1973 год ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ Выпуск 57

УДК 681.142.2

МЕТАЯЗЫК ТРАНСЛЯЦИИ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

В.И. Константинов, Р.М.Нуриев

Эффективность решения задач обработки информации, обладающей синтаксической структурой, в сильной степени определяется мета-языком, используемым для порождения и анализа входных цепочек [1, 2]. В работе в качестве такого метаязыка предлагается TR-метаязык, получаемый распространением правил метаязыка R-грамматик [3,4] на нетерминальный алфавит. TR-грамматики эффективны для порождения, анализа и трансляции контекстно-свободных языков. Преобразующие TR-грамматики близки к TI-машинам, рассмотренным в [5], однако они решают задачу трансляции на основе нового, удобного для машинной реализации метаязыка.

ОПРЕДЕЛЕНИЕ I. TR-грамматикой назовем шестерку

$$G = (A, V_N, P, R, \{\tau_c^{G(x_i)}\}, R^*), \qquad (I)$$

где I) $A = \{\alpha_0, \dots, \alpha_k\}$ - терминальный алфавит;

2) $V_N = \{x_0, \dots, x_n\}$ - нетерминальный алфавит имен атомных грамматик, $A \cap V_N = \emptyset$;

3) $P = \{0, 1\}$;

4) $R = R^{G(x_0)} \cup \dots \cup R^{G(x_n)}$; $R^{G(x_i)} \cap R^{G(x_j)} = \emptyset$, $i \neq j$; $R^{G(x_i)} = \{\tau_\phi, \tau_j^{G(x_i)}\}$ - множество имен подмножеств элементов из $(A \cup V_N) \times R^{G(x_i)} \times P$, $0 \leqslant j \leqslant \ell_i$, τ_ϕ - имя пустого подмножества;

5) $\{\tau_0^{G(x_i)}\}$ - начальное подмножество (аксиома) для $x_i \in V_N$, $0 \leqslant i \leqslant n$, $\tau_0^{G(x_i)} \in R^{G(x_i)}$. Одно из начальных подмножеств, $\tau_0^{G(x_0)}$, назовем аксиомой грамматики G;

6) R^* - подмножество R, возможно, пустое.

Так определенная TR-грамматика G является совокупностью TR - грамматик $G(x_0), \dots, G(x_n)$, каждая из которых есть $G(x_i) = (A, V_N, P, R^{G(x_i)}, \tau_0^{G(x_i)}, R^*_{G(x_i)})$, где $A, V_N, P, R^{G(x_i)}, \tau_0^{G(x_i)}$ - те же, что и ранее, а $R^*_{G(x_i)} = R^* \cap R^{G(x_i)}$. Грамматики $G(x_0), \dots, G(x_n)$ назовем а т о м н ы м и .

Элемент $(\sigma, \tau, p) \in ((A \cup V_N) \times R \times P)$ назовем правилом TR-грамматики G и при $p = 0$ запишем в виде:

$$\sigma \rightarrow \tau, \qquad (2)$$

а при $p = 1$ - в виде:

$$\sigma \rightarrow . \tau. \qquad (3)$$

Будем интерпретировать правила вида (2) следующим образом:

а) при $\sigma \in A$ элемент σ может конкатенировать с множеством терминальных левых частей правил из $\bar{\tau}$, либо со множеством слов, порожденных атомными грамматиками для нетерминальных левых частей правил из $\bar{\tau}$;

б) при $\sigma \in V_N$ слово, порожденное $G(\sigma)$, конкатенирует описанным выше образом.

Здесь $\bar{\tau} = \tau$, если $(\sigma \rightarrow \tau)$ входит в подмножество $\tau_p \in R^*$, и $\bar{\tau} = \tau \cup \tau_s$, если $(\sigma \rightarrow \tau) \in \tau_p \in R^*$, τ_s - текущее содержимое стека связи. При $\tau = \tau_\phi$ элемент σ (либо слово, порожденное $G(\sigma)$) последний в данной грамматике.

