TALLINN

TPI '78

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED

439

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

Труды экономического факультета
XXX1

АНАЛИЗ ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ, ВОПРОСЫ ПРОГРАММИРОВАНИЯ

439

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

TPI '78

УДК 519.24, 518.5, 681.3.068, 519.271/272, 517.5, 51:801

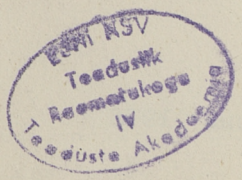
Труды экономического факультета XXX1

АНАЛИЗ ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ, ВОПРОСЫ ПРОГРАММИРОВАНИЯ

Таллин 1978

Содержание

ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
Труды ТПИ № 439
АНАЛИЗ ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ, ВОПРОСЫ ПРОГРАММИРОВАНИЯ
Труды экономического факультета XXXI
Редактор И.Амитан. Техн. редактор В.Ранник
Сборник утвержден коллегией Трудов ТПИ 7 дек. 1977 г.
Подписано к печати 4 апреля 1978 г.
Бумага 60x90/16. Печ. л. 5,75+0,125 приложение
Уч.-изд. л. 4,91. Тираж 300
МВ-02676
Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Заказ № 519
Цена 74 коп.



TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 439 1978

УДК 519.24

Л.К.Выханду, Х.О.Крусберг

О НЕЛИНЕЙНОМ ФАКТОРНОМ АНАЛИЗЕ II

I. Введение

Настоящая статья является продолжением статьи [I]. Все понятия и обозначения такие же, как и в первой части. Чтобы облегчить чтение данной статьи, повторим главное.

Из матрицы факторных нагрузок A линейной факторной модели [2]

$$Z = AF \tag{I}$$

выбираем три линейно независимые столбца:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & a_{m3} \end{pmatrix}.$$

Матрица факторных значений F теперь следующая:

$$F = \begin{pmatrix} f_{11} & f_{12} \cdots f_{1n} \\ f_{21} & f_{22} \cdots f_{2n} \\ f_{31} & f_{32} \cdots f_{3n} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}.$$

Для матрицы F выполнены условия ортогональности и нормированности:

$$E(f_p) = 0, \quad E(f_p f_q) = 0 \ (p \neq q), \quad E(f_p^2) = 1 \tag{2}$$

(здесь E обозначает математическое ожидание).

Если имеется ортогональная 3x3-матрица T и использовать обозначения

$$\begin{aligned} B &= AT, \\ G &= T'F, \end{aligned} \tag{3}$$

то модели (I) можно придать следующий вид:

$$Z = BG.$$

Пусть матрица G имеет вид $[I; 3]$:

$$G = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ x_{11}x_{21} & x_{12}x_{22} & \cdots & x_{1n}x_{2n} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_1x_2 \end{pmatrix}. \qquad (4)$$

Пусть ее строки ортогональны и нормированы.

Наряду с матрицей (4) рассмотрим матрицу

$$\bar{G} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

с ортогональными и нормированными строками:

$$\begin{aligned} &E(x_1)=0, \quad E(x_2)=0, \quad E(x_3)=0, \\ &E(x_1x_2)=0, \quad E(x_1x_3)=0, \quad E(x_2x_3)=0, \\ &E(x_1^2)=1, \quad E(x_2^2)=1, \quad E(x_3^2)=1. \end{aligned} \qquad (5)$$

Если потребовать, что

$$x_{3j} = x_{1j}\, x_{2j},$$

то к условиям (5) прибавляются еще условия

$$\begin{aligned} &E(x_1x_3) = E(x_1^2x_2) = 0, \\ &E(x_2x_3) = E(x_1x_2^2) = 0, \\ &E(x_3^2) = E(x_1^2x_2^2) = 1. \end{aligned} \qquad (6)$$

Исходя из геометрических соображений, можно провести вращение координатных осей поочередно на углы φ_{12}, φ_{13} и φ_{23} в факторном пространстве $(f_1; f_2; f_3)$:

$$T_{12} = \begin{pmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad T_{13} = \begin{pmatrix} c_{13} & 0 & -s_{13} \\ 0 & 1 & 0 \\ s_{13} & 0 & c_{13} \end{pmatrix},$$

$$T_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & -s_{23} \\ 0 & s_{23} & c_{23} \end{pmatrix}$$

(здесь использованы обозначения $c = \cos\varphi$; $s = \sin\varphi$). Тогда в качестве общей матрицы вращения T можно принять произведение

$$T = T_{12}\, T_{13}\, T_{23}$$

и формула (3) принимает вид

$$G = T'_{23}\, T'_{13}\, T'_{12}\, F.$$

Совершаем на плоскости $(f_1; f_2)$ поворот осей:

$$\bar{G}_{12} = T'_{12} F = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_{12} f_1 + s_{12} f_2 \\ -s_{12} f_1 + c_{12} f_2 \\ f_3 \end{pmatrix}. \qquad (7)$$

Условия (5) выполняются для матрицы $\bar{G}_{12}$ точно, так как матрицы T_{12} и F ортогональны (это можно проверить и прямо, учитывая условия (2)). Но условия (6) нельзя вывести таким же образом, они выполняются необязательно точно.

Введем следующие обозначения для упрощения формул:

$$\begin{aligned}
&K_1 = E_1(f_1^3), \quad K_2 = E(f_2^3), \quad K_3 = E(f_1^2 f_2), \\
&K_4 = E(f_1^2 f_3), \quad K_5 = E(f_1^2 f_2^2), \quad K_6 = E(f_1 f_2^2), \\
&K_7 = E(f_2 f_3^2), \quad K_8 = E(f_2^2 f_3^2), \quad K_9 = E(f_1 f_3^2), \\
&L_1 = E(f_1^2 f_3^2), \quad L_2 = E(f_2^2 f_3), \quad L_3 = E[(f_2^2 - f_1^2)^2], \\
&L_4 = E(f_1 f_2 f_3), \quad L_5 = E[f_1 f_2 (f_2^2 - f_1^2)], \quad L_6 = E(f_1 f_2^2 f_3), \\
&\qquad\qquad L_7 = E(f_1^2 f_2 f_3).
\end{aligned} \qquad (8)$$

