ТЕОРИЯ ВЫЧИСЛИМОСТИ И ЯЗЫКИ СПЕЦИФИКАЦИЙ
(Вычислительные системы)
1991 год Выпуск 139

УДК 519.9

О ТРАНСЛЯЦИИ СТАНДАРТНЫХ СХЕМ С МАГАЗИНАМИ

В.А.Гальперин, Л.П.Лисовик

Исследование проблем функциональной эквивалентности, включения, транслируемости и других (см. [1]) в различных классах схем программ сводится к рассмотрению их относительно свобод - ных интерпретаций. Схемам над свободными интерпретациями могут быть сопоставлены соответствующие преобразователи. Например, металинейным унарным рекурсивным схемам с засылками констант соответствуют MLC-схемы [2], представляющие собой специального вида преобразователи над размеченными деревьями. Таким образом, рассмотрение в общем виде преобразователей над размеченными деревьями (см. [3]) может приводить к решению конкретных проблем теории схем программ. В данной статье при исследовании проблем трансляции обогащенным стандартным схемам с одной переменной сопоставлены схемы над свободными интерпретациями и соответствующие им преобразователи над размеченными деревьями. Известно, что в класс стандартных схем транслируемы: класс стандартных схем со счетчиком [4], класс квазирациональных рекурсивных схем и, в частности, класс линейных унарных рекур - сивных схем [5]. В данной работе устанавливаются более общие результаты. Показано, что класс стандартных схем, обогащенных конечным числом конечноповоротных магазинов (или стеков), транслируем в класс стандартных схем. В первой части статьи будут даны определения, сформулированы основные теоремы и следствия

из них, приведено интуитивное (приближенное) описание метода трансляции. Во второй части алгоритм трансляции уточняется и будет описан строго.

Определим вначале класс преобразователей над деревьями, соответствующий классу стандартных схем с одной переменной, обогащенных конечным числом магазинов. Введем некоторые обозначения. Пусть $\mathbb{N} = \{0,1,\ldots\}$, $\mathbb{N}^+ = \mathbb{N}\setminus\{0\}$, ε - пустое слово, $|v|$ - длина слова v, Δ^* - множество всех слов в алфавите Δ, $|\Delta|$ - число элементов множества Δ, $v[i]$ - i-й символ слова v ($v[i] = \varepsilon$ при $i > |v|$), v^R - зеркальное обращение слова v. Для векторов слов $\bar{v} = (v_1,\ldots, v_k)$, $\bar{u} = (u_1,\ldots,u_k)$ обозначим $\bar{v}\cdot\bar{u} = (v_1\cdot u_1,\ldots, v_k\cdot u_k)$, $\bar{v}[i] = (v_1[i],\ldots,v_k[i])$, $\bar{v}^R = (v_1^R,\ldots, v_k^R)$.

ОПРЕДЕЛЕНИЕ 1. R^k-схема есть упорядоченная восьмерка

$$A = (K, \Sigma, \Delta, \Gamma, H, q_0, F, Z_0),$$

где K - конечное множество (состояний); Σ - конечное множество (меток); Δ - конечный алфавит (функциональных символов); Γ - конечный алфавит (магазинных символов); H - конечное множество помеченных продукций вида $(\sigma)(\xi,б) \to a\eta\bar{w}$, где $\sigma\in\Sigma$, $\xi\in K\setminus F$, $\eta\in K$, $б\in(\Gamma)^k$, $a\in\Delta\cup\{\varepsilon\}$, $\bar{w}\in(\Gamma^*)^k$; $q_0\in K$ (начальное состояние); $F\in K$ (множество заключительных состояний); $Z_0\in\Gamma$ (магазинный маркер). При этом для любых $\sigma\in\Sigma$, $\xi\in K\setminus F$, $б\in(\Gamma)^k$ существует не более одной продукции вида $(\sigma)(\xi,б)\to a\eta\bar{w}$ из множества H и при этом $б = (b_1,\ldots, b_k)$ и $\bar{w} = (w_1,\ldots, w_k)$ таковы, что для всех $i = 1,\ldots,k$, если $b_i\in\Gamma\setminus\{Z_0\}$, то $w_i\in(\Gamma\setminus\{Z_0\})^*$, а если $b_i = Z_0$, то $w_i\in(\Gamma\setminus\{Z_0\})^*\cdot\{Z_0\}$.

Элементы множества $Conf(A) = \Delta^* \times K \times ((\Gamma \setminus \{Z_0\})^* \{Z_0\})^k$ называются конфигурациями, а множества $Conf'(A) = \Delta^* \times K \times \Gamma^k$ - частичными конфигурациями (конфигурации $(u, \xi, \bar{v})$, где $\bar{v} = б \cdot \bar{v}'$, соответствует частичная конфигурация $(u, \xi, б)$). Предполагая, что $K \cap \Gamma = K \cap \Delta = \Gamma \cap \Delta = \emptyset$, будем наряду с записью $(u, \xi, \bar{v})$ пользоваться записью $u \xi \bar{v}$.

(Σ, Δ)-разметкой называется всюду определенное однозначное отображение $\mu : \Delta^* \to \Sigma$. Разметка μ на множестве $Conf(A)$ определяет бинарное отношение (непосредственной выводимости) $\vdash$ следующим образом: $u \xi \bar{v} \vdash u_1 \xi_1 \bar{v}_1$, если $\mu(u) = \sigma$, $\bar{v} = б \cdot \bar{v}'$, продукция $(\sigma)(\xi, б) \to a \xi_1 \bar{w}$ принадлежит H и $u_1 = u \cdot a$, $\bar{v}_1 = \bar{w} \cdot \bar{v}'$. Отношение $\vdash^*$ есть рефлексивно-транзитивное замыкание отношения $\vdash$. Последовательность конфигураций $\pi_1 \vdash \ldots \vdash \pi_1$ называется выводом R^k-схемы A. Результат применения R^k-схемы A к слову $x \in \Delta^*$ при разметке μ обозначим $A(x, \mu)$. Значение $A(x, \mu)$ определено, если для некоторых $\alpha \in \Delta^*$, $p \in F$, $\bar{\beta} \in ((\Gamma \setminus \{Z_0\})^* \{Z_0\})^k$ выполняется условие $(x, q_0, (Z_0, \ldots, Z_0)) \vdash^* \alpha p \bar{\beta}$. В этом случае $A(x, \mu) = \alpha$, и мы будем говорить, что R^k-схема A начинает работу в вершине x и останавливается в вершине α при содержимом магазинов $\bar{\beta}$. В противном случае значение $A(x, \mu)$ не определено. Считаем, что $L(A) = \{ \mu \mid \mu$ - разметка, $A(x, \mu)$ определено$\}$. Слово в алфавите Δ будем также называть вершинами, а разметку $\mu : \Delta^* \to \Sigma$ - размеченным деревом, считая, что вершины $u, v \in \Delta^*$ связаны ребром, если $v \in u \cdot \Delta$ или $u \in v \cdot \Delta$. Можно считать, что R^k-схема есть магазинный автомат над полным $|\Delta|$-арным размеченным деревом ((Σ, Δ)-деревом).

ОПРЕДЕЛЕНИЕ 2. R^k-схема A называется $FTR(m_1,\ldots,m_k)$-схемой, если для любых $1 \le j \le k$ (Σ,Δ)-разметки μ и определяемого ею вывода $(\varepsilon,q_0,(Z_0,\ldots,Z_0)) = \pi_1 \vdash \ldots \vdash \pi_l$, $\pi_i = (u_i,\xi_i,(v_i^1,\ldots,v_i^k)),1 \le i \le l$, нельзя указать числа $1 \le r_1 < \ldots < r_{2m_j+3} \le l$ такие, что

$$|v^j_{r_{2i+1}}| < |v^j_{r_{2i}}| > |v^j_{r_{2i+1}}| \text{ для всех } 1 \le i \le m_j+1.$$

Иначе говоря, $FTR(m_1,\ldots,m_k)$-схема A в любом выводе совершает не более m_j поворотов по магазину j.

R^1-схему будем называть просто R-схемой. R^1-схема A называется FTR-схемой (конечноповоротной R-схемой), если она является $FTR(m)$-схемой для некоторого $m \in \mathbb{N}$. Несложный анализ показывает, что по любому автомату с магазинной памятью можно проверить такие свойства: а) имеет ли он m поворотов при некотором поведении, б) является ли он вообще конечноповоротным. Первое следует из разрешимости проблемы пустоты в классе контекстно-свободных языков [6]. Второе легко выводится из рассмотрения факторизаций поведения МП-автоматов (см., например, [7] и ниже леммы 1,2). Соответственно по любой R-схеме можно эффективно определить, будет ли она $FTR(m)$-схемой при данном m и будет ли она, вообще, FTR-схемой. Следовательно, класс FTR-схем - эффективный подкласс класса R-схем. Переходя к бóльшему числу магазинов, можно заметить, что из неразрешимости проблемы соответствий Поста (см. [8]) легко следует, что проблема пустоты $L(A) = \emptyset$ неразрешима уже для $FTR(1,1)$-схем. Соответственно класс $FTR(1,1)$-схем не эффективный подкласс класса R^2-схем.

