$\Sigma-$спецификация языков программирования

В. Н. Глушкова
Донской государственный технический университет, Ростов-на-Дону

АННОТАЦИЯ. Выделен новый класс $\Delta_0 T-$формул в рамках концепции $\Sigma-$программирования [1], составляющий основу логической спецификации семантики языков программирования. Специфика этих формул состоит в том, что их префикс с ограниченными кванторами иерархизирован в соответствии с правилами КС-грамматики языка. Использование $\Delta_0 T-$формул упрощает спецификацию статической семантики языка по сравнению с квазитождествами с отрицанием, применяемыми в [2], для которых требуется наличие дополнительных позитивных предикатов перед их негативными вхождениями. Наличие иерархизированного префикса позволяет более эффективно организовать интерпретацию логической спецификации и получить оценку сложности ее реализации с учетом синтаксиса языка.

КЛЮЧЕВЫЕ СЛОВА: логическая спецификация семантики языков программирования, формулы многосортного языка ИП 1-го порядка с ограниченными кванторами, КС-грамматика

Концепция $\Sigma-$программирования использует аппарат теории моделей в качестве семантической основы логического программирования. В её рамках выделен вычислимый класс формул многосортного ИП 1-го порядка, позволяющий специфицировать свойства объектов сложной списочной структуры с явным использованием переменных сорта "список". В работе выделен практически значимый класс $\Delta_0 T-$формул, интепретируемый на моделях с иерархической надстройкой, описываемой КС-грамматикой. Показано, что степень полинома зависит от вида символов грамматики.

1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Используем обозначения и терминологию работ [3] - [4]. Пусть M – многосортная модель сигнатуры $\sigma_0 = \langle I, F_0, R_0 \rangle$, где $I-$ множество сортов; F_0,R_0 – множества символов операций и отношений соответственно. Каждый сигнатурный символ имеет специфичный тип: функция $f \in F_0$ имеет тип $\langle i_1, \ldots, i_n, i \rangle$, $n \geq 0$; предикат $p \in R_0-$ $\langle i_1, \ldots, i_n \rangle$. Модель M состоит из индексированного семейства множеств $\{M_i\}_{i \in I}$ (носителя) и индексированного семейства функций и предикатов, определенных на декартовом произведении соответствующих множеств. Каждый символ $f \in F$ типа $\langle i_1, \ldots, i_n, i \rangle$ интерпретируется функцией $f : M_{i_1} \times \cdots \times M_{i_n} \to M_i$; $p \in P$ типа $\langle i_1, \ldots, i_n \rangle$ — предикатом $p \subseteq M_{i_1} \times \cdots \times M_{i_n} \to M_i$. Функция f типа $\langle \varepsilon, i \rangle$ (ε —пустая строка) – константа сорта i, т.е.$f \in M_i$.

В $\Sigma-$ программировании вместе с моделью M рассматривается наследственно-конечная списочная надстройка над нею: модель $HFS(M)$ с множеством сортов $I \cup \{list\}$. Носитель $S(M)$ сорта $list$ формируется следующим образом. Множество $S^0(M)$ (линейные списки) конструируется из элементов множества $\hat{M} = \bigcup\limits_{i \in I} M_i$ как атомов; $S^{i+1}(M)$ (линейные списки) формируется из объектов множества $S^i(M) \bigcup \hat{M}$. Наконец, $S(M) = \bigcup\limits_{i \geq 0} S^i(M)$, т.е. списки из $S(M)$ имеют произвольную глубину вложенности. Для списков в модели $HFS(M)$ вводятся различные операции и отношения: $nil-$ пустой список; $conc$ — конкатенация двух списков; $cons$ — присоединение к списку нового элемента; $\in$ — отношение принадлежности элемента списку и др. Для списков $u = \langle u_1 \ldots, u_l \rangle$, $v = \langle u_1, \ldots, u_n \rangle$ выполняется отношение $u \sqsubseteq v$, если $l \leq n$.

Обозначим сигнатуру модели $HFS(M)$ через $\sigma = \langle I \cup \{list\}, F, R\rangle$, где F, R содержат F_0, R_0 с дополнительными операциями и предикатами для списков. Термы и формулы сигнатур σ_0, σ определяются обычным образом с применением логических связок $\neg$, $\wedge$, $\vee$,$\rightarrow$, кроме того для формул разрешается использовать ограниченные кванторы: $\forall x \in t$, $\exists x \in t$, $\forall x \sqsubseteq t$, $\exists x \sqsubseteq t$, где $x-$ переменная произвольного сорта; t — терм $list$-сорта (t не содержит x).