Правила вида (3) интерпретируем, как и правила вида (2), с той лишь разницей, что элемент σ (либо слово, порожденное $G(\sigma)$) может быть последним в данной грамматике и при $\tau \neq \tau_\phi$.

Правила вида (2) с $\tau = \tau_\phi$ либо вида (3) назовем з а к л ю ч и т е л ь н ы м и . Множество $\bar{\tau}$ назовем множеством г л о б а л ь н ы х правил-преемников данного правила. При $\sigma \in V_N$ правило $\sigma \rightarrow \tau$ имеет еще и множество л о к а л ь н ы х правил-преемников. Такое деление обусловлено тем, что при встрече правила с нетерминальной левой частью первоначально идем на порождение подслова в грамматику $G(\sigma)$, откуда с заключительного правила переходим на множество локальных правил-преемников. Процесс этот реализуется с помощью стека связи атомных грамматик. Именно, в момент ухода на порождение подслова в $G(\sigma)$ в стек записывается следующая информация о правиле $\sigma \rightarrow \tau$: имя подмножества

правил, его содержащего; номер этого правила. После выхода на заключительное правило грамматики $G(\sigma)$ эта информация считывается из стека, и процесс порождения продолжается.

В ы в о д о м слова $S=\alpha_1 \dots \alpha_m, S\in A^*$, назовем цепочку пар $(G(x),\sigma\to\tau),(\sigma\to\tau)\in G(x)$, каждое последующее правило которой принадлежит либо множеству глобальных правил-преемников предыдущего при $\sigma\in A$, либо множеству локальных, а затем глобальных правил-преемников при $\sigma\in V_N$ (при этом терминальные символы правил последовательности образуют слово S).

Вывод слова S назовем п о л н ы м, если первое из правил цепочки принадлежит $\tau_0^{G(x_0)}$, последнее правило цепочки - заключительное из $G(x_0)$ и стек связи пуст. Я з ы к $L(G)$, п о р о ж д е н н ы й TR-грамматикой G, определим как множество терминальных цепочек, для которых существует полный вывод.

Аналогично выводу осуществляется и распознавание цепочек из $L(G)$. TR-грамматики вида (I) ориентированы на решение задачи синтаксического анализа. Правилам TR- грамматик можно поставить в соответствие некоторые семантические эквиваленты, определяющие преобразование (трансляцию) входных цепочек.

ОПРЕДЕЛЕНИЕ 2. П р е о б р а з у ю щ е й TR-грамматикой назовем семерку вида

$$G=(A, V_N, P, R, \{\tau_0^{G(x_i)}\}, R^*, \Delta), \qquad (4)$$

где $A, V_N, P, R, \{\tau_0^{G(x_i)}\}, 0\le i\le n, R^*$- те же, что и ранее, а Δ-конечный выходной алфавит. Элементы Δ можно рассматривать, например, как номера подпрограмм, осуществляющих трансляцию элементов входной строки. Правила TR- грамматик вида (4) интерпретируем, как и правила грамматик вида (I), с той лишь разницей, что теперь формируем еще и выходную строку. При этом если применялось правило $\sigma_\delta\to\tau, \sigma\in A$, $\delta\in\Delta$, то δ посылаем в выходную строку сразу же. Если $\sigma\in V_N$, то правило записываем в стек связи вместе с элементом δ, который посылаем в выходную строку после считывания этого правила из стека.

Описанный формализм позволяет в компактном и удобном для трансляции виде задавать грамматики входных языков, давая возможность посредством нетерминальных символов транслировать синтаксические конструкции с законченным семантическим описанием.

