ACT}(\text{P2}, \lambda\text{x}[1](\text{F}_3(1))).$$

Далее, согласно алгоритму интерпретации ACT,

$$\text{ACT}(\text{P2}, \text{F}_3(1)) = \text{ACT}(\text{P2}, \lambda\text{x}[\text{F3(x)}](1)) = \text{ACT}(\text{P2}, \text{F3}(1)).$$

То есть алгоритм ACT будет функционировать бесконечно. Следовательно, $<1,1> \notin \text{Proc}_{ACT}(\text{P2})$.

$PAS(P1, F_1(1)) = PAS(P1, \lambda x[F_2(F_3(x))](1)) = PAS(P1, \lambda x[*(F_4(x), x)](F_3(1))) =$
$PAS(P1,* (F_4(F_3(1)), F_3(1))) = PAS(P1,* (\lambda x[F_4(x)](F_3(1)), F_3(1))) =$
$PAS(P1,* (F_4(F_3(1)), F_3(1)))).$

То есть алгоритм PAS будет функционировать бесконечно. Следовательно, $<1,0> \notin Proc_{PAS}(P1)$.

$PAS(P2, F_1(1)) = PAS(P2, \lambda x[F_2(F_3(x))](1)) = PAS(P2, F_2(F_3(1))) =$
$PAS(P2, \lambda x[1](F_3(1))) = PAS(P2,1) = 1.$

Следовательно, $<1,1> \in Proc_{PAS}(P2)$.

$LES(P1, F_1(1)) = LES(P1, \lambda x[F_2(F_3(x))](1)) = LES(P1, F_2(F_3(1))) =$
$LES(P1, \lambda x[*(F_4(x), x)](F_3(1))) = LES(P1,* (F_4(F_3(1)), F_3(1))) =$
$LES(P1,* (\lambda x[F_4(x)](F_3(1)), F_3(1))) = LES(P1,*(F_4(F_3(1)), F_3(1)))).$

То есть алгоритм LES будет функционировать бесконечно. Следовательно, $<1,0> \notin Proc_{LES}(P1)$.

$LES(P2, F_1(1)) = LES(P2, \lambda x[F_2(F_3(x))](1)) = LES(P2, F_2(F_3(1))) =$
$LES(P2, \lambda x[1](F_3(1))) = LES(P2,1) = 1.$

Следовательно, $<1,1> \in Proc_{LES}(P2)$.

$LIS(P1, F_1(1)) = LIS(P1, \lambda x[F_2(F_3(x))](1)) = LIS(P1, F_2(F_3(1))) =$
$LIS(P1, F2(\lambda x[0](1))) = LIS(P1, F_2(0)) = LIS(P1, \lambda x[*(F_4(x), x)](0)) =$
$LIS(P1,* (F_4(0), 0)) = LIS(P1, 0) = 0.$

Следовательно, $<1,0> \in Proc_{LIS}(P1)$.

$LIS(P2, F_1(1)) = LIS(P2, \lambda x[F_2(F_3(x))](1)) = LIS(P2, F_2(F_3(1))) =$
$LIS(P2, F2(\lambda x[F_3(x)](1))) = LIS(P2, F_2(F_3(1)))).$

То есть алгоритм LIS будет функционировать бесконечно. Следовательно, $<1,1> \notin Proc_{LIS}(P2)$.

$PIS(P1, F_1(1)) = PIS(P1, \lambda x[F_2(F_3(x))](1)) = PIS(P1, F_2(F_3(1))) =$
$PIS(P1, F_2(\lambda x[0](1))) = PIS(P1, F_2(0)) = PIS(P1, \lambda x[*(F_4(x), x)](0)) =$
$PIS(P1,* (F_4(0), 0)) = PIS(P1, 0) = 0.$

Следовательно, $<1,0> \in Proc_{PIS}(P1)$.

$PIS(P2, F_1(1)) = PIS(P2, \lambda x[F_2(F_3(x))](1)) = PIS(P2, F_2(F_3(1))) =$
$PIS(P2, F2(\lambda x[F_3(x)](1))) = PIS(P2, F_2(F_3(1)))).$

То есть алгоритм PIS будет функционировать бесконечно. Следовательно, $<1,1> \notin Proc_{PIS}(P2)$.

Покажем, что процедурные семантики, использующие алгоритмы интерпретации ACT и PAS, несравнимы. Из $<1,0> \in Proc_{ACT}(P1)$ и $<1,0> \notin Proc_{PAS}(P1)$ следует, что $Proc_{ACT}(P1) \not\subset Proc_{PAS}(P1)$, из $<1,1> \notin Proc_{ACT}(P2)$ и $<1,1> \in Proc_{PAS}(P2)$ следует, что $Proc_{PAS}(P2) \not\subset Proc_{ACT}(P2)$.

Покажем, что процедурные семантики, использующие алгоритмы интерпретации ACT и LES, несравнимы. Из $<1,0>\in \mathrm{Proc}_{ACT}(\mathrm{P1})$ и $<1,0>\notin \mathrm{Proc}_{LES}(\mathrm{P1})$ следует, что $\mathrm{Proc}_{ACT}(\mathrm{P1}) \not\subset \mathrm{Proc}_{LES}(\mathrm{P1})$, из $<1,1>\notin \mathrm{Proc}_{ACT}(\mathrm{P2})$ и $<1,1>\in \mathrm{Proc}_{LES}(\mathrm{P2})$ следует, что $\mathrm{Proc}_{LES}(\mathrm{P2}) \not\subset \mathrm{Proc}_{ACT}(\mathrm{P2})$.