Определение 1. Формула, в записи которой встречаются только ограниченные кванторы, называется ф о р м у л о й с о г р а н и ч е н н ы м и к в а н т о р а м и, или Δ_0-ф о р м у л о й.

2. Δ_0T- ФОРМУЛЫ

Рассмотрим иерархическую надстройку модели, формируемую в соответствии с правилами из P некоторой $KC-$грамматики $G = (T, N, P)$, где T, $N-$ множества терминальных и нетерминальных символов соответственно. Алфавит $I = T \cup N-$ задает сорта модели M, а семейство C_X, $X \in I-$ ее носитель, где C_X- не более чем счетное множество констант сорта X. Элементы сорта $X \in I$ рассматриваются как узлы c_X дерева вывода G, помеченные символом X. Введем контекстно-свободное множество списков, используемое в качестве носителя сорта $list$ модели $HFS(M)$.

Определение 2. Пусть $G-$ KC- грамматика и $C = \{C_X\}_{X\in I}-$ семейство констант. КС-множество списков над C – это наименьшее множество $D_G(C)$ всех списков $\langle t_1, \ldots, t_n\rangle$, сопоставляемых каждому правилу $A \rightarrow X_1 \ldots X_n$ из P, $n \geq 1$, $X_i \in I$, следующим образом : если $X_i \in T$, то t_i- произвольная константа из C_{X_i}, в противном случае ($X_i \in N$) t_i- произвольный список сорта X_i. Список $\langle t_1, \ldots, t_n\rangle$ имеет сорт A.

Рассмотрим арифметические выражения, описываемые правилами:

$$E \rightarrow E + T \mid T * F \mid (E) \mid id$$

$$T \rightarrow T * F \mid (E) \mid id$$

$$F \rightarrow (E) \mid id,$$

где терминальный символ id разпознается на лексическом уровне. Дереву вывода:

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow T * F + T \Rightarrow id_1 * F + T \Rightarrow id_1 * id_2 + T \\ &\Rightarrow id_1 * id_2 + id_3 \end{aligned}$$

соответствует индексированный сортами список $\langle\langle Id_1\rangle_T * \langle Id_2\rangle_F\rangle_E + \langle Id_3\rangle_T\rangle_E$; Id_1, Id_2, Id_3— константы $id-$сорта.

При таком представлении деревьев списками отношению $\in$ для списков соответствует отношение непосредственного подчинения соответствующих узлов дерева. Введем для списков отношение $\overset{d}{\prec}$ "точно левее". Для списка $v = \langle u_1, \ldots, u_n\rangle$ выполняется $u_i \overset{d}{\prec} u_{i+1}$, $1 \leq i \leq n-1$. Пусть $\overset{*}{\in}$, $\overset{l}{\prec}$ транзитивное замыкание отношения $\in$ and $\overset{d}{\prec}$ соответственно. Отношение "левее"$\prec$ задается формулой:

$$(\forall y, z \overset{*}{\in} t)(y \prec z) \Longleftrightarrow (y \overset{l}{\prec} z) \vee (\exists y_1, z_1 \overset{*}{\in} t)(y_1 \overset{l}{\prec} z_1 \wedge y \overset{*}{\in} y_1 \wedge z \overset{*}{\in} z_1).$$

Введем класс $\Delta_0 T$-формул с иерархическим префиксом. При этом будем использовать универсальные ограниченные кванторы $\forall x \in t$ и $\forall x \overset{*}{\in} t$; через $\bar{x}$ будем обозначать последовательность переменных x_i, $1 \le i \le m$; $\dot{\in}$ – отношение $\in$ или $\overset{*}{\in}$; $\prec$ – отношение $\overset{d}{\prec}$ или $\overset{l}{\prec}$.

Определение 3.