ПРИМЕР. Дана грамматика, бэкусово-науровская форма (БНФ) которой имеет вид:

⟨ выражение ⟩ ::= ⟨ формула ⟩;
⟨ формула ⟩ ::= ⟨терм ⟩ | ⟨ формула⟩ + ⟨терм ⟩
⟨ терм⟩ ::= ⟨множитель ⟩ | ⟨ терм⟩ * ⟨ множитель ⟩
⟨ множитель⟩ ::= a|b| ... |y|z| (⟨ формула ⟩)

Язык, порожденный этой грамматикой, состоит из простого вида арифметических выражений, ограниченных справа точкой с запятой. Обозначим, для краткости, В - ⟨ выражение ⟩, Ф - ⟨ формула⟩, Т - ⟨ терм⟩, М - ⟨ множитель⟩, δ - любая из букв. Тогда соответствующей TR-грамматикой будет

$$G=(A, V_N, P, R, \{\tau_0^{G(B)}, \tau_0^{G(\Phi)}, \tau_0^{G(T)}, \tau_0^{G(M)}\}, R^*), \qquad (5)$$

где 1) $A=\{\delta, ;, +, *, (,)\}$;

2) $V_N=\{B, \Phi, T, M\}$;

3) $P=\{0,1\}$;

4) $R=R^{G(B)}\cup R^{G(\Phi)}\cup R^{G(T)}\cup R^{G(M)}=\{\tau_0^{G(B)}, \tau_1^{G(B)}, \tau_0^{G(\Phi)}, \tau_1^{G(\Phi)}, \tau_0^{G(T)}, \tau_1^{G(T)}, \tau_0^{G(M)}, \tau_1^{G(M)}, \tau_2^{G(M)}\}$;

5) $\tau_0^{G(B)}, \tau_0^{G(\Phi)}, \tau_0^{G(T)}, \tau_0^{G(M)}$ - аксиомы атомных грамматик, $\tau_0^G=\tau_0^{G(B)}$;

6) $R^*=\{\tau_1^{G(\Phi)}, \tau_1^{G(T)}\}$.

Разомкнутые подмножества правил (имена которых принадлежат R^*) выделим в }{ .

$G(B)$: $\tau_0^{G(B)}=\{\Phi\to\tau_1^{G(B)}\}$, $\tau_1^{G(B)}=\{;\to\tau_\phi\}$.

$G(\Phi)$: $\tau_0^{G(\Phi)}=\{T\to\tau_1^{G(\Phi)}\}$, $\tau_1^{G(\Phi)}=\}+\to\tau_0^{G(\Phi)}\{$.

$G(T)$: $\tau_0^{G(T)}=\{M\to\tau_1^{G(T)}\}$, $\tau_1^{G(T)}=\}*\to\tau_0^{G(T)}\{$.

$G(M)$: $\tau_0^{G(M)}=\{\delta\to\tau_\phi, (\to\tau_1^{G(M)}\}$, $\tau_1^{G(M)}=\{\Phi\to\tau_2^{G(M)}\}$, $\tau_2^{G(M)}=\{)\to\tau_\phi\}$.

Процесс распознавания строки $a+b*(c+d)$; в грамматике (5), приведен на рис. I.

	a	+	в	*	(	c	+	d	)	;
Состояние ↓	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$	$z_0^{G(B)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$	$z_0^{G(T)}$ $z_0^{G(M)}$	$z_1^{G(M)}$ $z_1^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$	$z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$ $z_2^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$ $z_1^{G(B)}$
Стек →	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_1^{G(M)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_1^{G(M)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_1^{G(M)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	

Рис. I

Для той же грамматики в НФ преобразующей TR-грамматикой, дающей на выходе бесскобочную запись правильно построенных входных строк, будет

$$G=(A, V_N, P, R, \{z_0^{G(B)}, z_0^{G(\Phi)}, z_0^{G(T)}, z_0^{G(M)}\}, R^*, \Delta), \qquad (6)$$

где 1) A, V_N, P, R^* и множество аксиом – те же, что и ранее;

2) $R=R^{G(B)} \cup R^{G(\Phi)} \cup R^{G(T)} \cup R^{G(M)} = \{z_0^{G(B)}, z_1^{G(B)}, z_0^{G(\Phi)}, z_1^{G(\Phi)}, z_2^{G(\Phi)}, z_0^{G(T)}, z_1^{G(T)}, z_2^{G(T)}, z_0^{G(M)}, z_1^{G(M)}, z_2^{G(M)}\}$;

3) $\Delta=\{\lambda, ;, \delta, +, *\}$, λ – пустой символ.