Следуя [7], введем по аналогии термины, используемые ниже при описании процессов, моделирующих выводы (поведения) R-схемы A. Характеризуя конфигурацию $u\xi v$, будем говорить,

что схема $\mathbf{A}$ находится в вершине u (обозревает вершину u), в состоянии ξ, имеет в магазине слово v (содержимое магазина), d - верхний символ магазина (обозреваемый магазинный символ), если $v = dv'$, $d \in \Gamma$. R-схема $\mathbf{A}$ переходит из конфигурации $u\xi v$ в конфигурацию $u_1\xi_1 v_1$ в результате машинной операции (продукции) $(\sigma)(\xi, b) \to a\xi_1 w$ (при $\mu(u) = \sigma$), которая называется магазинной операцией при $|w| > 1$, операцией записи при $|w| = 1$, операцией стирания при $|w| = 0$ $(w = e)$. Вместо $u\xi v - u_1\eta v_1$ при $\mu(u) = \sigma$ будем писать $u\xi v \overset{\sigma}{\vdash} u_1\eta v_1$. Если $u \leq u'$, $u' = ua_{j_1} \dots a_{j_k}$, $a_{j_1} \in \Delta$, $1 \leq i \leq k$, $k \geq 0$, то слово $\mu(u)\mu(ua_{j_1}) \dots \mu(ua_{i_1} \dots a_{j_k})$ обозначается через $\mu[u,u']$. Пусть $\pi_1 \vdash \dots \vdash \pi_k$, $\pi_i = u_i\xi_i v_i$, $1 \leq i \leq k$. Тогда будем писать $\pi_1 \overset{\alpha}{\vdash} \pi_k$, если $\alpha = \mu[u,u']$, и говорить, что R-схема $\mathbf{A}$, читая α, переходит из π_1 в π_k. Объекты вида $\xi_i v_i$ будем называть ξ-конфигурациями. Соответственно будем писать $\pi_1 \downarrow(\alpha) \pi_k$, $\pi_1 \downarrow \pi_k$, если $|v_1| < |v_i|$ при $1 < i \leq k$. Будем писать $\pi_1 \uparrow(\alpha) \pi_k$, $\pi_1 \uparrow \pi_k$, если $|v_1| > |v_i|$ при $1 \leq i < k$. Следующие две леммы приведем без доказательств, поскольку они почти в точности соответствуют леммам из [7]. Они относятся к выводу $\pi_1 \vdash \dots \dots \vdash \pi_k$, где $\pi_i = u_i\xi_i v_i$, $1 \leq i \leq k$.

ЛЕММА 1. *Если* $|v_1| < |v_k|$, *то существует число* $1 \leq t < k$ *такое, что* $|v_1| = |v_t|$, $\pi_t \downarrow \pi_k$.

ЛЕММА 2 (*фактор-свойство*).

(i) *Если* $\pi_1 \downarrow \pi_k$, $v = b'_m \dots b'_1 v_1$, $b'_j \in \Gamma$, $1 \leq j \leq m$, *то существуют числа* $1 = t_1 < \dots$ $\dots < t_m < k$ *такие, что* $v_{t_j} = b'_{t_{j-1}} \dots b'_1 v_1$, $2 \leq j \leq m$, $\pi_{t_j} \downarrow \pi_{t_{j+1}}$, $1 \leq j < m$, $\pi_{t_m} \downarrow \pi_k$.

(ii) *Если* $\pi_1 \uparrow \pi_k$, $v_1 = b'_m \ldots b'_1 v_k$, *то существуют числа* $1 = t_m < \ldots < t_1 < k$ *такие, что* $v_{t_j} = b'_{t_j} \ldots$ $\ldots b'_1 v_k$, $1 \le j \le m$, $\pi_{t_{j+1}} \uparrow \pi_{t_j}$, $1 \le j < m$, $\pi_{t_1} \uparrow \pi_k$.

Далее опишем класс **UFP**-схем (с конечным числом переменных, одноместными функциями и предикатами), более выразительный, чем класс **FTR**-схем в том смысле, что любая **FTR**-схема будет транслируема в некоторую **UFP**-схему.

ОПРЕДЕЛЕНИЕ 3. **UFP**-схема есть упорядоченная семерка

$$B = (K, \Sigma, \Delta, S, H, q_0, F),$$

где $K, \Sigma, \Delta, q_0, F$ имеют тот же смысл, что и в определении R^k-схемы; $S = \{x_1, \ldots, x_s\}$ - конечное множество переменных; H - конечное множество помеченных продукций вида

$$(\sigma)(\xi,i) \to (\eta,j)(k,l,a),$$

где $\sigma \in \Sigma$, $\xi \in K \setminus F$, $\eta \in K$, $1 \le i,j,k,l \le s$, $a \in \Delta \cup \{\varepsilon\}$, причем для любых $\sigma \in \Sigma$, $\xi \in K \setminus F$, $1 \le i \le s$, существует единственная продукция $(\sigma)(\xi,i) \to (\eta,j)(k,l,a)$ из H.

Элементы вида $(\alpha_1, \ldots, \alpha_s, q, i)$, $\alpha_i \in \Delta^*$, $q \in K$, $1 \le i \le s$, называются конфигурациями. Система разметок $\vec{\mu}: \Delta^* \to \Sigma^*$, $\vec{\mu}(x) = (\mu_1(x), \ldots, \mu_s(x))$, на множестве конфигураций определяет отношение выводимости $\mapsto$ так, что $(\alpha_1, \ldots, \alpha_s, \xi, i) \mapsto (\beta_1, \ldots, \beta_s, \eta, j)$, когда $\mu_i(\alpha_i) = \sigma$, продукция $(\sigma)(\xi,i) \to (\eta,j)(k,l,a)$ принадлежит H, $\beta_t = \sigma_t$ для $t \ne k$, $\beta_k = \alpha_i a$. Отношение $\mapsto^*$ есть рефлексивное транзитивное замыкание отношения $\mapsto$. Последовательность конфигураций $\pi_1 \mapsto \ldots \mapsto \pi_n$ называется выводом **UFP**-схемы B. Результат применения **UFP**-схемы B к вектору слов $\bar{v} = (v_1, \ldots, v_s)$, $v_i \in \Delta^*$, $i = 1, \ldots, s$, при системе разметок $\vec{\mu} = (\mu_1, \ldots, \mu_s)$ обозначим $B(\bar{v}, \vec{\mu})$. Значение $B(\bar{v},$

$\bar{\mu}$) определено, если выполняется условие $(v_1,\dots,v_s, q_0,1) \mapsto^*(y_1,\dots,y_s,p,j)$ для некоторых $y_i \in \Delta^*$, $i = 1,\dots,s$, $p \in F$, $1 \le j \le s$. В этом случае $B(\bar{v},\bar{\mu}) = y_1$ в соответствии с соглашением о том, что имеется одна выходная переменная (первая). В остальных случаях $B(\bar{v},\bar{\mu})$ не определено. Характеризуя конфигурацию $(w_1,\dots,w_s,q,j)$, будем говорить, что i-я переменная (переменная x_i) схемы B имеет значение w_i, а схема B находится в (обобщенном) состоянии (q,j).

Положим, что R^k-схема A транслируема в UFP-схему B, если для любых $x \in \Delta^*$ и разметки μ значения $A(x,\mu)$ и $B(\bar{v},\bar{\mu})$ совпадают, где $\bar{v} = (x,\varepsilon,\dots,\varepsilon)$, $\bar{\mu} = (\mu,\dots,\mu)$.

ТЕОРЕМА 1. *Класс* FTR*-схем транслируем в класс* UFP*-схем.*

ДОКАЗАТЕЛЬСТВО. Пусть A - любая FTR-схема, $A = (K,\Sigma,\Delta,\Gamma,H,q_0,F,Z_0)$. Как отмечалось выше, по A эффективно строится натуральное число m такое, что A - FTR(m)-схема. Необходимо по A эффективно построить UFP-схему B так, что на любой разметке μ при $x = x_1 = \varepsilon$ поведение FTR(m)-схемы A моделируется поведением UFP-схемы B. Вначале покажем, как моделируется поведение (вывод) $\varepsilon q_0 Z_0 = \pi_1 \vdash \dots \vdash \pi_k$ схемы A, на котором происходит ровно один полный поворот в магазине схемы A. Для этого построим UFP-схему B_1 с четырьмя переменными $x_1 = x$, $x_2 = y$, $x_3 = z$, $x_4 = t$. Моделируя схему A на разметке μ, схема B_1 останавливается, во-первых, когда схема A останавливается до совершения второго поворота и, во-вторых, когда схема A переходит к выполнению второго поворота. Сохраним использованные выше обозначения $\pi_i = u_i\xi_i v_i$, $1 \le i \le k$, $\alpha = \mu[u,u']$, $\pi_1 \overset{\alpha}{\vdash} \pi_k$. Не умаляя общности, будем считать, что в H нет команд вида $(\sigma)(\xi,b) \to \eta w$, $|w| \ge 1$, отличных от зацикливающих команд $(\sigma)(\xi,b) \to \xi b$.

Пусть d - наибольшее значение $|w|$ для продукций $(\sigma)(\xi,b)\to$ $\to a\eta w$ из H. По R-схеме A легко построить эквивалентную R-схему A', у которой $|w| = r$ для каждой ее магазинной операции. Не умаляя общности, будем считать, что указанным свойством обладает уже сама R-схема A. Кроме того, учитывая возможность укрупнения магазинных символов, будем считать, что $r = 2$.

UFP-схема $B_1 = (K_1, \Sigma, \Delta, S_1, H_1, q_0^1, F_1)$ такова, что $S_1 = \{x, y, z, t\}$, $K \times \Gamma \subseteq K_1$, $F \times \Gamma \subseteq F_1$. Доопределение параметров K_1, H_1, q_0^1, F_1 схемы B_1 соответствует указанному ниже описанию процесса функционирования схемы B_1 относительно разметки μ. Переменная t используется для хранения начального значения переменной x. Вначале выполняется оператор $t := x$, и схема B_1 из состояния q_0^1 переходит в начальное состояние (q_0, Z_0) схемы A_1 - подсхемы схемы B_1. Схема A_1 моделирует схему A до первой операции стирания. Соответственно если $((\sigma)(\xi,b) \to a\eta w) \in H$, $|w| \geq 1$, то $(\sigma)((\xi,b),1 \to a((\eta,c),1)$ - команда схемы A_1, где $w = cw'$, $c \in \Gamma$. Элементы из $F \times \Gamma$ есть одновременно заключительные состояния схем A_1, B_1 и B. Если $((\sigma)(\xi,b) \to a\eta) \in H$, то $(\sigma)((\xi,b),1) \to a((\eta,\bar{\varepsilon}),1)$ - команда схемы A_1, а $(\eta,\bar{\varepsilon})$ - ее заключительное состояние, где $\bar{\varepsilon} = \varepsilon$ при $\eta \notin F$ и $\bar{\varepsilon} = Z_0$ при $\eta \in \Gamma$. Заметим, что схема A_1 на каждом такте переходит на следующую вершину, кроме, быть может, последнего такта.