2. Ослабление условий Макдональда

Макдональд в статье [3] предполагает, что

$$E(x_1^2 x_2^2) = 1$$

для матрицы $\bar{G}$ (7) на всех плоскостях. Постараемся справиться без этого условия. На плоскости $(f_1; f_2)$ составим выражение

$$\Phi_{12} = E[(x_3 - x_1 x_2)^2] = E(x_3^2) + E(x_1^2 x_2^2) - 2E(x_1 x_2 x_3)$$

и найдем его минимальное значение (тогда $x_3 \approx x_1 x_2$). Применяя формулу (7), свойства (2) и обозначения (8), находим, что

$$\Phi_{12} = 1 + K_5 \cos^2 2\varphi_{12} + 0{,}25 L_3 \sin^2 2\varphi_{12} - 2L_4 \cos 2\varphi_{12} + \\ + 0{,}5 \sin 4\varphi_{12} + (K_4 - L_2) \sin 2\varphi_{12}.$$

Для нахождения минимума Φ_{12}, дифференцируем это выражение по φ_{12} и получим уравнение

$$(L_2 - K_4 + L_5) t^4 + (4K_5 - L_3 + 4L_4) t^3 - 6L_5 t^2 - \\ - (4K_5 - L_3 - 4L_4) t - (L_2 - K_4 - L_5) = 0, \qquad (9)$$

где $t = \tan\varphi_{12}$.

(Практика показывает, что это уравнение имеет всегда по меньшей мере два действительных решения).

На плоскости $(f_1; f_3)$ сперва выпишем выражение для поворота осей

$$\bar{G}_{13} = T'_{13}\bar{G}_{12} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_{13}f_1 + s_{13}f_3 \\ f_2 \\ -s_{13}f_1 + c_{13}f_3 \end{pmatrix}.$$

Выражению

$$\Phi_{13} = E[(x_3 - x_1x_2)^2] = E(x_3^2) + E(x_1^2x_2^2) - 2E(x_1x_2x_3)$$

можно теперь придать вид

$$\Phi_{13} = 1 + K_5\cos^2\varphi_{13} + K_8\sin^2\varphi_{13} + (L_6 + K_3 - K_7)\sin 2\varphi_{13} - 2L_4\cos 2\varphi_{13}.$$

Приравнивая производное по φ_{13} нулю, получим для искомого угла

$$\tan 2\varphi_{13} = \frac{2(K_7 - K_3 - L_6)}{4L_4 + K_8 - K_5}. \qquad (10)$$

Вторая производная функции Φ_{13} при вычисленном значении $\tan 2\varphi_{13}$ (10) имеет вид

$$\Phi''_{13}(\varphi_{13}) = \pm 2\sqrt{(4L_4 + K_8 - K_5)^2 + 4(K_7 - K_3 - L_6)^2}.$$

Следовательно, существует как минимальное, так и максимальное значения функции Φ_{13}.

На плоскости $(f_2; f_3)$ составим выражение

$$\bar{G}_{23} = T'_{23}\bar{G}_{13} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ c_{23}f_2 + s_{23}f_3 \\ -s_{23}f_2 + c_{23}f_3 \end{pmatrix},$$

а минимизировать нужно теперь

$$\Phi_{23} = 1 + K_5\cos^2\varphi_{23} + L_1\sin^2\varphi_{23} + (L_7 + K_6 - K_9)\sin 2\varphi_{23} - 2L_4\cos 2\varphi_{23}.$$

Его минимальное значение достигается при

$$\tan 2\varphi_{23} = \frac{2(K_9 - K_6 - L_7)}{4L_4 + L_1 - K_5}. \qquad (11)$$

Соответствующие формулы для нахождения углов φ_{12}, φ_{13} и φ_{23} по Макдональду [1; 3] имеют вид

$$\tan 2\varphi_{12} = \frac{L_2 - K_4}{2L_4},$$

$$\tan 2\varphi_{13} = \frac{K_7 - K_3}{2L_4}, \qquad (I2)$$

$$\tan 2\varphi_{23} = \frac{K_9 - K_6}{2L_4}.$$

Они получаются как частные случаи из (I0) и (II), при дополнительных довольно строгих условиях, которых у Макдональда нет.

3. Нелинейная модель регрессионного типа

Найдем такие величины a и b, чтобы $ax_1x_2 + b$ лучше всего определяло x_3 при помощи x_1 и x_2. Для этой цели составим выражение

$$\Phi = E[(x_3 - ax_1x_2 - b)^2]$$

и приравняем его частные производные по a и b нулю. Таким путем получим, что

$$a = \frac{L_4}{K_5} \quad \text{и} \quad b = 0.$$

Так как

$$\frac{\partial^2\Phi}{\partial a^2}\cdot\frac{\partial^2\Phi}{\partial b^2} - \left(\frac{\partial^2\Phi}{\partial a\partial b}\right)^2 = 4E(x_1^2x_2^2) > 0$$

и

$$\frac{\partial^2\Phi}{\partial a^2} = 2E(x_1^2x_2^2) > 0,$$

то функция Φ имеет действительно минимальное значение.

Предполагая, что $x_3 \approx ax_1x_2$, проводим повороты на отдельных плоскостях.

На плоскости $(f_1; f_2)$ получаем уравнение

$$(L_2 - K_4 + aL_5)t^4 + (4aK_5 - aL_3 + 4L_4)t^3 - 6aL_5t^2 -$$
$$-(4aK_5 - aL_3 - 4L_4)t - (L_2 - K_4 - aL_5) = 0, \qquad (I3)$$

где $t = \tan\varphi_{12}$.

На плоскостях $(f_1; f_3)$ и $(f_2; f_3)$ выражения для определения угла поворота следующие:

$$\tan 2\varphi_{13} = \frac{2(K_7 - K_3 - aL_6)}{4L_4 + aK_8 - aK_5}, \qquad (I4)$$

$$\tan 2\varphi_{23} = \frac{2(K_9 - K_6 - aL_7)}{4L_4 + aL_1 - aK_5}.$$

При $a = 1$ из этих отношений получим выражения (9), (10) и (11) настоящей статьи.