Покажем, что процедурные семантики, использующие алгоритмы интерпретации PAS и LIS, несравнимы. Из $<1,0>\notin \mathrm{Proc}_{PAS}(\mathrm{P1})$ и $<1,0>\in \mathrm{Proc}_{LIS}(\mathrm{P1})$ следует, что $\mathrm{Proc}_{LIS}(\mathrm{P1}) \not\subset \mathrm{Proc}_{PAS}(\mathrm{P1})$, из $<1,1>\in \mathrm{Proc}_{PAS}(\mathrm{P2})$ и $<1,1>\notin \mathrm{Proc}_{LIS}(\mathrm{P2})$ следует, что $\mathrm{Proc}_{PAS}(\mathrm{P2}) \not\subset \mathrm{Proc}_{LIS}(\mathrm{P2})$.

Покажем, что процедурные семантики, использующие алгоритмы интерпретации PAS и PIS, несравнимы. Из $<1,0>\notin \mathrm{Proc}_{PAS}(\mathrm{P1})$ и $<1,0>\in \mathrm{Proc}_{PIS}(\mathrm{P1})$ следует, что $\mathrm{Proc}_{PIS}(\mathrm{P1}) \not\subset \mathrm{Proc}_{PAS}(\mathrm{P1})$, из $<1,1>\in \mathrm{Proc}_{PAS}(\mathrm{P2})$ и $<1,1>\notin \mathrm{Proc}_{PIS}(\mathrm{P2})$ следует, что $\mathrm{Proc}_{PAS}(\mathrm{P2}) \not\subset \mathrm{Proc}_{PIS}(\mathrm{P2})$.

Покажем, что процедурные семантики, использующие алгоритмы интерпретации LIS и LES, несравнимы. Из $<1,0>\in \mathrm{Proc}_{LIS}(\mathrm{P1})$ и $<1,0>\notin \mathrm{Proc}_{LES}(\mathrm{P1})$ следует, что $\mathrm{Proc}_{LIS}(\mathrm{P1}) \not\subset \mathrm{Proc}_{LES}(\mathrm{P1})$, из $<1,1>\notin \mathrm{Proc}_{LIS}(\mathrm{P2})$ и $<1,1>\in \mathrm{Proc}_{LES}(\mathrm{P2})$ следует, что $\mathrm{Proc}_{LES}(\mathrm{P2}) \not\subset \mathrm{Proc}_{LIS}(\mathrm{P2})$.

Покажем, что процедурные семантики, использующие алгоритмы интерпретации PIS и LES, несравнимы. Из $<1,0>\in \mathrm{Proc}_{PIS}(\mathrm{P1})$ и $<1,0>\notin \mathrm{Proc}_{LES}(\mathrm{P1})$ следует, что $\mathrm{Proc}_{PIS}(\mathrm{P1}) \not\subset \mathrm{Proc}_{LES}(\mathrm{P1})$, из $<1,1>\notin \mathrm{Proc}_{PIS}(\mathrm{P2})$ и $<1,1>\in \mathrm{Proc}_{LES}(\mathrm{P2})$ следует, что $\mathrm{Proc}_{LES}(\mathrm{P2}) \not\subset \mathrm{Proc}_{PIS}(\mathrm{P2})$.

Теорема 2 доказана.

Кафедра программирования и информационных технологий

Поступила 02.04.2008

ЛИТЕРАТУРА

1. **Нигиян С.А.** – Программирование, 1991, №5, с. 77–86.
2. **Нигиян С.А.** – Программирование, 1993, №2, с. 58 – 68.
3. **Будагян Л.Э**. – Ученые записки ЕГУ, 2003, №1, с. 27–36.

Ռ. Յու. ՀԱԿՈԲՅԱՆ

ԽԻՍՏ ՏԻՊԱԿԱՆԱՑՎԱԾ ՖՈՒՆԿՑԻՈՆԱԼ ԾՐԱԳՐԵՐԻ ՊՐՈՑԵԴՈՒՐԱՅԻՆ ՍԵՄԱՆՏԻԿԱՆԵՐԻ ՄԱՍԻՆ

Ամփոփում

Հոդվածը նվիրված է խիստ տիպականացված ֆունկցիոնալ ծրագրերի պրոցեդուրային սեմանտիկաներին, որոնք հիմնված են ինտերպրետացիայի

ալգորիթմների վրա: Այդ ալգորիթմները օգտագործում են երեք գործողություն՝ տեղադրում, միաքայլ β-ռեդուկցիա և միաքայլ δ-ռեդուկցիա: Ապացուցվում է, որ պրոցեդուրային սեմանտիկաները, որոնք հիմնված են այդպիսի ալգորիթմներից ցանկացածի վրա, անհակասելի են: Ապացուցվում է նաև պրոցեդուրային սեմանտիկաների անհամեմատելիությունը՝ հիմնված ինտերպրետացիայի որոշ ալգորիթմների վրա:

R. Yu. HAKOPIAN

ON PROCEDURAL SEMANTICS OF STRONG TYPED FUNCTIONAL PROGRAMS

Summary

The paper is devoted to procedural semantics of strong typed functional programs that are based on interpretation algorithms. These algorithms use three operations: substitution, one-step β-reduction and one-step δ-reduction. It is proved that the procedural semantics, based on any of such algorithms are consistent. It is also proved that procedural semantics, based on several interpretation algorithms are incomparable.