Δ_0- формула называется $\Delta_0 T-$ формулой, если ее префикс имеет "древесную"структуру:

$$(1) \quad (\forall\, x_1 \dot{\in} t_1) \dots (\forall\, x_m \dot{\in} t_m)\ (y_1 \prec z_1) \dots (y_p \prec z_p) \Phi(\bar{x}, \bar{t}), \quad m \ge 1, \quad p \ge 0$$

Здесь переменные удовлетворяют условию: $t_{i+1} = t_i$ или $t_{i+1} = x_k$ для всех $1 \le i \le m$, $k \le i$; если $t_{i+1} = x_k$, то $x_{i+1} \ne x_l$ и $x_{i+1} \ne t_l$, для всех $l \le i$; $y_j, z_j \in (\bar{x}, \bar{t})$, $1 \le j \le p$.

Представляя все префиксные переменные узлами с дугами, идущими от узла t_i к узлу x_i, $1 \le i \le m$ получим дерево с корнем t_1.

Множество $D_G(C)$ можно представить как объединение $\bigcup\limits_{A \in N} D_G^A(C)$ и рассмотреть две модели M и $KC(M)$ сигнатуры $\sigma = \langle I \cup \{list\}, F, R \rangle$, для которой сорта и списочная надстройка определяются некоторой $KC-$ грамматикой G; F, R — множества функциональных и предикатных символов. Модель M имеет многосортный носитель $\{C_X\}_{X \in I}$ и списочную надстройку $D_G(C)$.

Модель $KC(M)$ будем называть $N-$ структурированным представлением модели M, если они имеют одни и те же носители, исключая носители нетерминальных сортов. Множество $D_G^A(C)$ является носителем сорта A, $A \in N$ в модели $KC(M)$.

Рассмотрим теорию из $\Delta_0 T$-квазитождеств, для которых формула $\Phi(\bar{\alpha})$ является импликацией вида: $\varphi(\bar{x}, \bar{t}) \to \psi(\bar{x}, \bar{t})$, где φ (ψ) конъюнкция атомных формул или их отрицаний вида p, $\tau_1 = \tau_2$ $(r,\ f = \tau)$; $p, r, f-$ предикатные и функциональный символы τ, τ_1, τ_2- термы. Пусть теория $Th = T_0 \cup Tr \cup H$ включает конечные множества T_0 замкнутых атомарных формул вида $f(\bar{c}) = c_i, p(\bar{c})$ и $Tr : \{c_i \in c_j \mid c_i, c_j \in C_t, C_t \subseteq C\}$. Отношения из Tr определяют дерево tr, в котором константы c_i являются узлами, такими что $sort(c_i) \in I$, $sort(c_j) \in N$; дереву tr соответствует некоторая последовательность правил грамматики G. Теория $H-$ это конечное множество универсальных $\Delta_0 T$-квазитождеств.

Теория Th трактуется как система определений функций и отношений, заданных на дереве tr, узлы которого представлены константами соответствующего сорта из C. Теория из формул рассматриваемого вида интерпретируется на исходном КС-списке по правилу вывода modus ponens, исходя из фактов вида $r(\bar{c})$, $f(\bar{c}) = d$ для констант $\bar{c}$, d. При интерпретации арифметические операции считаются встроенными, отрицание интерпретируется по принципу "замкнутого мира". Для существования модели на теорию накладываются такие же ограничения как в [4]. Если теория из квазитождеств обладает свойством нётеровости и конфлюентности, то для неё можно построить индуктивно вычислимую модель с иерархической надстройкой из констант.

3. СПЕЦИФИКАЦИЯ ПОСЛЕДОВАТЕЛЬНЫХ ПРОГРАММ

Для построения формальной модели языка программирования будем использовать теорию из $\Delta_0 T$- формул вида:

$$(2) \quad (\forall x_1 \dot{\in} t_1) \dots (\forall x_m \dot{\in} t_m)(y_1 \prec z_1) \dots (y_p \prec z_p)\varphi(\bar{x}, \bar{t}) \to \psi(\bar{x}, \bar{t}),$$

где $m \geq 1$, $p \geq 0$, $y_j, z_j \in < \bar{x}, \bar{t} >$, $1 \leq j \leq p$; $\dot{\in}$ обозначает отношение принадлежности элемента списку или его транзитивное замыкание, $\prec$ – отношение "левее". Формулы φ (ψ) конъюнкция атомных формул вида r, $\tau_1 = \tau_2$ (r, $f = \tau$) или их отрицаний; $r, f-$ предикатный и функциональный символы, τ, τ_1, τ_2- термы. Формулами этого вида можно описывать предикаты и функции, определенные на поддеревьях дерева вывода.