$G(B)$: $z_0^{G(B)}=\{\Phi \xrightarrow{\lambda} z_1^{G(B)}\}$, $z_1^{G(B)}=\{; \xrightarrow{;} z_\phi\}$.

$G(\Phi)$: $z_0^{G(\Phi)}=\{T \xrightarrow{\lambda} z_1^{G(\Phi)}\}$, $z_1^{G(\Phi)}=\{+ \xrightarrow{\lambda} z_2^{G(\Phi)}\}$, $z_2^{G(\Phi)}=\{T \xrightarrow{+} z_1^{G(\Phi)}\}$.

$G(T)$: $z_0^{G(T)}=\{M \xrightarrow{\lambda} z_1^{G(T)}\}$, $z_1^{G(T)}=\{* \xrightarrow{\lambda} z_2^{G(T)}\}$, $z_2^{G(T)}=\{M \xrightarrow{*} z_1^{G(T)}\}$.

$G(M)$: $z_0^{G(M)}=\{\delta \xrightarrow{\delta} z_\phi, (\xrightarrow{\lambda} z_1^{G(M)}\}$, $z_1^{G(M)}=\{\Phi \xrightarrow{\lambda} z_2^{G(M)}\}$, $z_2^{G(M)}=\{) \xrightarrow{\lambda} z_\phi\}$.

Процесс формирования бесскобочной записи строки $a+b*(c+d)$; в грамматике (6) приведен на рис. 2.

	a	+	в	*	(	c	+	d	)	;
Состояние ↓	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$	$z_2^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$	$z_2^{G(T)}$ $z_0^{G(M)}$	$z_1^{G(M)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$	$z_2^{G(\Phi)}$ $z_0^{G(T)}$ $z_0^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$ $z_2^{G(M)}$	z_ϕ $z_1^{G(T)}$ $z_1^{G(\Phi)}$ $z_1^{G(B)}$
Стек →	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_2^{G(T)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_2^{G(\Phi)}$	$z_0^{G(B)}$ $z_2^{G(\Phi)}$ $z_2^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_2^{G(T)}$ $z_2^{G(T)}$ $z_1^{G(M)}$ $z_0^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_2^{G(T)}$ $z_2^{G(T)}$ $z_1^{G(M)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_2^{G(T)}$ $z_2^{G(T)}$ $z_1^{G(M)}$ $z_2^{G(\Phi)}$ $z_0^{G(T)}$	$z_0^{G(B)}$ $z_0^{G(\Phi)}$ $z_2^{G(T)}$ $z_2^{G(T)}$	
	a		в			c		d	+	*, +, ;

Рис. 2

TR-грамматика (6), может управлять процессом формирования машинных команд, реализующих вычисления по заданным формулам. Для этого введем еще стек операндов. Последнюю заполненную ячейку стека обозначим $ST[n]$, предпоследнюю ячейку стека – $ST[n-1]$ и т.д. Для одноадресной машины выходные символы интерпретируем как номера подпрограмм (ПП), выполняющих следующие действия:

ПП δ :

$$n \leftarrow n+1,$$
$$ST[n] \leftarrow A[x],$$

где n – указатель вершины стека операндов (в момент начала трансляции он равен нулю); $A[x]$ – адрес ячейки, содержащей значение переменной x.

ПП + : 1) если $ST[n]$ = НЕД и $ST[n-1]=A[x]$ либо если $ST[n]=A[x]$ и $ST[n-1]$ = НЕД, то сформировать команду СЛО $A[x]$ после чего выполнить

$$n \leftarrow n-1,$$
$$ST[n] \leftarrow \text{НЕД}$$

2) если $ST[n]=A[x], ST[n-1]=A[y]$, то сформировать команды:

ЗСМ A[x],
СЛО A[y],

после чего выполнить $n \leftarrow n-1$, $ST[n] \leftarrow$ НЦД.