При $a = 0$ формулы (14) есть формулы Макдональда (12) для выражения углов φ_{13} и φ_{23}. Чтобы проверить формулу Макдональда (12), из (13) при $a = 0$ запишем

$$(L_2 - K_4)t^4 + 4L_4t^3 + 4L_4t - (L_2 - K_4) = 0.$$

Разложим левую часть этого уравнения на множители:

$$(L_2 - K_4)(t^2 + 1)\left[t^2 + \frac{4L_4}{L_2 - K_4}t - 1\right] = 0.$$

Выразим из (12) при помощи тригонометрических формул $\tan 2\varphi_{12}$ как

$$\tan\varphi_{12} = \frac{-2L_4 \pm \sqrt{4L_4^2 + (L_2 - K_4)^2}}{L_2 - K_4}.$$

Теперь легко убедиться, что данный $\tan\varphi_{12} = t$ удовлетворяет нашему уравнению. Значит, уравнение (13) и формулы (14) являются более общими по сравнению с формулами (12), полученными Макдональдом [3].

4. Численный пример

В качестве примера приведем в таблице I часть матрицы факторных значений, полученных при помощи метода главных компонент [2], соответствующую трем первым наибольшим собственным числам.

Таблица I

Факторные значения		
0,7759	-0,1566	1,0276
-0,5355	-0,1886	1,5258
-0,4532	0,5708	0,6217
1,2841	1,3638	-1,5165
0,2512	1,3378	1,5111
-1,1367	0,7240	0,8428
-1,1579	0,4206	-0,0983
0,4219	1,1076	-2,4301
1,4613	-0,6903	-0,1040
-0,3139	0,2713	-0,6395
-0,6339	-0,1409	-1,1352
-0,5585	0,9798	0,9154
-0,9759	-0,1561	-0,0853
-0,4481	0,9288	-0,2346
-0,9528	-0,1668	0,1957
0,3814	1,1595	-1,1034

-0,6294	-0,5I9I	-I,I892
-0,I278	0,2488	-0,I737
0,08I2	-I,4990	-0,5628
-0,3694	-2,0307	0,3798
-0,3066	-2,5327	0,I243
-0,6847	-I,I6I6	-0,7I29
0,7I75	I,I627	2,2I32
-0,3749	0,6I70	-0,2975
3,7552	-0,6437	0,I772
0,59I6	-0,756I	0,8374
-0,0620	-0,2505	-0,0890

Вычисления для получения нелинейной факторной модели выполнялись а) по формулам Макдональда (I2); б) по формулам (9; I0; II); в) по формулам (I3; I4) и г) по формулам, выведенным в статье [I]. Критерием точности итерации принимались: во-первых $|\Delta\varphi| < 0{,}03$ радиана для углов поворота на каждой плоскости и во-вторых величина $\left|\frac{\Delta\Phi}{\Phi}\right|$. По последнему критерию изучалась сходимость с относительной точностью 0,02 и 0,0I, а для формул статьи [I] сходимость получалась при 0,06.

Характерные данные вычислений для сравнения результатов приведены в таблице 2. По данному материалу является выгодным преобразование линейной модели (I) в нелинейную (4) при помощи формул (9; I0; II). В таблице 3 приведена динамика итерации по этому методу, а в таблице 4 окончательная матрица факторных значений типа (4) и отдельно еще действительные произведения $x_{1j}\,x_{2j}$.

В таблицах 2 и 3 в столбце "R" приведены корреляционные коэффициенты между $x_1 x_2$ и x_3.

Таблица 2

Сравнение результатов

		R	$E[(x_3-x_1x_2)^2]$	$E(x_1^2x_2)$	$E(x_1^2x_2^2)$	$E(x_1x_2^2)$
Начальные данные		-0,2129	1,8666	-0,2620	0,5507	0,0339
Формулы (12)	$\lvert\Delta\varphi\rvert < 0{,}03$	-0,5268	4,2646	-0,1865	1,8368	0,2028
Макдональда	$B < 0{,}01$	-0,5265	4,2658	-0,1781	1,8382	0,2086
Формулы (9; 10; 11) ($a=1$)	$\lvert\Delta\varphi\rvert < 0{,}03$	0,6816	0,5946	0,0019	0,8142	-0,0204
	$B < 0{,}01$	0,6969	0,5612	-0,0179	0,8032	-0,0648
Формулы (13; 14) $a=-0{,}2869$	$\lvert\Delta\varphi\rvert < 0{,}03$	-0,5807	3,7368	-0,1843	1,6330	-0,0877
	$B < 0{,}01$	-0,6592	3,3487	0,1146	0,9920	0,0721
Статья [1]	$B < 0{,}06$	-0,5418	3,1542	-0,0392	1,0532	0,0363

Примечание: $B = \left|\frac{\Delta\Phi}{\Phi}\right|$.

Таблица 3

Динамика итерации

	Углы поворота осей			R	$E[(x_3 - x_1 x_2)^2]$
	φ_{12}	φ_{13}	φ_{23}		
0				-0,2I29	I,8666
I	I,4045	-0,7535	-0,3539	0,23I2	I,0674
2	-0,0707	-0,I442	-0,0994	0,3592	0,9404
3	-0,0082	-0,0I25	-0,2II6	0,4560	0,86I8
4	-0,05I8	-0,096I	-0,0I83	0,5I73	0,8I58
5	0,0077	0,0094	-0,I350	0,54I8	0,7900
6	-0,0500	-0,0657	0,0I5I	0,5746	0,7700
7	0,0087	0,0II8	-0,I026	0,58I0	0,7270
8	-0,0052	-0,093I	-0,0749	0,632I	0,6656
9	-0,0296	-0,0422	-0,0697	0,656I	0,6328
I0	-0,0330	-0,0430	-0,0364	0,6723	0,6I04
II	-0,0288	-0,0306	-0,0335	0,68I6	0,5946
I2	-0,024I	-0,0309	-0,02I6	0,6882	0,5827
I3	-0,0232	-0,0246	-0,0I83	0,6923	0,5736
I4	-0,0I88	-0,0238	-0,0I30	0,6952	0,5666
I5	-0,0I80	-0,0I94	-0,0I08	0,6969	0,56I2