Пусть схема A_1 попала в состояние (p,ε), $p \in K \setminus F$, и остановилась в вершине γ_1. После этого схема B_1 выполняет операторы $y := t$; $z := t$; $A_1^y\downarrow$ (см.ниже) и переходит к выполнению схемы A_2. Работа схемы A_2 состоит в параллельном и чередующемся потактном выполнении схем A_1^y, A_1^z, сопровождаемом

проверкой корректирующих условий. Схема A_1^y (то же для A_1^z) есть схема A_1, поставленная на обработку переменной y (соответственно z) вместо x. Пусть $A_1^y\downarrow$ обозначает: "выполнить очередную команду схемы A_1^y" (аналогично для $A_1^z\downarrow$). Прежде чем задать схему A_2, приведем нестрогое пояснение принципа ее работы. Прогоном A_1 по y (посредством A_1^y) и отстающим от него на 1-2 такта прогоном A_1 по z схема A_2 пытается найти магазинный символ (d), лежавший под символом, стертым FTR-схемой A. Если это не удается в один прогон (до попадания A_1^y в состояние (p,ε)), то производятся новые прогоны (после $y:= t;\ z:= t$) с отставанием A_1^z от A_1^y еще на один такт при каждом следующем прогоне. Найдя искомый символ (d), схема A_2 далее моделирует FTR-схему A. Если при моделировании найденный символ (d) стирается, то новые прогоны A_1 по y,z позволяют найти магазинные символы, лежащие еще ниже, и продолжить моделирование FTR-схемы A.

Схема A_2.

```
    y:= t;
L:  A1^y↓ ;
    IF (¬(A1^y попала в (p,ε))
    THEN {A1^z↓: IF (A1^z попала в (p,ε)) THEN z:= t;
    GOTO L;}
    ELSE {IF (¬ Условие МО) THEN (y:= t; GOTO L;}
         ELSE { схема A20;
              IF (A20 попала в (q,ε), q∈ K \F)
                 THEN {y:= t; GOTO L;}}
```

Здесь условие $M0$ означает, что предстоящая команда $A_1^z\downarrow$ есть магазинная операция $(\sigma)(\xi,b) \to a\eta w$, где $|w| = 2$, $w = dc$. Символ d используется в работе схемы A_{20}, которая начинается в состоянии (p,d) и обрабатывает x подобно схеме A_1. Более точно, если $((\sigma)(\xi,b) \to a\eta w) \in H$, $|w| = 1$, то $(\sigma)((\xi,b),1) \to a((\eta,w),1)$ - команда схемы A_{20}, причем (η,w) - заключительное состояние схем A_{20}, A_2, B_1 при $|w| = 2$ или $\eta \in F$. Если $((\sigma)(\xi,b) \to a\eta) \in H$, то $(\sigma)((\xi,b),1) \to a((\eta,\bar{\varepsilon}),1)$ - команда схемы A_{20}, где $\bar{\varepsilon} = \varepsilon$ при $\eta \notin F$ и $\bar{\varepsilon} = Z_0$ при $\eta \in \Gamma$. Если схема A_{20} попала в состояние (q,ε), $q \in K \setminus F$, то схема A_2 начинает новый цикл поиска верхнего магазинного символа, причем поскольку p заменится на q при новом запуске подсхемы A_{20}, последняя начнет в состоянии (q , новое d) и далее будет в последующих циклах начинать соответственно в новых состояниях. Описание процесса функционирования схемы B_1 на этом закончено.

Пусть схема A_{20} попала в состояние $(\bar{p},w_0)$, $p \in K \setminus F$, $|w_0| = 2$, и остановилась в вершине γ_2. Заметим,что вершина γ_2 еще не обозревалась FTR-схемой A и схемой B_1. Соответственно в магазине моделируемой FTR-схемы A содержится слово $w_0\delta \in \Gamma^*Z_0$, $\delta = \delta_n \dots \delta_1 Z_0$ и символы δ_1, $n \le i \ge 1$, могут быть при необходимости извлечены новыми запусками схемы A_2 при возникновении запроса, определяемого ниже UFP-схемой B_2. Последняя определяет и состояние p, исходное при включении подсхемы A_{20}.

Продолжим описание поведения схемы B. Это описание имеет индуктивный характер. Число возможных поворотов в магазине при дальнейшем поведении $FTR(m)$-схемы A уже не больше $m-2$. Поэтому по предположению индукции имеем UFP-схему B_2, точнее, схему $B_2(\bar{p},w_0)$, которая моделирует FTR-схему A, установленную в вершине γ_2 в состоянии $\bar{p}$ с со -

держимым магазина w_0Z_0. UFP-схема B_2, кроме переменных x,y, имеет еще дополнительные переменные x_i, $5 \le i \le 2+s(m-2)$, где $s(n)$ - число переменных, необходимое для моделирования поведения FTR-схемы A с n поворотами. Здесь возможны два случая: а) в процессе дальнейшей работы FTR-схема A, начавшая с магазином w_0Z_0, никогда не приходит к пустому магазину, содержащему только Z_0; б) она приходит к пустому магазину в состоянии p (вообще говоря, отличном от состояния p в описании схемы B_1).

В случае "а" схема $B_2(\bar{p},w_0)$ сама произведет до конца моделирование FTR-схемы A и выдаст эквивалентный результат по переменной x. В случае "б" значения по переменным x_i, $5 \le i \le 2+s(m-2)$, и переменной y схемы B_2 стираются (например, операторами $y:=t$; $x_i:=t$, $5 \le i \le 2+s(m-2)$), возникает запрос схемы B_2 к схеме A_2 по поводу магазинного символа δ_n, и включается схема A_2. Для ответа на запрос достаточно значений переменных z,t, поэтому переменная y используется как общая рабочая переменная подсхем A_2 и B_2. Дальнейшее моделирование FTR-схемы A или завершится схемой A_2, или ее подсхема A_{20} попадет в новое состояние $(\bar{p},w_0)$. Тогда снова включается схема $B_2(\bar{p},w_0)$ (с новыми $\bar{p},w_0$) и повторяются описанные выше действия. Для функции s имеем такие оценки: $s(0) = 1$, $s(1) = s(2) = 4$, $s(2i-1) = s(2i) = 2i+2, i \in \mathbb{N}^+$. На самом деле, число переменных, используемое при моделировании поведения ρ FTR(m)-схемы A, зависит не только от m, сколько от вида графика изменения высоты магазина схемы A при поведении ρ. Например, схема B может, последовательно переключая свои подсхемы A_2, B_2, любое конечное число раз промоделировать некоторое поведение ρ R-схемы A (даже не конечноповоротной R-схемы A), если каждое включение подсхемы B_2 моделирует одноповоротный фрагмент поведения ρ.

Iогда в поведении ρ может быть сколько угодно поворотов,но **UFP**-схема **B** при его моделировании использует только шесть переменных. Развивая последнее соображение, получим в виде следствия 1 некоторое обобщение теоремы 1.

СЛЕДСТВИЕ 1. *Пусть существует число* $m \in \mathbb{N}^+$ *такое, что для любого вывода* $\pi_1 \vdash \ldots \vdash \pi_k$ **R**-*схемы* **A** *найдется не более* **m** *чисел* $1 \le i_1 < \ldots < i_m \le k$, *удовлетворяющих условиям* $\pi_{i_j} \downarrow \pi_{i_{j+1}}$, $1 \le j < m$, *и таких, что в каждом выводе* $\pi_{i_j} = \pi_{i_{j+1}}$, $1 \le j < m$, **R** -*схема* **A** *совершает по крайней мере один поворот в магазине. Тогда схема* **A** *транслируема в эквивалентную* **UFP**-*схему* **B** *с* $2m$ *переменными.*

СЛЕДСТВИЕ 2. *Класс металинейных унарных рекурсивных схем с засылками констант транслируем в класс стандартных схем с засылками констант,имеющих четыре переменные.*

Из следствия 1 вытекает также теорема о транслируемости квазирациональных рекурсивных схем в эквивалентные стандартные схемы, установленная в [5]. Класс квазирациональных рекурсивных схем есть минимальный класс, содержащий линейные унарные рекурсивные схемы и замкнутый относительно функциональной подстановки. Аналогично относительно линейных унарных рекурсивных схем с засылками констант определим класс квазирациональных рекурсивных схем с засылками констант. Тогда из следствия 1 вытекает

СЛЕДСТВИЕ 3. *Класс квазирациональных рекурсивных схем с засылками констант транслируем в класс стандартных схем с засылками констант.*

Множество $\{A^{\#}(\varepsilon,\mu) \mid \mu$ - разметка , $A(\varepsilon,\mu)$ определено$\}$ назовем языком интерпретированных значений **R**-схемы **A**, где при $A(\varepsilon,\mu) = a_1 \ldots a_n$, $a_i \in \Delta$, $1 \le i \le n$, интерпре-

тированное значение есть $\sigma_0 a_1 \sigma_1 \dots a_n \sigma_n$, $\sigma_0 = \mu(\varepsilon)$, $\sigma_i = \mu(a_1 \dots a_i), 1 \le i \le n$. Легко задать FTR(1)-схему A_0, имеющую при $\Sigma = \{0,1\}$, $\Delta = \{a,b,c\}$ язык интерпретированных значений

$$L_0 = \{1(a0)^n a1 a1 (b0)^n,$$

$$1(a0)^{n_1} a1 (a0)^{n_2} a1 (c0)^n \mid n = n_1 + n_2,\ n_1, n_2 \in \mathbb{N}^+\}.$$

Этот язык не является языком интерпретированных значений никакой квазирациональной рекурсивной схемы. Пусть FTR(1)-схема A_0 имеет множество состояний $K = \{q_0, \dots, q_k\}$, $\tilde{q}_i \in \{a0,a1\}^{\lfloor \log_2 \langle k \rangle \rfloor + i}$ - код состояния q_i, $\tilde{q}_i \ne \tilde{q}_j$, при $0 \le i < j \le k$. Возьмем язык $L_1 = \{1 v_0 \tilde{f}_1 \dots v_{2n+2} \tilde{f}_{2n+2} \mid v_i = \alpha_i \beta_i$, $\alpha_i \in \{a,b,c\}$, $\beta_i \in \{0,1\}$, $1 \le i \le 2n+2$, $f_0 = q_0$, $1 v_1 \dots v_{2n+2} \in L_0$, $\langle f_0, f_1, \dots, f_{2n+2} \rangle$ - последовательность состояний, возникающих при обработке FTR(1)-схемой A_0 разметки μ, где $\mu(\varepsilon) = 1$, $\mu(\alpha_1 \dots \alpha_i) = \beta_i$, $1 \le i \le 2n+2\}$. Тогда L_1 есть язык интерпретированных значений унарной рекурсивной схемы A^{01}, совершающей один поворот в магазине в процессе работы, но язык L_1 не является языком интерпретированных значений никакой квазирациональной рекурсивной схемы.