Рассмотрим некоторые типовые операторы языка программирования.

Будем считать, что программа ($prog$) составлены из операторов (op) присваивания (as), условных операторов (con) и циклов $while(cyc)$. Синтаксис программ задается грамматикой:

$prog \to \{op\}^+$;
$op \to as \mid con \mid cyc$;
$as \to var := exp$;
$con \to \ if \ \ bexp \ \ then \, \{op\}^+ \ else \, \{op\}^+$;
$cyc \to \ bexp \, \{op\}^+$

Пусть $\widetilde{pr} = \langle \widetilde{op_1}, ..., \widetilde{op}_i, ..., \widetilde{op}_m \rangle-$ списочное представление конкретной (текстовой) программы, соответствующей $\widetilde{pr}$. $\Sigma-$спецификация описывает процесс выполнения операторов $\widetilde{op_i}$ программы шаг за шагом посредством квазитождеств вида (2), левая часть которых содержит исполняемый оператор, а правая часть – результат его исполнения.

Пусть $Var-$ множество всех переменных программы, $Q-$ множество рациональных чисел, $N-$ множество натуральных чисел, $n \in N-$ номер шага вычисления программы. Функция $Val : Var \times N \to Q \cup \{\perp\}$ задает значения переменных на $n-$ом шаге вычисления, $\perp-$ неопределенное значение; $Vald : Var \times N \to Q \cup \{\perp\}-$ задает продолжение функции Val по n, именно $Vald(var, n) = \max\limits_{m<n} Val(var, m)$. Предикат $Act \subseteq Op \times N$ выделяет оператор $op \in Op$, исполняемый на $n-$ом шаге работы программы pr; предикат $Ter \subseteq Op \times N$ выделяет операторы, завершившие свое исполнение на $n-$ом шаге работы программы.

Одним шагом вычисления считается выполнение одного оператора присваивания, т.е. программа делает общее количество шагов равное числу выполняемых операторов присваивания. Если все $\widetilde{op}_i$ программы $\widetilde{pr}$ являются операторами присваивания, то количество выполняемых шагов программы равно m, причем справедливо $Act(\widetilde{op}_i, i)$. Для программы, содержащей операторы цикла или условные операторы, общее число шагов вычисления зависит от входных данных и количества операторов этого же вида, вложенных в них. Для программ, не завершающих своей работы, общее количество шагов неограниченно. Константы, переменные обрабатываются на лексическом уровне. Для всех переменных, входящих в выражения (exp, $bexp$), считаются известными все их атрибуты (типы, значения), позволяющие вычислить значения самих выражений.

В логической спецификации все переменные индексируются сортами. Аксиомы теории Th интерпретируются на дереве разбора одной программы $\widetilde{pr}$, поэтому переменная t_{prog} интерпретируется одним списком, представляющим анализируемую программу. Переменная n (шаг вычисления) связана неограниченным квантором $\forall$. Для всех вводимых переменных x известны их значения на 0−ом шаге, заданные функцией $Val(x, 0)$.

1. Спецификация активности операторов программы.

1.1 $(\forall\, x_{op} \in t_{prog})(nil \overset{d}{\prec} x_{op})Act(x_{op}, 1)$

Первым исполняется самый левый оператор программы, т.к. он удовлетворяет префиксу формулы $nil \prec x_{op}$. Если $t_{prog} = \widetilde{pr}$, то этим оператором является $\widetilde{op_1}$. Далее могут применяться аксиомы разделов 2, 3, 4 в зависимости от вида оператора $\widetilde{op_2}$ программы $\widetilde{pr}$.

1.2. $(\forall\, x_{op}, y_{op} \in t_{prog})(x_{op} \overset{d}{\prec} y_{op})(Ter(x_{op}, n) \to Act(y_{op}, n))$

Операторы программы активизируются последовательно. После завершения работы оператора $\widetilde{op_i}$ на шаге n (выполняется $Ter(\widetilde{op_i}, n)$) активизируется следующий за ним оператор $\widetilde{op}_{i+1}$.

1.3. $(\forall\, x_{op} \in t_{prog})(x_{op} \overset{d}{\prec} nil)(Ter(x_{op}, n) \to Ter(t_{prog}, n))$

После исполнения последнего оператора $\widetilde{op}_m$ программы на этом же шаге завершается исполнение программы.

2. Спецификация оператора присваивания.