Таблица 4

Окончательные факторные значения

x_{1j}	x_{2j}	x_{3j}	$x_{1j}x_{2j}$
-0,2I28	-0,9937	-0,806I	0,2II4
-I,4I43	-0,7990	-0,I080	I,I30I
-0,4865	-0,575I	0,59I9	0,2798
2,3476	-0,I479	0,5249	-0,3473
-0,3074	-I,9323	0,5549	0,5940
-I,0234	-0,5363	I,09I6	0,5489
-0,5422	0,3272	I,06I3	-0,I774
2,2732	I,0I89	I,0509	2,3I6I
0,77I7	-0,I598	-I,4I48	-0,I233
0,3098	0,4552	0,5272	0,I4I0
0,2638	I,I733	0,5I39	0,3095
-0,59I7	-0,9665	0,9086	0,57I9
-0,6486	0,55II	0,5095	-0,3575
0,2075	-0,I392	I,0277	-0,0289
-0,8I95	0,3406	0,43I6	-0,2792
I,406I	0,032I	0,8540	0,0452
0,I586	I,4I38	0,2363	0,2243
0,I223	0,0475	0,3020	0,0058
0,I487	I,I834	-I,07I4	-0,I760
-I,2587	0,96I9	-I,3766	-I,2I07
-I,24I5	I,3927	-I,7444	-I,729I
-0,4298	I,4307	-0,3080	-0,6I49
-0,953I	-0,6357	0,5997	0,6058
0,I786	0,0437	0,7589	0,0078
2,I227	-I,3430	-2,870I	-2,8508
-0,438I	-0,4562	-I,I059	0,I998
-0,078I	0,2254	-0,I328	-0,0I76

Литература

I. Выханду Л.К., Крусберг Х.О. О нелинейном факторном анализе. -"Тр. Таллинск. политехн. ин-та", 1977, № 426, с. 3.

2. Харман Г. Современный факторный анализ. М., "Статистика", 1972.

3. McDonald, R.P. Factor interaction in nonlinear factor analysis.- Brit. J. Math. and Statist.Psychol., 1967, 20.

L. Vyhandu, H. Krusberg

About Nonlinear Factor Analysis II

Summary

In this paper some generalizations to R.P. McDonald's nonlinear factor analysis method are considered.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 439 1978

УДК 518.5

Х.Я.Хярсинг

ДОПОЛНИТЕЛЬНЫЕ СРЕДСТВА К СИСТЕМЕ PL/I ДОС/ЕС

Подмножество языка программирования PL/I ДОС/ЕС [1, 2, 3] имеет много ограничений сравнительно с полным языком PL/I. Эти ограничения осложняют технику программирования. Ниже рассмотрим некоторые дополнительные средства к системе PL/I ДОС-ЕС, чтобы расширить возможности техники программирования в данной системе и тем самым освободиться от ограничений.

Все дополнительные средства разработаны как модули языка Ассемблера и оформлены в виде внешних процедур для программ PL/I по правилам [2]. Обращение к этим процедурам производится с помощью оператора типа CALL или через обращения к функции. В обоих случаях при обращении к этим процедурам аргументы, если неявное объявление не подходит, необходимо объявить явно с требуемыми атрибутами. При обращении к процедуре как к функции дополнительно потребуется объявить атрибуты имени точки входа (ENTRY).

1. Средства присваивания и передачи данных

Процедура присваивания MOVEC

Процедура предусмотрена для присваивания значения (значений) одной переменной (выражения) другой переменной.

Обращение к процедуре

```
CALL MOVEC (N,A,I,B,J);,
```

где N – количество присваиваемых (переписываемых) байтов, максимальное значение – 256;

A – имя переменной, значение которой присваивается B;

B – имя переменной, которой присваивается N байтов значения переменной A;

I – порядковый номер первого переписываемого байта, считая с начала переменной A;

J – порядковый номер первого байта, которому присваивается новое значение, считая с начала переменной B.

Аргументы N, I и J требуется объявить с атрибутами BINARY FIXED(31).

Аргумент A может быть именем любой переменной класса проблемных данных, выражением или константой. Например, аргументом A может быть массив, структура или структура, содержащая массивы с разными атрибутами и т.д. Допускается даже в качестве A использовать имена управляющих данных, например, типа ENTRY или LABEL, но в таком случае часто осложняется применение этих данных в виде переменной A.

Аргументом B может быть имя любой переменной класса проблемных данных.

При перезаписи любые данные формально рассматривают как строку знаков (STRING). При случае чрезмерного значения N, перезаписываемые байты могут переходить за границу полей объектов A или B, при этом программных предупреждений не будет.

Процедура MOVEC заменяет языковую конструкцию

```
SUBSTR(B,J,N)=SUBSTR(A,I,N);
```

но процедура MOVEC имеет следующие преимущества перед упомянутой конструкцией:

– количество байтов N может быть переменной, а не константой как в подмножестве PL/I ДОС/ЕС;

– объекты A и B могут быть любого типа данные.

Процедура реализована на командах EX и MVC.

Передача данных из одного шага задания в другой

Часто возникает потребность передать некоторые данные (например, признаки) из одного шага задания в другой.

Для этой цели служат процедуры WRITCOM и READCOM.

В некотором шаге $Ш_n$, откуда необходимо передать данные в следующие шаги $Ш_{n+k}$, в программу записывают оператор

```
CALL WRITCOM (A);
```

здесь A – имя переменной или выражение, значение которых необходимо передать в следующие шаги.

В шаге (шагах), где используют данные, полученные от переменной A, пишут оператор

```
CALL READCOM (B);
```

здесь B – имя переменной, тип и строение (структура) которой обычно такие же, как и у объекта A.

Для передачи данных пользуются полем области связи. Процедура WRITCOM записывает II байтов с начала поля объекта A на поле пользователя области связи, а процедура READCOM считывает оттуда II байтов данных и присваивает их переменной B, занося также с начала поля. A и B могут быть любого типа скалярные или агрегатные данные. Данные, записанные с помощью процедуры WRITCOM, можно считывать из области связи с процедурой READCOM многократно и в нескольких шагах.

2. Процедура управления магнитной лентой

Обращение к процедуре имеет следующий вид:

```
CALL MTC (F,C[,N]);
```

здесь F – имя файла;

C – мнемонический код операции управления с МЛ (в виде трехсимвольной константы (строка символов) или переменной, которая объявлена как CHAR (3);

N – количество повторений данной операции; N можно опустить – в таком случае операция управления выполняется один раз; N объявляется как BINARY FIXED (31).