Здесь НЦД – код находящегося на сумматоре результата предыдущего действия, СЛО – код команды сложения с сумматором, ЗСМ – код команды занесения на сумматор.

Ш* : аналогична Ш +, только вместо команды СЛО A[x] формирует команду умножения УМН A[x] .

Ш; : 1) если $ST[n]=A[x]$, то сформировать команды

ЗСМ A[x],
ССМ R

и присвоить n значение 0;

2) если $ST[n]=$ НЦД, то сформировать команду ССМ R и присвоить n значение 0.

Здесь ССМ – код команды считывания из сумматора; R – адрес рабочей ячейки.

Тогда TR-грамматика (6) будет для правильно построенных исходных выражений формировать последовательности машинных команд, вычисляющих их значения и запоминающих результат в ячейке R.

В рассмотренном выше примере (рис.2) перед трансляцией закрывающей скобки стек операндов имеет вид:

A[a]	A[b]	A[c]	A[d]	

В процессе анализа закрывающей скобки из стека связи выталкивается правило из подмножества $z_2^{G(\Phi)}$, семантика Ш+ которого формирует команды:

ЗСМ A[d],
СЛО A[c],

после чего стек операндов принимает вид:

A[a]	A[b]	НЦД	

Затем в процессе анализа точки с запятой из стека связи выталкивается правило из $z_2^{G(T)}$, семантика Ш* которого формирует

команду УМН A[b], после чего выталкивается правило из $z_2^{G(\Phi)}$, семантика Ш+ которого формирует команду СЛО A[a], и, наконец, применяется правило из $z_1^{G(B)}$, семантика Ш, которого формирует команду ССМ R и очищает стек операндов. Таким образом, выражение $a+b*(c+d)$; транслируется в последовательность команд:

ЗСМ A[d],
СЛО A[c],
УМН A[b],
СЛО A[a],
ССМ R.

Приведенный пример в достаточной мере иллюстрирует возможности TR-грамматик. Рассмотрим теперь одно их обобщение.

ОПРЕДЕЛЕНИЕ 3. Обобщенной TR-грамматикой назовем девятку

$$G=(A, V_N, P, R, \{z_0^{G(x_i)}\}, R^*, \Delta, Q, \mathcal{A}),$$

где 1) A – терминальный алфавит, $A=\{a_0, \ldots, a_k\}$;

2) V_N – нетерминальный алфавит имен атомных TR-грамматик,

$$V_N=\{x_0, \ldots, x_n\}, \quad A \cap V_N=\emptyset;$$

3) $P=\{0,1\}$;

4) $R=R^{G(x_0)} \cup \ldots \cup R^{G(x_n)}$, $R^{G(x_i)} \cap R^{G(x_j)}=\emptyset$, $0 \le i, j \le n$, $i \neq j$, $R^{G(x_i)}=\{z_\emptyset, z_j\}$ – множество имен подмножеств элементов из $(A \cup V_N) \times R^{G(x_i)} \times P \times \Delta \times Q$, $0 \le i \le n$, $z_\emptyset$ – имя пустого подмножества;

5) $z_0^{G(x_i)}$ – начальное подмножество (аксиома) для $x_i \in V_N$, $0 \le i \le n$, $z_0^{G(x_0)}$ – аксиома грамматики G;

6) $R^* \subseteq R$;

7) Δ – конечный выходной алфавит, $\Delta=\{\delta_1, \ldots, \delta_p\}$;

8) Q – конечное множество рекурсивных функций $Q_s^{z_1, \ldots, z_k, v_{k+1}, \ldots, v_n}$, $s \in \{0,1,2\}$, $z_i \in R$, $v_k \in \mathcal{A}$;

9) $\mathcal{A}$ – конечный словарь стековых символов, $\mathcal{A}=\{v_1, \ldots, v_q\}$.