2.1. $(\forall x_{op} \dot{\in} t_{prog})(\forall\, y_{as} \in x_{op})(Act(x_{op}, n) \to Act(y_{as}, n))$

Если $x_{op} = \widetilde{op}_i$ является оператором присваивания, то активизируется соответствующий ему оператор y_{as}.

2.2. $(\forall\, x_{op} \dot{\in} t_{prog})(\forall\, y_{as} \in x_{op})(\forall\, z_{var}, w_{exp} \in y_{as}), (nil \overset{d}{\prec} z_{var})$
$(Act(y_{as}, n) \to Val(z_{var}, n) = Vald(w_{exp}, n-1), Ter(y_{as}, n), Ter(x_{op}, n))$

Результатом работы оператора присваивания на $n-$ ом шаге вычисления является изменение значения переменной z_{var} (из левой части оператора). Функция $Val(z_{var}, n)$ равна значению выражения w_{exp} из правой части оператора, вычисленного на предыдущем $(n-1)-$шаге для избежания коллизии при вхождении той же самой переменной z_{var} в w_{exp}. Вычисление значения выражения можно определить формулами этого же вида с учетом его синтаксиса. Далее операторы y_{as}, x_{op} завершают свою работу.

3. Спецификация условного оператора.

3.1. $(\forall\, x_{op} \in t_{prog})(\forall\, y_{con} \in x_{op})(Act(x_{op}, n) \to Act(y_{con}, n))$

Для активного условного оператора $x_{op}{=}\widetilde{op}_i$ активизируется соответствующий оператор y_{con}.

3.2.$(\forall\, y_{con} \dot{\in} t_{prog})(\forall\, w_{bexp}, x_{op} \in y_{con})(u_{then} \overset{d}{\prec} op_1)$
$(Act(con, n), \neg Act(con, n-1), Vald(bexp, n-1) = 1 \to Act(op_1, n))$

При первой активации условного оператора con на шаге n и истинности подчиненного ему булевского выражения $bexp$, вычисленного на предыдущем $(n-1)-$ом шаге, активизируется первый оператор, стоящий после лексемы $then$.

3.3.$(\forall\, y_{con} \dot{\in} t_{prog})(\forall\, w_{bexp}, x_{op} \in y_{con})(u_{else} \overset{d}{\prec} op_1)$
$(Act(y_{con}, n), \neg Act(y_{con}, n-1), Vald(w_{bexp}, n-1) = 0 \to Act(x_{op}, n))$

В активном условном операторе y_{con} при ложности подчиненного ему булевского выражения w_{bexp} первоначально активизируется первый оператор после лексемы $else$.

3.4.$(\forall\, y_{con} \dot{\in} t_{prog})(\forall\, x_{op}, z_{op} \in y_{con})(x_{op} \overset{d}{\prec} z_{op})$
$(Act(y_{con}, n), Ter(x_{op}, n) \to Act(z_{op}, n+1), Act(y_{con}, n+1))$

После завершения работы оператора x_{op} на шаге n активизируется следующий за ним z_{op} на шаге $n+1$.

3.5.$(\forall\ x_{op}\dot{\in} t_{pr})(\forall\ y_{con} \in x_{op})(\forall\ z_{op} \in y_{con})(\forall\ z_{op} \overset{d}{\prec} u_{else})$
$(Ter(z_{op},n) \to Ter(y_{con},n), Ter(x_{op},n))$

Истинность предиката $Ter(z_{op},n)$ констатирует завершение работы последнего оператора z_{op}, активируемого при истинности булевского выражения, входящего в условный оператор. На этом же шаге завершается работа оператора x_{op} и подчиненного ему условного оператора y_{con}.

3.6.$(\forall\ x_{op}\dot{\in} t_{pr})(\forall\ y_{con} \in x_{op})(\forall\ z_{op} \in y_{con})(z_{op} \overset{d}{\prec} nil)$
$(Ter(z_{op},n) \to Ter(y_{con},n), Ter(x_{op},n))$

В правой части формулы констатируется завершение работы последнего оператора z_{op} после лексемы $else$, активируемого при ложности булевского выражения, входящего в условный оператор. На этом же шаге завершаются операторы y_{con} и x_{op}.

4. Спецификация оператора цикла.