Процедура MTC позволяет выполнить те же 8 операций управления, что и управляющая карта MTC:

BSF – перемотка ленты на N блоков вперед;

BSR – перемотка ленты на N блоков назад;

FSF – перемотка ленты на N ленточных маркеров вперед;
FSR – перемотка ленты на N ленточных маркеров назад;
REW – перемотка ленты в начало;
RUN – перемотка ленты в начало и снятие ее с механизма;
ERG – стирание межблочного промежутка;
WTM – записывание ленточных маркеров.

Процедура построена на базе обращения к логическому модулю ввода-вывода аналогично ассемблеровской макрокоманде CNTR.

3. Средства передачи данных при обращении к процедуре

Основным недостатком передачи значений аргументов к параметрам является строгое соответствие их количеств. Как показывает изучение системы, требование соответствия количества аргументов и параметров вызвано скорее отсутствием самого значения количества аргументов внутри процедуры, чем прямой необходимостью строгого соответствия этих количеств. Система PL/I ДОС/ЕС не устанавливает никаких ограничений по требованию этого соответствия.

Для устранения этого недостатка разработаны нижеследующие средства.

Функция ARGNO

Обращение к функции ARGNO следует писать в процедуре, где нужно определить количество аргументов при обращении к данной процедуре. Значение функции есть подсчитанное количество аргументов. Атрибуты функции объявляют следующим образом: DCL ARGNO [ENTRY] RETURNS (BIN FIXED(31));. Сама функция не имеет никаких аргументов.

Надо обратить внимание на следующий факт. Если функцию ARGNO применяют внутри процедуры, к которой обращаются или в данном случае обращались как к функции, то результат функции ARGNO будет на I больше. Значение количества аргументов функция ARGNO определяет из оператора обращения к процедуре, где присутствует команда STM, с помощью которой записывают все адреса аргументов на соответствующее поле в DSA той программы, откуда произойдет обращение.

Внутри процедуры или функции, количество аргументов которой определяют с помощью ARGNO, пустим на обработку столько же параметров (считая с начала списка), сколько присутствует при данном обращении аргументов. Однако количество аргументов не должно превышать количества параметров. При этом сохраняется и ограничение, обусловленное конструкцией, которое не допускает количества аргументов и параметров более I2.

Процедура RVALUE

Если произойдет обращение к функции, количество параметров которой не равно количеству аргументов данного оператора обращения, возникает нарушение при адресации значения результата функции. Для устранения этого нарушения разработана вспомогательная процедура RVALUE.

Обращение к данной процедуре имеет следующий вид:

```
CALL RVALUE;.
```

Обращение к RVALUE можно записать в любое место текста процедуры, к которой намереваются обратиться как к функции. При этом единственным условием является требование, чтобы при выполнении процедуры проходили по крайней мере I раз через CALL RVALUE.

Функция ARGADDR

Функция ARGADDR (I) дает в качестве результата адрес аргумента с порядковым номером I, причем аргументы пронумерованы слева направо от единицы до N.

Переменную I следует объявить с атрибутами BIN FIX(3I);, а результат функции – в виде

```
DCL ARGADDR RETURNS (POINTER);.
```

При значении I = 0, результат функции получает значение NULL. Неправильное значение I > N не проверяется. Внутри процедуры можем с помощью ARGADDR определить адрес нужного аргумента, который присваивают переменной типа указатель (POINTER). На указатель объявляется базированная переменная, через которую обрабатывается

внутри процедуры значение аргумента. При таком способе передачи данных в распоряжение процедуры не требуются вообще параметры. Однако не исключается совместное использование функции ARGADDR с параметрами.

4. Средства для управляемой памяти

Средства, которые рассматриваются в данном разделе, служат для создания и использования управляемой памяти, возможности организации которой в PL/I ДОС/ЕС отсутствуют.

Процедура CSTOR.

Процедура предусмотрена для создания управляемой памяти (CSA) в данном блоке. Обращение к процедуре

```
CALL CSTOR;
```

не имеет аргументов. Оператор обращения требуется записать в данном блоке один раз в начале блока, в котором необходимо создать CSA. CSA создается после динамической памяти данного блока.

Процедура ALLOC.

Процедура служит для создания поля с заданной длиной в CSA. Обращение к процедуре имеет следующий вид

```
CALL ALLOC(P[,N[,'D']]);.
```

Здесь P – переменная типа указатель, идентифицирующая созданное поле;

N – длина созданного поля в байтах (объявляется как BINARY FIXED (31);. При отсутствии N см. "Поколения данных в CSA";

'D' – признак, указывающий на то, что создаваемое поле состоит из двойных слов. В этом случае происходит выравнивание начала создаваемого поля на границу двойного слова, при отсутствии 'D' только на границу слова.

Процедурой создается поле длиной N байтов, начальный адрес которого присваивают указателю P. До начала поля процедура создает так называемый описатель поля CSA длиной

в 3 слова. Описатель содержит следующие данные:

- адрес указателя P,
- номер поколения данных,
- длина поля данных,
- указатель поля предыдущего поколения,
- указатель поля следующего поколения,
- указатель до этого созданного поля данных U,
- признак выравнивания на границу двойного слова.

Процедура FREE

Процедура служит для освобождения памяти от поля или ликвидации всей управляемой памяти. Оператор обращения к процедуре имеет вид

```
CALL FREE [(P)];
```

здесь P – указатель поля, из-под которого желают освободить память.

При обращении к процедуре без аргумента P ликвидируется вся управляемая память данного блока. Полная ликвидация CSA также осуществляется при выходе из блока с помощью оператора RETURN или END в конце блока.

При использовании FREE с указателем память от поля физически освобождается только в том случае, когда это поле последнее активное в CSA. В любом случае указателю присваивают значение NULL и в описателе отмечают, что данное поле ликвидировано.

Функция FRSTOR

Функция FRSTOR дает длину свободной памяти, из которой с помощью ALLOC можно выделить требуемые поля. Функция не имеет аргументов. Значение результата функции объявляется как BINARY FIXED (31).

Функция POINTR.

Функция POINTR дает значение указателя, которое определяет адрес памяти, смещенный относительно заданного указателя на величину N, выражающей смещение.