Элемент $(\sigma, p, Q_s^{z_1, \ldots, v_n}, \delta, z_i) \in z_j$ назовем правилом обобщенной TR-грамматики, $\sigma \in (A \cup V_N)$, $p \in P$, $Q_s^{z_1, \ldots, v_n} \in Q$, $\delta \in \Delta$, $z_i, z_j \in R$, и будем интерпретировать следующим образом.

а) При $s = 0$ и $\sigma \in A$ элемент σ может конкатенировать с множеством терминальных левых частей правил из $\bar{\tau}$, либо со множеством слов, порожденных атомными TR-грамматиками для нетерминальных левых частей правил из $\bar{\tau}$. При $\sigma \in V_N$ описанным выше образом конкатенирует слово, порожденное $G(\sigma)$. При этом выходной символ пройденного правила (либо цепочка символов при $\sigma \in V_N$) является следующим элементом выходной цепочки. При $p = 0$ и $\tau = \tau_\varnothing$ порожденный элемент (либо слово) - последний в данной грамматике, при $p = 1$ порожденный элемент (либо слово) может быть последним и при $\tau \neq \tau_\varnothing$.

б) При $s = 1$ элемент σ (либо слово, порожденное $G(\sigma)$) конкатенирует, как и при $s = 0$. При этом происходит запись непустых $\tau_1, \ldots, \tau_k, v_{k+1}, \ldots, v_n$ в соответствующие стеки.

в) При $s = 2$ конкатенация элемента σ (либо слова, порожденного $G(\sigma)$) имеет место при совпадении непустых $v_{k+1}, \ldots, v_n$ с элементами, стоящими в вершинах стеков $k+1, \ldots, n$. При этом происходит считывание совпавших элементов.

Здесь $\bar{\tau} = \tau_i, \tau_j \bar{\in} \mathcal{R}^*$ либо $\tau_i \cup \tau_s \cup \tau'_1 \cup \ldots \cup \tau'_k, \tau_j \in \mathcal{R}^*$, τ_s - текущее содержимое вершины стека связи, $\tau'_1, \ldots, \tau'_k$ - содержимое вершин стеков $1, \ldots, k$. Очевидно, что при $V_N = \varnothing$ и $\Delta = \varnothing$ обобщенная TR-грамматика есть разомкнутая $\mathcal{R}$-грамматика [3] с W_m - отношениями типа 0,1,2.

Таким образом, разработанный метаязык TR-грамматик удобен для порождения, анализа и трансляции контекстно-свободных языков. Он может найти широкое применение при построении эффективных систем автоматизации программирования как для машин, так и для однородных вычислительных систем [6].

Литература

1. ВЕЛЬБИЦКИЙ И.В., КОСАРЕВ Ю.Г. Системный подход к построению трансляторов для вычислительных систем. - "Вычислительные системы", Новосибирск, 1970, вып. 42, с. 12-21.

2. ГЛУШКОВ В.М., ВЕЛЬБИЦКИЙ И.В., СТОГНИЙ А.А. Об одном подходе к построению системного математического обеспечения современных вычислительных машин. - "Кибернетика", Киев, 1972, №3, с. 25-35.

3. ВЕЛЬБИЦКИЙ И.В. О метаязыке синтаксически управляемого транслятора. - "Вычислительные системы", Новосибирск, 1970, вып. 42, с. 22-31.

4. ВЕЛЬБИЦКИЙ И.В., ЩЕНКО Е.Л. Метаязык, ориентированный для синтаксического анализа и контроля. - "Кибернетика", Киев, 1969, №2, с. 50-53.

5. **VERE S. Translation Equations. - Communications of the ACM, 1970, vol.13, N 2, Feb., p.83-89.**

6. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности, Новосибирск, "Наука", 1966.

Поступила в ред.-изд.отд.
6 февраля 1973 года