4.1.$(\forall\ x_{op}\ \dot{\in} t_{prog})(\forall\ y_{cyc} \in x_{op})(\forall\ u_{bexp} \in y_{cyc})$
$(Act(x_{op},n), \neg Act(x_{op},n-1), Vald(u_{bexp},n-1) = 0 \to Ter(x_{op},n))$

Если в первоначально активированном операторе цикла x_{op} ложно булевское выражение из заголовка цикла, оператор завершает свою работу на этом же шаге вычисления.

4.2.$(\forall\ x_{op}\ \dot{\in} t_{prog})(\forall\ y_{cyc} \in x_{op})(\forall\ u_{bexp}, z_{op} \in y_{cyc})(u_{bexp} \overset{d}{\prec} z_{op})$
$(Act(x_{op},n), \neg Act(x_{op},n-1), Vald(u_{bexp},n-1) = 1 \to Act(y_{cyc},n),$
$Act(z_{op},n))$

При первой итерации оператора цикла x_{op} ($Act(x_{op},n), \neg Act(x_{op},n-1)$) активизируется y_{cyc} и первый оператор тела цикла (левее лексемы $bexp$).

4.3.$(\forall\ x_{op}\ \dot{\in} t_{prog})(\forall\ y_{cyc} \in x_{op})(\forall\ z_{op}, w_{op} \in y_{cyc})(z_{op} \overset{d}{\prec} w_{op})$
$(Ter(z_{op},n) \to Act(w_{op},n+1), Act(y_{cyc},n+1), Act(x_{op},n+1))$

Операторы, входящие в тело цикла активизируются последовательно. При завершении z_{op} на шаге n активизируется непосредственно следующий за ним w_{op} на шаге $n+1$.

4.4.$(\forall\ x_{op}\ \dot{\in} t_{prog})(\forall\ y_{cyc} \in x_{op})(\forall\ z_{op}, w_{op} \in y_{cyc})(\forall\ u_{bexp} \overset{d}{\prec} z_{op})(w_{op} \overset{d}{\prec} nil)$
$(Ter(w_{op},n), Vald(u_{bexp},n) = 1 \to Act(z_{op},n+1), Act(x_{op},n+1),$
$Act(y_{cyc},n+1))$

В правой части формулы определяется, что следующая итерация тела цикла начинается с активизации первого оператора цикла u_{op} на шаге $n+1$. Левая часть формулы специфицирует, что эта итерация осуществляется после завершения на шаге n последнего оператора цикла ($w_{op} \prec nil$) при истинности булевского выражения заголовка цикла.

4.5.$(\forall\ x_{op}\ \dot{\in} t_{prog})(\forall\ y_{cyc} \in x_{op})(\forall\ z_{op}, u_{bexp} \in y_{cyc})(z_{op} \overset{d}{\prec} nil)$
$(Ter(z_{op},n), Vald(u_{bexp},n) = 0 \to Ter(y_{cyc},n), Ter(x_{op},n))$

При ложности булевского выражения заголовка цикла после выполнения последнего оператора цикла завершается выполнение операторов y_{cyc} и x_{op} на одном шаге вычисления.

Теория из формул рассматриваемого вида интерпретируется на исходном КС-списке по правилу вывода modus ponens, исходя из фактов вида $r(\overline{c})$, $f(\overline{c}) = d$ для констант $\overline{c}$, d. При интерпретации арифметические операции считаются встроенными, отрицание интерпретируется по принципу "замкнутого мира". Если теория из квазитождеств обладает свойством нётеровости и конфлюентности, то для неё можно построить индуктивно вычислимую модель с иерархической надстройкой из констант.

$\Delta_0 T-$формулами можно специфицировать контекстные свойства и ограничения языка программирования. Пусть сначала определены функции $Type$, $Name$, предикат $\underline{declare - in} \subseteq Id \times Prog$ на определяющих вхождениях идентификаторов и предикат $\underline{use - in}$ на использующих вхождениях идентификаторов. Описание ($\underline{describe - in}$) использующих вхождений специфицируется формулой:

$(\forall\ x_{id}, y_{id}\ \dot{\in} t_{prog})(x_{id}\ \underline{declare - in}\ t_{prog},\ y_{id}\ \underline{use - in}\ t_{prog},$
$Name(x_{id}) = Name(y_{id}) \rightarrow y_{id}\ \underline{describe - in}\ t_{prog}, Type(y_{id}) = Type(x_{id}))$