В обращении к функции

```
POINTR (N, P)
```

N – переменная или выражение, указывающие на значение смещения, объявляется как BIN FIXED(31);

P – указатель, относительно которого учитывается смещение.

При использовании этой функции обычно P получает значение в операторе ALLOC, т.е. P установлен на начало одного из полей CSA, а указатель, который является результатом этой функции, имеет значение адреса, находящегося внутри этого поля.

Использование средств управляемой памяти

Так как в PL/I ДОС/ЕС отсутствуют переменные типа CONTROLLED [4], приходится CSA создавать с помощью процедуры CSTOR в начале блока, а идентификатором создаваемого поля данных служит указатель, которому присваивают начальный адрес поля. Все необходимые поля в CSA создают с помощью процедуры ALLOC, но для использования этих полей требуется на указатели последних объявить базированные переменные (массивы, структуры, простые переменные и т.п.). Использование этих базированных переменных в операторах программы ничем не отличается от любых переменных с атрибутам BASED. Однако после прохождения оператора

CALL FREE (P)

отменяется доступ к данным, базированным на указатель P.

Для удобства обработки массивов, которые не объявлены в CSA как массивы, введена новая функция POINTR, с помощью которой можно некоторому указателю PE последовательно присваивать значения адресов элементов (строк) массива.

Пример. Обработать последовательно массив, состоящий из 100 одинаковых структур ST в управляемой памяти. После обработки освободить память от массива.

Соответствующий отрезок программы:

```
.....
CALL CSTOR;
.....
DCL(P, PE) POINTER,
```

```
    1 ST BASED (P),
      2 NR DEC FIX(3),
      2 NIM CHAR (15),
      2 T BIT (8),
      2 SUM DEC FIX (3),
   (K,N) BIN FIX (31);
   N = 2000;
   CALL ALLOC (P,N);
   DO I=1 TO 100;
   K=20*(I-1);
   PE = POINTR (K,P);
.....
(обработка структуры ST как элемента массива)
.....
END;
CALL FREE (P);
.....
```

Управляемые поля данных и блочная структура

Так как CSA может создаваться отдельно в каждом блоке программы при каждом обращении, и резервируется за DSA, включаясь в него, то относительно CSA в силе те же правила организации, что и для DSA. Поэтому при обращении из какого-либо блока к другому блоку, CSA созданная до обращения, будет недоступна, но после возврата снова будет доступна. А при рекурсивном обращении к процедуре можно создавать столько же поколений CSA, какой будет глубина рекурсивных обращений.

Поколения данных в CSA

При повторном прохождении оператора

```
CALL ALLOC (P,N);
```

с тем же именем указателя создается каждый раз новое поле для нового поколения тех же самых данных, которые базированы на указателе P. Доступным будет только последнее поколение данных. При этом указывать длину поля N не обязательно. Без указания N длина нового поколения берется из описателя старого поколения.

Применяя повторно оператор

CALL FREE (P);,

ликвидируют поколения друг за другом, начиная с последнего, при этом будут доступны предыдущие поколения.

Пример. Приведем отрезок программы, где будут доступны разные поколения структуры S.

```
    . . .
CALL CSTOR;
DCL  P POINTER,
   1S BASED (P),
    2 (A,B) BIN FLOAT (53),
            N BIN FIX (31);
N = 20;
CALL ALLOC (P, N,'D');
    . . .
```

(доступно I-е поколение S)

```
    . . .
CALL ALLOC (P);
    . . .
```

(доступно 2-е поколение S)

```
    . . .
CALL ALLOC (P);
    . . .
```

(доступно 3-е поколение S)

```
    . . .
CALL FREE (P);
    . . .
```

(доступно 2-е поколение S)

```
    . . .
CALL FREE(P);
    . . .
```

(доступно I-е поколение S)

```
    . . .
CALL ALLOC(P);
    . . .
```

(доступен новый экземпляр 2-го поколения S)

```
    . . .
CALL FREE (P);
    . . .
```

(доступно I-е поколение S)

```
    . . .
CALL FREE (P);
    . . .
```

(полностью ликвидированы все 20-байтовые поля для всех поколений структуры S , а указателю P присвоено значение NULL).

Максимально допустимое количество поколений для каждого поля данных в CSA - 127.

5. Средства отладки

Наряду с известными средствами распечатки содержимого основной памяти в PL/I DUMP и DYNDUMP разработаны еще процедуры SDUMP и DDUMP. Обе они действуют на базе известного в Ассемблере PDUMP-a, и выводят содержимое памяти в шестнадцатеричной системе.

SDUMP

Процедура предназначена для вывода содержимого статической памяти. Обращение к процедуре имеет вид:

```
CALL SDUMP [(N)];
```

здесь N - количество байтов, которое нужно распечатать, начиная от начала статической памяти. N объявляется как BIN FIXED (31).

При не указании количества байтов N распечатывается от начала статической памяти до конца данной фазы содержимое памяти.

DDUMP

Процедура предназначена для вывода содержимого динамической памяти. Обращение к процедуре имеет вид:

```
CALL DDUMP [(N)];
```

здесь N - количество байтов, которое требуется распечатать, начиная от начала динамической памяти. N объявляется как BIN FIXED(31).

При не указании количества байтов N распечатывается вся динамическая память, а если создана управляемая память, то и вся управляемая память.

6. Включение дополнительных средств в ДОС/ЕС

Для применения в программах вышеописанных средств следует эти процедуры и функции ввести в системную библиотеку объектных модулей дисковой операционной системы. Для этого все объектные модули процедур и функций в виде одно-

го пакета вводят с перфокарт или с магнитной ленты, назначенной логическим устройством SYSIN, с помощью служебной программы MAINT. А при редактировании проблемной программы, необходимые при этом процедуры или функции в режиме AUTOLINK, присоединяются к проблемной программе автоматически без всяких дополнительных управляющих карт задания.

Общая потребность памяти в библиотеке объектных модулей не больше 10 дорожек.

Литература

1. ЕС ЭВМ. Операционная система ДОС/ЕС.ПЛ/I. Пособие по языку. ЕІО.І32.070 ДІ, Минск, 1972.

2. ЕС ЭВМ. Операционная система ДОС/ЕС. ПЛ/I. Руководство для программиста. ЕІО.І32.07І ДІ, Минск, 1972, с.176-208.