СЛЕДСТВИЕ 4. *В классе унарных рекурсивных схем с одним поворотом в магазине есть не квазирациональные рекурсивные схемы.*

Таким образом, следствие 1 определяет класс унарных рекурсивных схем, транслируемый в класс стандартных схем и строго включающий класс квазирациональных схем, рассмотренный в [5].

ТЕОРЕМА 2. *Класс стандартных схем с одним конечноповоротным магазином транслируем в класс стандартных схем.*

ДОКАЗАТЕЛЬСТВО. В теореме 1 фактически доказывалось, что относительно свободных интерпретаций класс схем Янова с одним конечноповоротным магазином транслируем в класс стандартных схем с дополнительными ограничениями (унарность функций и предикатов). При том же способе доказательства могут быть расширены возможности транслируемой схемы A вместе с соответствующим расширением возможностей схемы B, в которую транслируется схема A. Так, разделение переменной x на несколько переменных $x^1, \ldots, x^k$ в схеме A будет сопровождаться соответствующим разделением переменных x (на $x^1, \ldots, x^k$) и t на $t^1, \ldots, t^k$. Использование многоместных функциональных и предикатных символов в схеме A сопровождается использованием их в схеме B. Это указывает способ переноса доказательства теоремы 1 для получения новых результатов. Следовательно, стандартные схемы с конечноповоротным счетчиком транслируемы в стандартные схемы.

ТЕОРЕМА 3. *Класс* $\mathbf{FTR}(m_1, \ldots, m_k)$*-схем транслируем в класс* **UFP***-схем.*

ДОКАЗАТЕЛЬСТВО. По $\mathbf{FTR}(m_1, \ldots, m_k)$-схеме A необходимо построить эквивалентную **UFP**-схему B'. По аналогии с доказательством теоремы 1 приведем индуктивное описание работы **UFP**-схемы B'. Вначале на первом этапе работает схема A'_1, действующая с переменными t, x, моделирующая схему A подобно схеме A_1, указанной выше, до тех пор, пока схема A осуществит, по крайней мере, один поворот по одному из магазинов. Исходное значение переменной x остается в t. Далее, в подсхеме A'_2 есть новые переменные y^j, z^j (соответственно по две переменные для каждого магазина), $1 \le j \le k$. Пусть первый этап промоделировал поведение схемы A, которое привело к состоянию p схемы A и содержимому ее магазинов $\bar{w} = (w_1, \ldots, w_k)$. Второй этап начинается подсхемой A'_2 схемы B', обеспечивающей аналогично схеме A_2, указанной

выше, нахождение магазинного символа, лежащего под символом, стертым $FTR(m_1, \ldots, m_k)$-схемой по соответствующему магазину. Если стирание произошло одновременно по нескольким магазинам, то A_2' находит символ, лежавший под стертым символом по каждому магазину. Схема A_2' есть начальная подсхема схемы B_2', которая аналогично схеме B_2, указанной выше, моделирует оставшееся поведение $FTR(m_1, \ldots, m_k)$-схемы A, в котором уже для некоторого $1 \le j \le k$ число поворотов по j-му магазину меньше, чем m_j. Поэтому по предположению индукции соответствующая схема B_2' может быть построена. При возникновении запроса на лежащие под стертыми символами новые магазинные символы, поступившие в магазины на первом этапе, схема B_2' снова включает свою подсхему A_2'. Найдя искомые магазинные символы, схема B_2' продолжает моделирование $FTR(m_1, \ldots, m_k)$-схемы A.

Пусть, например, все поведение схемы A с двумя магазинами разбито на три этапа так, что высота первого магазина возрастает на первом этапе, убывает на втором и третьем этапах; высота второго магазина возрастает на первом и втором этапах, убывает на третьем этапе. Опишем поведение моделирующей схемы B'. На первом этапе после выполнения оператора $t := x$ схема A моделируется схемой A_1'. На втором этапе включается подсхема B_2'. Здесь решается задача отыскания верхних символов первого магазина и моделируется движение вниз по первому магазину. Для этого, как и в схеме A_2, используются дополнительные переменные y, z. На третьем этапе дополнительно решается задача отыскания верхних символов второго магазина, поступивших в него на втором этапе. Для этого используется значение переменной x к началу второго этапа (оно запоминается переменной t^1), дополнительные переменные y^1, z^1, на которых производится прогон второго этапа по первому магазину, и переменные y^2, z^2, служащие для прогона по второму

магазину. Переменные y, y^1, y^2 могут быть совмещены, так как прогоны по разным магазинам можно проводить поочередно. Всего для моделирования рассмотренного поведения FTR(1,1)-схемы A понадобится семь переменных $x, t, t^1, y, z, z^1, z^2$ UFP-схемы B'.

В общем случае для подсчета числа переменных, используемых в схеме B', разобьем на этапы поведение схемы A относительно фиксированной разметки μ и соответственно процесс моделирования его UFP-схемой B. Как уже указано выше, первый этап поведения схемы A заканчивается первым поворотом по одному (или одновременно нескольким) из магазинов или остановом схемы A. Последний такт первого этапа считается первым тактом второго этапа. Пусть $t_1 = 1$ и для $i = 1,2,\ldots$ число t_{i+1} - номер первого такта $(i+1)$-го этапа и последнего такта i-го этапа. Тогда t_{i+1} - минимальное число $t_{i+1} > t_i$ такое, что при поведении схемы A на тактах $t_1, \ldots, t_{i+1}$ совершается один поворот по одному из магазинов или происходит останов схемы A. Пусть m - общее число этапов. Тогда $m \le 1 + \sum_{j=1}^{k} m_j$. На первом этапе в схеме B' используются только две переменные x, t. На втором этапе дополнительно используются переменные y^j, z^j для всех $1 \le j \le k$ таких, что произошел поворот по j-му магазину. При этом все переменные типа y^j можно отождествить с одной переменной y, так как и одной переменной y будет достаточно, чтобы обеспечить поочередные прогоны A'_1 по y и (с отставанием) по z^j, $1 \le j \le k$. Значит, на втором этапе необходимо не более чем $k+3$ переменных $x, t, y, z^1, \ldots, z^k$. При этом переменная y опустошается при завершении этапа и может быть использована свободно на следующем этапе. Появление на последующих этапах дополнительных переменных типа z^j, определяющих своими значениями куски слова, хранящегося в j-м ма-

газине, соответствует факторизации поведения схемы A относительно j-го магазина, указанной в следствии 1. Для любого j на i-м этапе число переменных типа z^j не больше, чем число колебаний "вверх-вниз" высоты j-го магазина на этапах $1,\ldots,i$. Запоминаются также значения переменной x, начальные для каждого этапа. Это производится переменными типа t. Число дополнительных переменных типа y, связанных с прогонами на $(i+1)$-м этапе, равно числу переменных типа z плюс число переменных типа y^z, представляющих копии переменных типа z, возникших на этапах $1,\ldots,i$. После завершения очередного этапа переменные типа y, y^z опустошаются. Таким образом, схема B' использует m переменных типа x, $\sum_{j=1}^{k}\left[\frac{m_j+1}{2}\right]$ переменных типа z и столько же переменных типа y, y^z, где $m \le 1 + \sum_{j=1}^{k} m_j$. Значит, схема B' использует не более чем $3k(M+1)$ переменных, где $M = \max\{m_j \mid 1 \le j \le k\}$.

Соединяя доказательства теорем 2,3, получаем такой результат.

ТЕОРЕМА 4. *Класс стандартных схем с конечноповоротными магазинами транслируем в класс стандартных схем.*

ЗАМЕЧАНИЕ 1. Здесь подразумевается, что по любой стандартной схеме A с магазинами, для которой известно максимальное число M поворотов в магазинах, эффективно строится эквивалентная стандартная схема.

Стандартная схема S со стеками имеет, по определению, конечноповоротные стеки, если при любом поведении возрастание высоты читающей головки i-го стека сменяется на убывание не более m_i раз. Легко убедиться, что стек может быть промоделирован парой магазинов, при этом обычное стирание и запись

моделируется одним из магазинов, а движение читающей головки вниз и вверх по стеку - переносом символов из основного мага - зина во вспомогательный и обратно. При данном способе модели - рования конечноповоротному стеку соответствуют два конечнопо - воротных магазина. Получаем следующий обобщающий результат.

ТЕОРЕМА 5. *Класс стандартных схем с конечноповоротными стеками транслируем в класс стандартных схем.*

Дальнейшее изложение посвящено формальному описанию конструкций **UFP**-схем, моделирующих **FTR(m)** и **FTR(m_1... ..., m_k)**-схемы.

Опишем строго подкласс R^k-схем, характеризующийся таким свойством, что при выполнении любой продукции высота любого магазина изменяется не более чем на единицу.