Контекстное ограничение "каждый используемый в программной единице идентификатор должен быть описан единственным образом" специфицируется двумя формулами:

$(\forall\ x_{id}\ \dot{\in} y_{oper})(\forall\ y_{oper}\ \dot{\in} t_{prog})(x_{id}\ \underline{describe - in}\ t_{prog});$
$(\forall\ x_{id}, y_{id}\ \dot{\in} z_{descr})(\forall\ z_{descr} \in t_{prog})(x_{id} \neq y_{id} \rightarrow Name(x_{id}) \neq Name(y_{id}))$

Сложность проверки $\Delta_0 T-$ формул вида:

$$(\forall x_1 \dot{\in} t_1) \dots (\forall x_m \dot{\in} t_m) \psi(\bar{x}, \bar{t})$$

реализуется со сложностью $O(n^{m+1})$ по времени и $O(n^2)$ по памяти относительно $n-$ количества объектов, составляющих список.

Список литературы

1. Goncharov S.S., Ershov Yu.L., Sviridenko D.I., *Semantic programming* // Information processing. – 1986. – V. 11. – № 10 . – p. 1093–1100.

2. Глушкова В.Н., Ильичева О.А.,*Средства диагностики ошибок и эффективной реализуемости в системах логического моделирования* // Логика и семантическое программирование. Вычислительные системы – 1992. – Вып. 146 . – стр. 3–17.

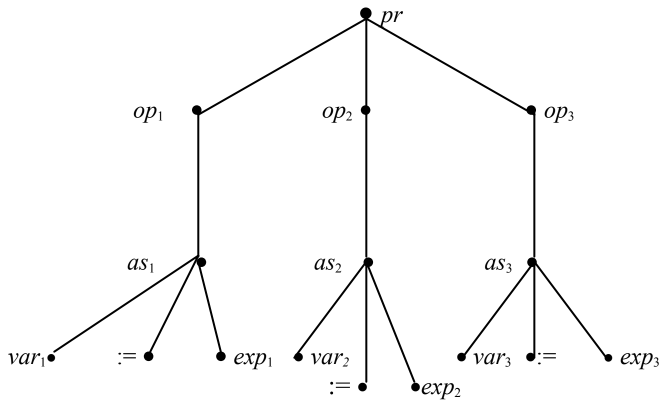
3. Гончаров С.С., $\Sigma-$*программирование*// Вычислительные системы: Логико-математические основы проблемы МОЗ.–1985.– Вып.107.–стр. 3-29.

4. Гончаров С.С.*Модели данных и языки их описаний*//Вычислительные системы: Логико-математические основы проблемы МОЗ.-1985.-Вып.107.-стр. 52-70

Приложение.

Пример. Программа *pr* состоит из трех операторов: $x{:}=3$; $y{:}= x+ 2$; $y{:}=y + x$.

Ее дерево вывода *Tr* имеет вид:



Для программы *pr*: $var_1 = x$, $exp_1 = 3$; $var_2 = y$; $exp_2 = x+ 2$; $var_3 = y$; $exp_3 = y + x$.

Списки, представляющие поддеревья *Tr* обозначим мнемонически меткой их корня. Приведем последовательность шагов вывода интерпретатора, индкксированные номерами шагов:

$$Th \overset{1.1}{\vdash} Act\,(op_1,1) \overset{2.1}{\vdash} Act\,(as_1, 1) \overset{2.2}{\vdash} Val\,(x,1) = Vald\,(3,1) = 3,\ Ter\,(as_1, 2),\ Ter\,(op_1, 2) \overset{1.2}{\vdash}$$

$$Act\,(op_2, 2) \overset{2.1}{\vdash} Act\,(as_2, 2) \overset{2.2}{\vdash} Val\,(y, 2) = Vald\,(x+2, 1) = 5,\ Ter(as_2, 3),\ Ter(op_2, 3) \overset{1.2}{\vdash}$$

$$Act\,(op_3, 3) \overset{2.1}{\vdash} Act\,(as_3, 3) \overset{2.2}{\vdash} Val\,(x, 3) = Vald\,(y + x, 2) = Val\,(y, 2) + Val\,(x, 1) = 5+ 3= 8,$$

$$Ter\,(as_3, 4),\ Ter\,(op_3,4) \overset{1.3}{\vdash} Ter\,(pr, 4)$$

СПАСИБО ЗА ВНИМАНИЕ