3. ЕС ЭВМ. Операционная система ДОС/ЕС. ПЛ/I. Описание языка. ЕІО.І32.077 ДІ и Д2, Минск, 1974.

4. J ü r g e n s o n, R. PL/1 keele eriteemasid. TPI Tallinn, 1975, lk. 52-63 ja 68-91.

H. Härsing

Supplementary Means to System PL/I in DOS/ES

Summary

In this paper a method of the controlled allocating of the core storage and some other means for supplement to compiler PL/I in the operating system DOS/ES are described.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 439 1978

УДК 681.3.068

А.Гасенас, Р.Палуоя

ТРАНСЛЯТОР BASIC – ПЛ/I

Введение

В вычислительном центре Таллинского политехнического института разработан транслятор с языка BASIC (WANG 2200 В) на язык ПЛ/I (ЕС-1020 ДОС 2.I). Цель данного транслятора является ускорение этапа отладки программ. Отладка программы происходит на ЭВМ WANG 2200 В, после чего транслятор переводит программу на язык ПЛ/I. Под языком BASIC будем подразумевать язык BASIC для ЭВМ WANG 2200 В, а под языком ПЛ/I – ПЛ/I для операционной системы ДОС 2.I [10, 9, 11].

Минимашина WANG 2200 В [2,3,4,6] обладает большими программными и диалоговыми возможностями. При отладке программ возможен пошаговый режим выполнения программы, а в режиме TRACE (трассировка) на дисплее появляются имена изменяющихся переменных и присвоение им значения. В режиме редактирования можно тут же на дисплее исправить любую строку программы и запустить программу или с начала или с произвольно выбранной строки. Однако небольшой объем памяти (16 К байт) и сравнительно малая скорость (программа выполняется в режиме интерпретации) ограничивает сферу применения этой ЭВМ. С другой стороны, ЕС-1020 обладает достаточно большим объемом памяти и скоростью, но лишена указанных диалоговых возможностей. Язык BASIC является языком высокого уровня и удобен для изучения и использования. Конкретный вариант языка BASIC является расширенным вариантом языка, описанного в литературе [12]. В рассматриваемой версии есть мощные средства символьной обработки. Данный транслятор работает под управлением операционной системы ДОС 2.I.

I. Описание транслятора

Транслятор создан на базе системы построения трансляторов (СПТ), которая разработана в НЦ ТПИ [7]. Для использования данной СПТ должна быть описана контекстно-свободная грамматика исходного языка. В частности, грамматика языка BASIC имела более 200 правил подстановки. В результате работы анализатора получается синтаксическое дерево программы. Генерация объектного языка осуществляется проходом по дереву соответствующего предложения и выдачей заранее заготовленных кусков текста на ПЛ/I. Если получаемый кусок текста слишком большой (например, при переводе оператора получения обратной матрицы) или отсутствует на ПЛ/I аналогичная стандартная функция (например, NUM, которая определяет, сколько знаков в строке образует число по правилам грамматики языка BASIC), то вставляется обращение к подпрограмме или функции, которая находится в библиотеке перемещаемых модулей машины ЕС-1020. Объектным языком выран ПЛ/I, так как из языков высокого уровня он наиболее близок к языку BASIC по своим возможностям. С точки зрения трансляции можно выделить следующие отличия языка BASIC от языка ПЛ/I:

1. Скалярным переменным и массивам не присваиваются атрибуты.

2. Все переменные глобальные.

3. Можно изменять границы матриц и длину строковых переменных.

4. Подпрограммой может быть любой кусок программы, заканчивающийся предложением RETURN. При выполнении программы тело подпрограммы не обходится, а выполняется так же, как и главная программа (см. П.2).

5. На машине WANG 2200 В есть специальные операции для перекодировки, упаковки и распаковки данных в различных форматах.

Условно можно выделить два уровня трансляции. На первом уровне пользователь должен знать только несколько ограничений, налагаемых программе на языке BASIC (о втором уровне см. п.2).

Пример I.

Предположим, что программа на языке BASIC выглядит следующим образом:

```
10 COM A (100)
20 FORI = I TO 100
30 A(I) = I ↑ 2
40 NEXT I
50 PRINTUSING 90,"TABLE": PRINT
60 FOR I=1 TO 100 STEP 2
70 PRINTUSING 100, I, A(I), I+1, A(I+1)
80 NEXTI: END
90 % ␣␣␣␣␣␣####
100 % ␣␣##␣␣␣##␣␣##␣␣␣␣␣␣##␣␣␣##␣␣
```

Программа вычисляет квадраты чисел от I до 100 и печатает по четыре числа в строке: число, его квадрат, число, его квадрат. Предложения с номерами 50 и 70 – это печать переменных по формату. Число после слова PRINTUSING указывает, в какой строке находится нужный формат. После работы транслятора получается следующая программа на языке ПЛ/I:

```
B00000: PROCEDURE;
DCL AA(100) EXTERNAL;
DO I=1 TO 100;
AA(I)=I**2; END;
PUT SKIP EDIT ('TABLE')(X(6),A(5));
PUT SKIP;
DO I=1 TO 100 BY 2;
PUT SKIP EDIT (I,A(I),I+1,A(I+1))
(X(2),F(5.0),X(3),F(7.0),X(7),F(5.0),X(3),F(7.0));
END; END;
```

Название процедуры генерируется автоматически и печатается сообщение о названии программы.