Будем называть приведенной R^k-схему **A**, удовлетво - ряющую следующему свойству: для любой продукции $(\sigma)(\xi,б) \to$ $\to a\,\eta\bar{w} \in H, б = (b_1,\ldots,b_k)$, $\bar{w} = (w_1,\ldots,w_k)$ выполняется

$$w_i \in \{\varepsilon\} \cup \Gamma \cup \Gamma\{b_i\}.$$

ЛЕММА 3. *По любой* R^k*-схеме* **A** *может быть построена эквивалентная ей приведенная* R^k*-схема* **A'**.

ДОКАЗАТЕЛЬСТВО. Пусть $(\sigma)(\xi,б) \to a\,\eta\bar{w} \in H$ - произволь - ная продукция, $б = (b_1,\ldots,b_k)$, $\bar{w} = (w_1,\ldots,w_k)$, $l = \max\{|w_i|, i = 1,\ldots,k\}$, $l \geq 2$. Рассмотрим векторы

$$б^1 = \bar{w}^R[1],\ldots, б^l = \bar{w}^R[l].$$

Расширим множество состояний схемы **A** состояниями $[\xi,б,1],\ldots,[\xi,б,l-1]$, а продукцию $(\sigma)(\xi,б) \to a\,\eta\bar{w}$ заменим множеством продукций

$$(\sigma)(\xi,б) \to (\varepsilon,[\xi,б,1],б^1),$$

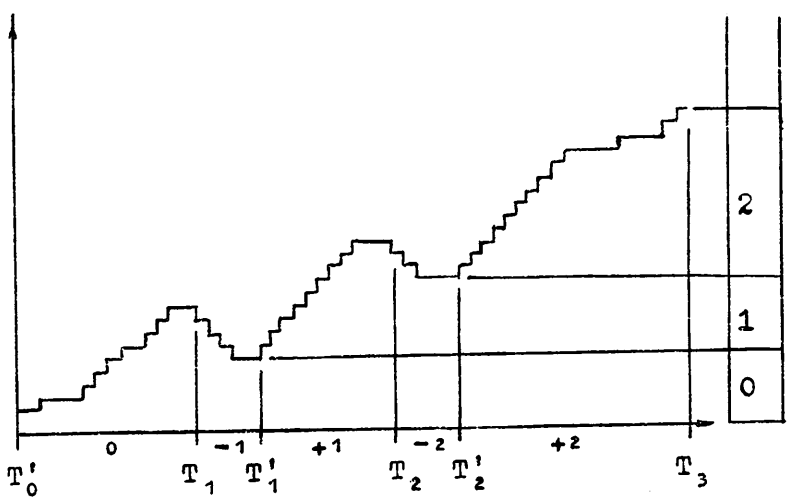
$$(\sigma)([\xi,б,i],\bar{u}) \to (\varepsilon,[\xi,б,i+1],б^{i+1}\bar{u}),\ i = \overline{1,l-2},$$

$(\sigma)([\xi, б, l-1], \bar{u}) \to (a, \eta, б^{1}\bar{u})$,

где $\bar{u}$ пробегает все $(\Gamma)^{K}$. Проделав это для всех продукций из H, нарушающих свойство из условия леммы, получим схему A', которая отличается от A только множеством состояний и системой команд. Легко видеть, что A' является приведенной. Кроме того, устройство системы команд A' таково, что $\pi_1 \vdash_A \vdash_A \pi_2 \Leftrightarrow \pi_1 \vdash_{A'}^{*} \pi_2$, где π_1, π_2 - произвольные конфигурации A (а значит, и A'). Лемма доказана.

Для $FTR(m_1, \ldots, m_K)$-схемы A приведенная схема A', построенная указанным в доказательстве леммы 3 способом, также является $FTR(m_1, \ldots, m_K)$-схемой. Поэтому везде в дальнейшем все R^{k}-схемы будут предполагаться приведенными.

Введем еще несколько понятий. На приведенном рисунке представлен график изменения высоты одного из магазинов на некотором вычислении R^{k}-схемы.



Точку T_i, $i = 1,2,\ldots$, будем называть началом i-го поворота в магазине. На участке $[T_i, T'_i]$ будем говорить, что ма-

газин находится на спуске i-го поворота, на участке $[T'_i, T_{i+1}]$ - на подъеме. Поворот 0 не имеет участка спуска. (T_i - состояние схемы перед выполнением операции стирания, T'_i - перед выполнением магазинной операции.)

Опишем вспомогательную нотацию, более приближенную к традиционным языкам программирования, которая будет использована для описания результата трансляции. Мы будем использовать составные имена вида $p_1, p_2, \ldots, p_n$ для переменных и меток операторов. Любой программный фрагмент (или блок) будет состоять из описания переменных и/или оперативной части. Описания переменных выглядят следующим образом $p: V$, где p - имя переменной, V - множество ее значений. В качестве V может выступать любое конечное множество (управляющие переменные) или Δ^* (рабочие переменные). Операторная часть блока состоит из последовательности (помеченных) базовых операторов, отделенных точкой с запятой.

Базовыми операторами являются:

1. Оператор присваивания вида

$x := ya$, где x, y – рабочие переменные, $a \in \Delta \cup \{\varepsilon\}$,

или

$x := f(x_1, \ldots, x_n)$, где $x, x_1, \ldots, x_n$ - управляющие переменные, f - произвольная всюду определенная функция, согласованная с $x, x_1, \ldots, x_n$ по типам значений.

2. Оператор перехода

goto l, где l - метка некоторого оператора.

3. Пусть оператор

skip (является эквивалентом $x := x$ и используется только из методических соображений).

4. Оператор ветвления вида

if $p(x_1, \ldots, x_n)$ then l_1 {else l_2}

или

if $\sigma(x)$ **then** l_1 {**else** l_2} , где $x_1, \ldots, x_n$ - управляющие переменные, p - всюду определенный предикат, x - рабочая переменная, $\sigma \in \Sigma$, l_1, l_2 - метки операторов. Часть, заключенная в фигурные скобки, может отсутствовать.

5. Оператор останова **stop** .

Помеченный оператор - это оператор, которому предшествует метка, отделенная двоеточием. Метка должна быть уникальной в пределах данного фрагмента.

В целях компактности записи мы будем использовать более широкий набор обозначений, являющихся сокращениями для типич - ных конструкций базового набора. В частности, мы будем вводить широкий набор обозначений, являющихся сокращениями для типич - ных конструкций базового набора. В частности, мы будем вводить массивы переменных при помощи описаний вида `p[1:k]: V` , где `k` - фиксированная целочисленная константа. Эта запись будет служить сокращением для последовательности определений вида `p.1: V; p.2: V;...; p.k: V.` Возможно также использова - ние массивов операторов следующего вида: `(p1[1:k]: V ; p2[1:k]: V; i: {1...k}; active[1:1]:{1...k};);`

1) `p1:= p2` является сокращением для `p1.1:=p2.1;... ...; p1.k:= p2.k;`

2) **for** `i`$\in${`1...k`}**do** `F` **end**, где `F = [...p1[i]... ... l:... ]` - некоторый фрагмент, содержащий вхождения переменной `p1[i]` и метки `l:` , является сокращением для $F_1; \ldots; F_k$, , где F_j , $j = 1, \ldots, k$, - это копия `F` , в которой каждое вхождение `p1[i]` заменено на `p1.j` , а всякое вхождение метки `l` на `l.j` (что обеспечивает ее уни - кальность);

3) `F = [...p1[active[i]]...]` может рассматриваться как сокращение для

```
j: {1...k}; m: {1...k};
for j∈ {1...k} do
    if j ≠ i then next j;
    for m∈ {1...k} do
        if m ≠ active.j then next m;
            [... p1.m ...]
    next m: skip;
    end
next j: skip;
end
```

Пусть **F** - некоторый блок, **p** - некоторое имя. Переименованием блока, обозначаемым **p.F**, назовем копию блока **F**, в которой каждое вхождение любой переменной **V** заменяется на переменную **p.V**. Переименование позволит создавать копии одного и того же фрагмента, обеспечивая уникальность имен переменных. Будем употреблять переименованные фрагменты внутри других фрагментов, что будет обозначать текстуальную подстановку. При этом, естественно, будет требоваться уникальность меток операторов и описаний переменных после выполнения подстановки. Это будет достигаться следующим образом: будем выделять описания переменных в отдельный фрагмент, который будет подставляться только один раз. Фрагменты, не содержащие описаний переменных, могут иметь несколько вхождений в тело другого фрагмента. При этом будем предполагать, что выполняется соответствующее переименование меток внутри каждой копии фрагмента, обеспечивающее глобальную уникальность меток. (Передачи управления внутри фрагмента сохраняются, вход и выход в подставляемом фрагменте осуществляется через начало и конец фрагмента.)

Остановимся на вопросе о реализуемости данной нотации средствами **UFP** -схем. Пусть блок

$$Fr = [x_1,\ldots,x_n : \Delta^* : c_1 : V_1 ; \ldots ; c_l : V_l ; s_1 ; \ldots ; s_m]$$

состоит из описаний рабочих переменных $x_1,\ldots,x_n$, управляющих переменных $c_1,\ldots,c_l$ и последовательности помеченных операторов $s_1,\ldots,s_m$. Реализующая **UFP**-схема может быть построена следующим образом:

$$B = (K,\Sigma,\Delta,S,H,q_0,F),$$

где $S = \{x_1,\ldots,x_n\}$; $K = \bar{V}_1 \times \ldots \times \bar{V}_l \times \{1,\ldots,m\}$, $\bar{V}_i = V_i \cup \{\perp\}$, $\perp$ - символ неопределенного значения, Σ,Δ имеют обычный смысл, $q_0 = \langle \perp,\ldots,\perp,1\rangle$, $F = \{\langle v_1,\ldots,v_l,m'\rangle \mid s_{m'} = \underline{stop};\ v_i \in \bar{V}_i,\ i = \overline{1,l}\}$.

Множество команд H определяется следующим образом:

1. Оператор s_j вида $x_p := x_q \cdot a$ реализуется множеством команд вида

$$\{(\sigma)(\langle \bar{v},j\rangle,i) \to (\langle v,j+1\rangle,i)(p,q,a) \mid \sigma \in \Sigma, 1 \le i \le n,\ \bar{v} \in \bar{V}_1 \times \ldots \times \bar{V}_l\}.$$

2. Оператор s_j вида $c_{l_0} := f(c_{l_1},\ldots,c_{l_t})$ реализуется командами

$$\{(\sigma)(\langle \bar{v},j\rangle,i) \to (\langle \bar{v}',j\rangle,i)(i,i,\varepsilon) \mid \sigma \in \Sigma,\ 1 \le i \le n,$$

$$\bar{v} = (v_1,\ldots,v_l) \in \bar{V}_1 \times \ldots \times \bar{V}_l,\ \bar{v}' = (v_1',\ldots,v_l'),$$

$$v_r' = v_r,\ r \ne l_0,$$

$$v_{l_0}' = f(v_{l_1},\ldots,v_{l_t})\}.$$

3. Оператор s_j вида goto l, где l - метка оператора $s_{j'}$, реализуется командами

$$\{(\sigma)(\langle\bar{v},j\rangle,i)\to$$

$$(\langle\bar{v},j'\rangle,i)(i,i,\varepsilon)\,|\,\sigma\in\Sigma,\ 1\le i\le n,\ \bar{v}\in\bar{V}_1\times\ldots\times\bar{V}_l\}.$$

4. Оператор s_j вида if $p(c_{l_1},\ldots,c_{l_t})$ then l, где l - метка оператора $s_{j'}$, реализуется командами

$$\{(\sigma)(\langle\bar{v},j\rangle,i)\to(\langle\bar{v},j'\rangle,i)(i,i,\varepsilon)\,|\,\sigma\in\Sigma,\ 1\le i\le n,$$

$$\bar{v}=(v_1,\ldots,v_l)\in\bar{V}_1\times\ldots\times\bar{V}_l,\ p(v_{l_1},\ldots,v_{l_t})=1\}\cup$$

$$\cup\{(\sigma)(\langle\bar{v},j\rangle,i)\to(\langle\bar{v},j+1\rangle,i)(i,i,\varepsilon)\,|\,\sigma\in\Sigma,1\le i\le n,$$

$$\bar{v}=(v_1,\ldots,v_l)\in\bar{V}_1\times\ldots\times\bar{V}_l,\ p(v_{l_1},\ldots,v_{l_t})=0\}.$$

5. Оператор s_j вида if $\sigma(x_i)$ then l, где l - метка оператора $s_{j'}$, реализуется командами

$$\{(\sigma)(\langle\bar{v},j\rangle,i)\to(\langle\bar{v},j'\rangle,i)(i,i,\varepsilon),$$

$$(\sigma')(\langle\bar{v},j\rangle,i)\to(\langle\bar{v},j+1\rangle,i)(i,j,\varepsilon)\,|\,\sigma'\in\Sigma,$$

$$\sigma'\ne\sigma,\ 1\le i\le n,$$

$$v\in V_1\times\ldots\times V_l\}\cup$$

$$\cup\{(\sigma')(\langle\bar{v},j\rangle,i')\to(\langle\bar{v},j\rangle,i)(i,i,\varepsilon)\,|\,\sigma'\in\Sigma,$$

$$1\le i'\le n,\ i'\ne i,\ \bar{v}\in\bar{V}_1\times\ldots\times\bar{V}_l\}.$$

(Вторая группа команд осуществляет переход к проверке условия именно для рабочей ленты x_i, а первая - собственно проверку.)

Транслирующий алгоритм T^m для $FTR(m)$-схем мы зададим индуктивным способом. Сначала будет описан T^0, а затем, в предположении о существовании T^l будет конструктивно определен T^{l+1}.

Базис индукции. Произвольная $FTR(0)$-схема характеризуется тем, что на любом протоколе высота ее магазина не убывает, т.е. выполняются только магазинные операции записи. Для произвольной R-схемы A результат $T^0(A)$ есть UFP-схема, моделирующая поведение A либо до первой операции удаления, либо до останова A (если он произойдет раньше). Таким образом,для $FTR(0)$-схемы A схема $T^0(A)$ - эквивалентная UFP-схема:

$T^0(A) =$

$[Var^0;$

$Init^0;$

$loop: Body^0;$

$\underline{if}$ flag = "go" $\underline{then}$ loop;

$\underline{stop}],$

где

$Var^0 = [x:\Delta^*;\ q: K;\ top: \Gamma;\ flag: \{"go","finish","overflow"\};op\_type:\{"+","0","-"\};turn:\{0,-1\}];$

$Init^0 = [q := q_0;\ top := Z_0;\ flag := "go";\ turn := 0;];$

$Body^0 = [b_{h_1};\ldots;b_{h_n}]$, где b_{h_i}, $\{h_1,\ldots,h_n\} = H$, - блока моделирующий выполнение команды h_i. Возможны два случая:

1. h - магазинная операция вида $(\sigma)(\xi,b)\to$ $\to(a,\eta,b'b)$ или операция записи вида $(\sigma)(\xi,b)\to$ $\to(a,\eta,b')$. Тогда b_h имеет вид

```
h.start:  if q ≠ ξ ∨ top ≠ b then h.end;
          if σ(x) then h.do else h.end;
h.do:     x:= xa;
          q:= η ;
          top:= b';
          op_type:= ? ;
          {flag:= "finish";}
h.end:    skip.
```

Знак вопроса следует заменить на "+" для магазинной операции и на "0" - для операции записи. Оператор в фигурных скобках должен присутствовать только в случае, если $\eta\in F$.

2. h - операция стирания вида $(\sigma)(\xi,b)\to(a,\eta,\varepsilon)$. Тогда b_h имеет вид

```
h.start: if q ≠ ξ ∨ top ≠ b then h.end;
         if  σ(x) then h.do else h.end;
h.do:    flag:= "overflow";
         op_type:= "-";
         turn:= -1;
h.end:   skip
```

Построенная конструкция $T^0(A)$ содержит на первый взгляд избыточные элементы, которые,однако, будут использованы при описании шага индукции. Схема $T^0(A)$ моделирует схему A в следующем смысле: при проходе управления через метку loop переменные x, q и top содержат соответственно состояние

рабочей ленты, управляющее состояние и верхний символ магазина схемы **A** перед очередным шагом вычислений.

Шаг индукции. Пусть теперь мы имеем алгоритм трансляции T^l. Этот алгоритм для R-схемы **A** дает UFP-схему $T^l(A)$, которая моделирует поведение схемы **A** до начала $l+1$-го поворота или до остановки **A** (если она произойдет раньше). Таким образом, T^l - транслирующий алгоритм для схем класса $FTR(l)$. Сделаем, кроме того, дополнительные предположения о структуре результирующей UFP-схемы $B^l = T^l(A)$. В терминах введенной выше нотации схема B^l имеет следующее устройство:

```
T^l(A) = [
        Var^l;
        Init^l;
loop:   Body^l;
        if flag = "go" then loop;
        stop
],
```

где Var^l - фрагмент, содержащий все переменные. Среди переменных в Var^l обязательно содержатся: рабочая переменная $x: \Delta^*$, хранящая состояние рабочей ленты модулируемой схемы; управляющая переменная $q: K$, хранящая состояние управления; $top: \Gamma$, хранящая верхний символ магазина; $flag: \{"go", "finish", "overflow"\}$, хранящая признак состояния процесса моделирования; $op\_type: \{"+", "0", "-"\}$, хранящая тип последней промоделированной операции ("+"- магазинная, "0" - запись, "-" - стирание); $turn: \{\pm i \mid i = 0, \ldots, l+1\}$. Состояние переменной $turn$ изменяется следующим образом: в $Init^l$ выполняется оператор $turn := 0$, в каждом фрагменте, моделирующем магазинную операцию, имеется

оператор $turn := (turn)$, а во фрагментах, моделирующих операцию стирания - $turn := \pm(turn)$. Функции "$\pm$" и "$\mp$" определяются следующим образом:

$$\pm(i) = \begin{cases} i, & i < 0, \\ -(i+1), & i \geq 0; \end{cases}$$

$$\mp(i) = |i|.$$

Содержательно $turn = i$, $i > 0$ соответствует подъему i-го поворота ($i < 0$ - спуску) (см. рисунок). Переменная $turn$ будет использоваться для отслеживания начала нужного поворота.

Дадим неформальное описание процесса моделирования поведения $FTR(l+1)$-схемы A. Для моделирования поведения на первых l поворотах можно воспользоваться экземпляром схемы $T^l(A)$. Моделирование поведения в пределах $(l+1)$-го поворота распадается на 2 последовательных этапа: моделирование спуска и моделирование подъема. Моделирование подъема не представляет труда и технически полностью аналогично тому, как это сделано в схеме $T^0(A)$. Для моделирования спуска необходимо уметь восстанавливать последовательность магазинных символов в порядке, обратном тому, в котором они помещались в магазин. Для этого мы построим вспомогательную подсхему BH^l, которая позволит восстанавливать в обратном порядке последовательность символов, записанных в магазин на подъеме одного поворота. Схема BH^l будет содержать управляющую переменную $sentinel: \{1, \ldots, l+1\}$, значение которой будет задаваться внешним по отношению к схеме BH^l образом. Значение переменной $sentinel$ параметризует работу схемы BH^l и представляет собой номер интересующего нас поворота. Опишем принцип функционирования BH^l. Пусть мы имеем два экземпляра схемы $T^l(A)$: By и Bz. Эти экземпляры будут поочередно

выполнять шаги моделирования одной операции схемы A, причем состояние моделирования Bz будет отставать на некоторое число шагов от By. Вначале отставание равно одному шагу. За один цикл работы By будет проходить путь от начального состояния до начала поворота с номером, хранящимся в sentinel. На каждом последующем цикле отставание экземпляра Bz от By будет увеличиваться на один шаг моделирования. Тогда последовательность состояний переменых x, q и top экземпляра Bz в конце каждого цикла моделирования представляет собой взятую в обратном порядке последовательность частичных конфигураций исходной схемы на подъеме поворота с номером sentinel. Поскольку нас интересуют только те моменты поведения схемы A, которые изменяют высоту магазина, то вычеркнем из указанной последовательности все те состояния, в которых следующей операцией экземпляра Bz будет операция записи. Если из оставшейся последовательности выписать отдельно последовательность состояний переменной top, то это и будет последовательность символов, записанных в магазин на подъеме поворота с номером sentinel (за исключением последнего записанного символа),взятая в обратном порядке.

Опишем точно устройство схемы BH^1. Пусть $t: \Delta^*$ - переменная, хранящая начальное состояние рабочей ленты моделируемой схемы (t инициализируется внешним образом):

$$BH^1 = [BH^1\_Var;\ BH^1\_Init;\ BH^1\_Body],$$

где

$$BH^1\_Var =$$

$$[By.Var^1;\ Bz.Var^1;\ prev\_top: \Gamma \cup \{\varepsilon\};\ sentinel:$$

$$\{1...l+1\},\ prev\_op\_type: \{"+","0","-"\};\ t: \Delta^*],$$

```
BH¹_Init =
[By.x:= t; Bz.x:= t; By.Init¹; Bz.Init¹;
prev.top:= ε; prev_op_type:= "0"],
BH¹_Body = [
run:        By.Body¹;
            if |By.turn| = sentinel then check_Bz;
            prev_top:= Bz.top;
            prev_op_type:= Bz.op_type;
            Bz.Body¹;
            if |Bz.turn| = sentinel then init_Bz;
            goto run;
init_Bz:    Bz.x:= t;
            Bz.Init¹;
            goto run;
check_Bz:   By.x:= t;
            By.Init¹;
            if prev_op_type = "+" then finish;
            if prev_op_type = "-" then empty;
            goto run;
empty:      prev>top:= ε ;
finish:     skip]
```

Схема BH^1 устроена так, что после первого выполнения фрагмента BH^1_Body переменная prev_top хранит магазинный символ, записанный предпоследним на подъеме поворота с номером sentinel-1. После следующего выполнения BH^1_Body в prev_top окажется предыдущий магазинный символ. Этот процесс можно продолжить до тех пор, пока переменная prev_top не станет равной ε, что означает исчерпание магазинных символов, записанных на подъеме поворота с номером sentinel-1.

Для моделирования FTR(1)-схемы нам необходимо промоделировать магазин, содержащий не более 1 участков (см. рисунок), каждый из которых состоит из символов, записанных на подъеме одного поворота. Для этого мы заведем массив из 1 экземпляров схемы BH^1 и переменную active, играющую роль указателя вершины магазина. Массив и переменная будут использоваться нами для моделирования магазина, который будет содержать экземпляры схемы BH^1. По обнаружении начала очередного поворота i мы будем увеличивать active на 1 и настраивать переменную sentinel экземпляра BH^1 с номером active на i-й поворот (sentinel:= i). Таким образом, мы как бы помещаем в магазин новый участок, соответствующий подъему поворота i-1. При моделировании любой операции стирания будет выполняться тело экземпляра BH^1 с номером active, что соответствует удалению верхнего символа верхнего участка магазина. В случае, если при очередном обращении к BH^1 с номером active переменная prev_top принимает значение ε (опустошился верхний участок), active будет уменьшаться на 1, что соответствует переходу к участку, лежащему ниже. Как было сказано выше, процесс моделирования поведения FTR(1+1)-схемы A состоит из двух этапов. На первом этапе, делящемся от начального состояния до начала 1+1-го поворота, используется экземпляр схемы $T^1(A)$. Описанные пра-

вила изменения массива экземпляров BH^1 и active применяются, но результат (а именно, значение переменной prev_top экземпляра с номером active) не используется. Главная цель этих действий на первом этапе - поддерживать соответствие между состоянием массива экземпляров BH^1 и состоянием магазина моделируемой схемы. Таким образом, к началу $1+1$-го поворота состояние массива экземпляров схемы BH^1 соответствует состоянию магазина схемы A. Для моделирования операций стирания на спуске $1+1$-го поворота мы будем обращаться к массиву экземпляров BH^1 аналогичным образом, однако теперь содержимое переменной prev_top экземпляра с номером active будет переноситься в переменную top , что соответствует выталкиванию верхнего символа из магазина и замену его на лежащий под ним. Формальное задание схемы $T^{1+1}(A)$ имеет следующий вид:

$T^{1+1}(A)$ = [

Var^{1+1};

Init^{1+1};

loop: Body^{1+1};

if flag = "go" then loop;

stop],

где

Var^{1+1} = [

x: Δ^*; q: K; top: Γ; flag: {"go","finish", "overflow"}; op_type: {"+","0","-"}; turn: {±i | i = 0,..., 1+2} ;

BH[1:1].BH^1_Var; B.Var^1;

active: {1...1}; i: {1...1}; step: {1,2};].

```
Init^{l+1}= [
  top:= Z_0; q:= q_0;
  flag:= "go"; turn:= 0; step:= 1;
  B.x:= x; B.Init^l;
  for i∈{1...l} do
      BH[i].t:= x;
      BH[i].BH^l_Init;
  end
  active: = 1;
  BH[1].sentinel:= 1;],
Body^{l+1} = [
  if step = 2 then step2
step1:
  B.Body^l;
  if B.flag = "finish" then final
  if B.flag ≠ "overflow" then $1
  step: = 2;
  goto step 2;
$1:  q:= B.q;
     flag:= B.flag;
     top:= B.top;
     x:= B.x;
     op_type:= B.op_type;
     if op_type = "-" then react_del_1;
```

```
        turn:= B.turn;
        goto endbody;
    react_del_1:
        if turn = B.turn then $2;
        active:= active + 1;
        BH[active].sentinel:= |B.turn|;
        BH[active].BH^1_Init;
    $2 : turn:= B.turn;
    $3: BH[active].BH^1_Body;
        if BH[active].prev_top ≠ ε then endbody;
        active:= active-1;
        goto $3;
    step2 :
        Do^{l+1};
        if op_type = "-" then react_del_2;
        goto endbody;
    react_del_2:
        BH[active].BH^1_Body;
        top:= BH[active].prev_top;
        if top ≠ ε then endbody;
        active:= active-1;
        goto react_del_2;
    endbody:
        skip],
```

$Do^{l+1} = [Do^{l+1}_{h_1}; \ldots; Do^{l+1}_{h_n}]$,

$Do^{l+1}_{h_i}$, $\{ h_1, \ldots, h_n \} = H$ - блок, моделирующий выполнение команды h_i. Возможны два случая:

1. h - магазинная операция вида $(\sigma)(\xi,b) \to (a,\eta,b'b)$ или операция записи вида $(\sigma)(\xi,b) \to (a,\eta,b')$. Тогда b_h имеет вид:

```
h.start:  if q ≠ ξ ∨ top ≠ b then h.end;
          if σ(x) then h.do else h.end;
h.do:     x:= xa;
          q:= η;
          top:= b';
          op_type:= ?;
          turn:= ∓(turn);
          {flag:= "finish";}
h.end:    skip
```

Знак вопроса следует заменить на "+" для магазинной операции и на "0" - для операции записи. Оператор в фигурных скобках должен присутствовать только в случае, если $\eta \in \mathbf{F}$.

2. h - операция стирания вида $(\sigma)(\xi,b) \to (a,\eta,\varepsilon)$. Тогда b_h имеет вид:

```
h.start:  if q ≠ ξ ∨ top ≠ b then h.end;
          if  σ(x) then h.do else h.end;
```

```
h.do:       turn:= ±(turn);
            if turn ≠ -(1+2) then h.continue
            flag:= "overflow";
            goto endbody;
h.continue:
            op_type:= "-";
            x:= xa;
            q:= η;
            top:= ε ;
            {flag:= "finish";}
h.end:      skip
```

Построение T^m закончено.

Транслирующий алгоритм для $FTR(m_1,\ldots,m_k)$-схем будет, как и для $FTR(m)$-схем, определен индуктивно. Однако теперь параметром индукции будет служить $m=\max\{m_1,\ldots.m_k\}$. $T^0_k(A)$ есть UFP-схема, моделирующая A до первой стирающей операции в любом из магазинов (или до остановки A). $T^0_k(A)$ - UFP-схема, моделирующая A до начала $m+1$-го поворота в любом из магазинов. Нам потребуется небольшая модификация конструкции, приведенной выше. В частности, все переменные, связанные с состоянием магазина (top, turn, op type), теперь превратятся в массивы из k элементов, а операторы, в которых они присутствуют, - в циклы с числом повторений k. Введем дополнительные обозначения: для любой команды $h\in H$ обозначим $Dec(h)$ - множество номеров магазинов, в которых команда h выполняет стирание, $Inc(h)$ - множество номеров

магазинов, высота которых возрастает при выполнении команды h, type(h,j) равно "+", если $j \in$ Inc(h), "-", если $j \in$ Dec(h), и "0" - в противном случае. Опишем сначала устройство $T_k^0(A)$:

$T_k^0(A)$ = [

    Var_k^0;

    $Init_k^0$;

loop: $Body_k^0$;

    if flag = "go" then loop;

    stop

],

где

Var_k^0= [x:Δ^*; q: K; top[1:k]: Γ; flag: {"go", "finish","overflow"); op_type[1:k]:{"+", "0","-"}; turn[1:k]: {0,-1}],

$Init_k^0$ = [

    q:= q_0; flag:= "go";

    for i$\in$ {1..k} do

        turn[i]:= 0;

        top[i]:= Z_0;

    end

],

$Body^0_k = [b_{h_1}; \ldots; b_{h_n}]$, где b_{h_i}, $\{h_1, \ldots, h_n\} = H$ имеет одну из следующих форм:

1) команда h вида $(\sigma)(\xi, б) \to (a, \eta, \bar{w})$ такова, что $Dec(h) = \emptyset$. Тогда b_h имеет вид:

```
h.start:  if q ≠ ξ ∨ top ≠ б then h.end;
          if σ(x) then h.do else h.end;
h.do:     x:= xa;
          q:= η ;
          top:= w̄[1];
          for i∈ {1..k} do
                op_type[i]:= type(h,i);
          end
          {flag:= "finish";}
h end:    skip
```

Оператор в фигурных скобках должен присутствовать только в случае, если $\eta \in F$;

2) h - операция вида $(\sigma)(\xi, б) \to (a, \eta, \bar{w})$ и $Dec(h) \neq \emptyset$. Тогда b_h имеет вид:

```
h.start:  if q ≠ ξ ∨ top ≠ б then h.end;
          if σ(x) then h.do else h.end;
h.do:     flag:= "overflow";
          for i∈ {1..k} do
                op_type[i]:= type(h i);
```

```
                    if op_type[i] ≠ "-" then  $1
                    turn[i]:= -1;
$1                  skip
            end
h.end:      skip
```

Вспомогательные схемы BH_k^1 для описания шага индукции будут теперь, помимо переменной **sentinel**, параметризоваться переменной **magno**, имеющей смысл номера магазина, по которому происходит отслеживание начала поворота с номером **sentinel**. Схема BH_k^1 получается из схемы BH^1 заменой всех вхождений переменных **top**, **turn** и **op_type** соответственно на **top[magno]**, **turn[magno]** и **op_type[magno]**:

$$BH_k^1 = [BH_k^1\_Var;\ BH_k^1\_Init;\ BH_k^1\_Body],$$

где

$BH_k^1\_Var =$

$[By.Var_k^1;\ Bz.Var_k^1;$ **prev_top**: $\Gamma \cup \{\varepsilon\}$;
sentinel: {1...l+1}; **magno**: {1... k};
prev_op_type: {"+","0","-"}; **t**: Δ^*],

$BH_k^1\_Init =$

$[By.x:= t;\ Bz.x:= t;\ By.Init_k^1;\ Bz.Init_k^1;$
prev_top:= ε; **prev_op_type**:= "0"],

$BH_k^1\_Body = [$

```
run:        By.Body_k^1;
            if |By.turn[magno] = sentinel then
            check_Bz;
```

```
            prev_top:= Bz.top[magno];
            prev_op_type:= Bz.op_type[magno];
            Bz.Body_k^l;
            if |Bz.turn[magno]| = sentinel then
            init_Bz;
            goto run;
init_Bz:    Bz.x:= t;
            Bz.Init_k^l;
            goto run;
check_Bz:   By.x:= t;
            By.Init_k^l;
            if prev_op_type = "+" then finish;
            if prev_op_type = "-" then empty;
            goto run;
empty:      prev_top:= ε;
finish:     skip].
```

Опишем теперь конструкцию $T_k^{l+1}(A)$ в предположении о наличии $T_k^l(A)$. Отличия ее от схемы $T^{l+1}(A)$ заключаются в том, что все операторы, связанные с состоянием магазина, заменяются на циклы по числу магазинов (при реализации используемой нами нотации средствами **UFP**-схем, как уже отмечалось выше, эти циклы будут текстуально развернуты в последовательность операторов). Вместо массива **BH[1...l]** вводится двумерный массив **BH[1..k][1...l]**, который можно мыслить как набор из **k** одномерных массивов **BH[i]**, **i** = $\overline{1,k}$, каждый из которых используется для моделирования **l** поворотов в магазине **i** полностью аналогично конструкции для **FTR(m)**-схем. Переменная **active** соответственно заменяется на массив **active[1...k]**. Формально

```
T_k^{l+1}(A) = [
        Var_k^{l+1};
        Init_k^{l+1};
loop:   Body_k^{l+1};
        if flag = "go" then loop;
        stop],
```

где

```
Var_k^{l+1} = [
    x: Δ*; q: K; top[1:k]: Γ;
    flag: {"go","finish","overflow"};
    op_type[1:k]: {"→","0","-"}; turn[1:k]: {±i |
    i = 0,..., l+2};
    BH[1:k][1:l].BH_k^l_Var; B.Var_k^l;
    active[1:k]: {1...l}; i: {1...k}; j: {1...l};
    step: {1,2};],
Init_k^{l+1} = [
    q:= q_0;
    flag:= "go"; step:= 1;
    B.x:= x; B.Init^l;
    for i∈ {1...k} do
        top[i]:= Z_0;
        turn[i]:= 0;
        active[i]:= 1;
        for j∈ {1...l} do
            BH[i][j].t:= x;
            BH[i][j].magno:= i;
```

```
            BH[i][j] BH^1_k_Init;
        end
        BH[i][j].sentinel:= 1;
    end],
Body^{i+1}_k = [
    if step = 2 then step2
step1:
    B.Body^i_k;
    if B.flag = "finish" then final
    if B.flag ≠ "overflow" then $1
    step:==2;
    goto step2;
$1: q:= B.q;
    flag:= B.flag;
    top:= B.top;
    x:= B.x;
    op_type:= B.op_type;
    for i∈ {1...k} do
        if op_type[i] = "-" then react_del_1;
        turn[i]:= B.turn[i];
        goto next1;
react_del_1:
        if turn[i] = B.turn[i] then $2;
        active[i]:= active[i]+1;
        BH[i][active[i]].sentinal:= |B.turn[i]|;
        BH[i][active[i]] BH^1_k_Init ;
$2:     turn[i]:= B.turn[i];
$3:     BH[i][active[i]].BH^1_k_Body;
        if BH[i][active[i]].prev_top ≠ ε then next1;
```

```
        active[i]:= active[i]-1;
        goto  $3;
next 1 :skip;
   end
   goto endbody;
step 2 :
   Do_k^{l+1};
   for i∈ {1...k}  do
        if op_type[i]= "-" then react_del_2;
        goto next 2 ;
react_del_2:
        BH[i][active[i]].BH_k^l_Body;
        top[i]:= BH[i][active[i]].prev_top;
        if top[i] ≠ ε then next 2 ;
        active[i]:= active[i]-1;
        goto react_del_2;
next2: skip
   end
endbody:
   skip],
```

$Do_k^{l+1} = Do_{h_1}^{l+1};\ldots;\ Do_{h_n}^{l+1}]$,

$Do_{h_i}^{l+1}, \{ h_1, \ldots, h_n \} = H$ – блок, моделирующий выполнение команды h_i. Возможны два случая:

1. Команда h вида $(\sigma)(\xi,б) \to (a,\eta,\overline{w})$ такова, что $Dec(h) = \emptyset$. В этом случае массив top при выполнении оператора top:= $\overline{w}[1]$; получает значения новых верхних символов магазинов, b_h имеет вид

```
h.start:  if q ≠ ξ ∨ top ≠ б then h.end;
          if σ(x) then h.do else h.end;
h.do:     x:= xa;
          q:= η ;
          top:= w̄[1];
          for i∈ {1...k} do
              turn[i]:= (turn[i]);
              op_type[i]:= type(h,i);
          end
          {flag:= "finish";}
h.end:    skip
```

Оператор в фигурных скобках должен присутствовать только в случае, если $\eta \in F$.

2. Команда h вида $(\sigma)(\xi,б) \to (a,\eta,\bar{w})$ такова, что $Dec(h) \neq \emptyset$. В этом случае после выполнения оператора $top:= \bar{w}[1];$ для всех $i \in Dec(h)$ (или $op\_type[i] =$ $= "-")$ значение $top[i] = \varepsilon$. Восстановление недостающих верхних символов соответствующих магазинов на шаге 2 (step = = 2, что означает, что по крайней мере один из магазинов находится на спуске $1+1$-го поворота) осуществляется фрагментом, начинающимся меткой react_del_2 , при помощи обращения к соответствующей активной вспомогательной схеме $BH[i][active[i]]$. Ситуация переполнения вызывается достижением любым из магазинов начала $1+2$-го поворота:

```
h.start:  if q ≠ ξ  top ≠ ƃ then h.end;
          if  σ(x) then h.do else h.end;
h.do:     top:= w̄[1];
          x: = xa;
          q:= η ;
          for  i∈ {1...k} do
               op_type[i]:= type(h,i);
               if op_type[i] ≠ "-" then  $1;
               turn[i]:= ±(turn[i]) ;
               if turn[i]≠ -(1+2) then continue;
               flag:= "overflow";
               goto continue;
$1:            if op_type[i]≠"+" then continue;
               turn[i]:= ∓(turn[i]);
continue:      skip
          end
          {flag:= "finish";}
h.end:    skip
```

Построение $T_k^{i+1}(A)$ закончено.

Возможным обобщением этих результатов является рассмотрение R^k-схем со стеками, допускающими конечное число поворотов высоты стека и произвольное число поворотов высоты читающей головки. Техника, предложенная в данной работе, не может

быть непосредственно распространена на этот случай. Представляет интерес поиск модификации моделирующей конструкции для схем со стеками такого вида.

Литература

1. КОТОВ В.Е. Введение в теорию схем программ. - Новоси - бирск: Наука СО, 1978. - 257 с.

2. ЛИСОВИК Л.П. Металинейные схемы с засылками констант //Программирование. - 1985. - № 2. - С. 29-38.

3. ЛИСОВИК Л.П. Алгебры полулинейных преобразователей над размеченными деревьями //Языки спецификаций и логическое программирование. - Новосибирск, 1988. - Вып. 124: Вычислитель - ные системы. - С. 114-143.

4. PLAISTED D. Program schemas with counters //Proc. 4th Ann. ACM Symp. on Theory of Comp., Denver. - 1972. - P.44-51.

5. ГАРЛЭНД С., ЛАКХЭМ Д. Стандартные схемы, рекурсивные схемы и формальные языки //Киб.сб. Новая серия. - М., 1976. - Вып.13.- С.73-119.

6. ГИНЗБУРГ С. Математическая теория контекстно-свободных языков. - М.: Мир, 1970. - 326 с.

7. СТИРНЗ Р. Проверка регулярности для магазинных автоматов// Киб. сб. Новая серия. - М., 1971. - Вып. 8. -С.117-139.

8. POST E.L. A variant of a recursive unsolvable problem //Bull. Am. Math. Soc. - 1946. - Vol. 52, N. 4. -P. 264-268.

Поступила в ред.-изд.отд.

12 апреля 1991 года