2. Дополнительные возможности

В некоторых случаях желательно более широко использовать возможности ЭВМ ЕС-1020 или языка ПЛ/I. Для этих целей вводятся псевдокомментарии. На языке BASIC они воспринимаются как обычные комментарии. В процессе трансляции эти псевдокомментарии позволяют вставить куски текста на ПЛ/I и заменить получаемый стандартный текст на указанный в комментариях. Общий вид псевдокомментария следующий:

REMp текст

REM на языке BASIC означает, что далее следует комментарий, p – буква или специальный знак, указывающий тип псевдокомментария. Символ p должен следовать за словом REM без пробела, иначе комментарий воспринимается как обычный. Символ p может принимать следующие значения:

A – указывает, что в псевдокомментарии находятся атрибуты, которые должны быть присвоены переменным или массивам, находящимся в операторе DIM или COM, идущим перед этими псевдокомментариями (DIM и COM операторы описания массивов, но можно описывать и неиндексированные переменные);

& – указывает, что в транслируемую программу надо вставить кусок текста на ПЛ/I из псевдокомментария;

% – указывает, что в данном псевдокомментарии находится метка, которую следует вставить в транслируемую программу;

* – указывает, что предложение, идущее перед этим псевдокомментарием, надо игнорировать. В остальном действия такие как и при символе & (т.е. комментарии заменены);

F – этот псевдокомментарий применяется только с предложениями DIM и COM. Переменные и (или) массивы, описанные в DIM или COM, транслируются в структуру записей файла прямого доступа. В псевдокомментарии указывается название файла, длина записи и т.д., а также атрибуты, присваиваемые переменным и массивам предложений DIM или COM. Название файла должно быть идентификатором массива на языке BASIC. Если

этот идентификатор массива встречается в предложениях $PACK или $UNPACK (предложения упаковки и распаковки данных), то предложения $PACK или $UNPACK транслируются в предложения чтения или записи соответствующего файла. Указанный массив моделирует набор данных, находящийся на внешнем устройстве (на дисках). Эта конструкция позволяет отладить программу на массиве данных, находящемся в памяти, а после трансляции применять программу к набору данных на внешнем устройстве, т.е. на этапе отладки можно исключить работу с внешними устройствами.

Применяется еще один тип псевдокомментария для указания названия программы и (если есть) ее параметров. Этот псевдокомментарий должен быть первым предложением программы. Параметр P в данном случае не применяется. При отсутствии этого псевдокомментария название программы генерируется автоматически и она транслируется как процедура без параметров.

Пример 2.

Предположим, что программа на языке BASIC выглядит следующим образом:

```
10 REM PR PROCEDURE OPTIONS (MAIN);
20 DIM A$(10,10)20, A(10,10): REMA FIXED DECIMAL(3)
30 REM& DCL P POINTER EXTERNAL; P=ADDR(AA);
40 MAT IN PUT A$
50 REM *GET LIST(AAS);
60 FOR A=1 TO 10: FORI=1 TO 10
70 ADD(A$(A,I), OF)
80 GOSUB 110
90 NEXT I: NEXT A
100 LOAD "ANALIZ"
110 A(A,I)=VAL(A$(A,I))
120 IF A(A,I)<100 THEN 140
130 A(A,I)=A(A,I)-100
140 RETURN
```

Программа рассматривает массив строк как последовательность чисел и прибавляет к каждому байту шестнадцатеричную константу 0F. Если значение первого байта в строке больше ста, то из этого значения вычитается сто. В любом случае соответствующий элемент массива A получает это значение.

Пояснения к программе:

№ стр.	Комментарий
10	Первый комментарий указывает, является эта программа главной или нет, а также название программы. Если программа не главная, этот комментарий необязателен.
20	Определение массивов. Следующий комментарий определяет атрибуты массива A. Атрибуты массива A$ известны.
30	Комментарий вставки. Этот текст будет вставлен в программу на языке ПЛ/I.
40	Ввод значений матрицы A$ с клавиатуры машины WANG 2200 B.
50	Границы массивов могут быть переменными и матричные операции переводятся в несколько предложений. В данном случае границы массива не меняются и целесообразно вставить на место предложение в строке 40, одно предложение ввода матрицы. Эти действия производит данный комментарий замены. Первая буква массива удваивается, чтобы отличить на языке ПЛ/I скалярную переменную от идентификатора массива (массив A и переменная A).
70	Каждый байт A$(A,I) складывается по модулю 256 с шестнадцатеричной константой 0F.
80	Обращение к подпрограмме без параметров. Не надо забывать, что все переменные глобальные.
100	С магнитной ленты загружается в память программа с названием ANALIZ, которая выполняет какие-то дополнительные действия.
110	Встроенная функция VAL рассматривает первый байт символьной переменной как двоичное число без знака и это число присваивается элементу массива A(A,I).

I40 Возврат из подпрограммы.

Строки с номерами 60, 90, I20 и I30 в комментариях не нуждаются. После работы транслятора получается следующая программа на языке ПЛ/I:

```
PR: PROCEDURE OPTIONS (MAIN);
DCL DIM(286,2) FIXED DECIMAL(3);
DCL SDIM(286,2) FIXED DECIMAL(3);
DCL AAS(100) CHAR(20);
SDIM(1,1)=10; SDIM(1,2)=10;
DCL AA(100) FIXED DECIMAL(3);
DIM(1,1)=10; DIM(1,2)=10;
DCL P POINTER EXTERNAL; P=ADDR(AA);
GET LIST (AAS);
DO A=1 TO 10; DO I=1 TO 10;
CALL ADD(AAS((A-1)*SDIM(1,2)+I), 20,
        REPEAT(SIXTN('0F'),63),64);
MH = MH+1; LAB(MH) = G00000;
GOTO L0110;
G00000: END; END;
CALL ANALIZ; GOTO GALAS;
L0110: AA((A-1)*DIM(1,2)+I) = FVAL(AAS((A-1)*SDIM(1,2)+I),20);
IF AA((A-1)*DIM(1,2)+I)<100 THEN GOTO L0140;
AA((A-1)*DIM(1,2)+I) = AA((A-1)*DIM(1,2)+I)-100;
L0140: MH = MH-1; GOTO LAB(MH+1);
DCL MH FIXED BINARY(15)INIT(0);
DCL LAB(45) LABEL;
DCL SIXTN RETURNS(CHAR(1));
GALAS: END;
```

Поясним некоторые места полученной программы на языке ПЛ/I. Массив меток LAB программно реализует стэк, используемый для подпрограмм на ЭВМ WANG 2200 B. В массивах DIM и SDIM хранятся размерности декларируемых массивов. Это связано с динамическим переопределением границ массивов в ходе выполнения программ на ЭВМ WANG 2200 B. С этой особенностью связано и преобразование всех массивов в одномерные. Аргументом функции SIXTN является строка, состоящая из двух символов. Символы обозначают две шестнадцатеричные цифры, которые должны быть упакованы в один байт. SIXTN возвращает этот байт в качестве своего значения. REPEAT является встроенной функцией языка ПЛ/I. Первым аргументом должна быть строка знаков, а второй аргумент указывает, сколько раз сцепить заданную строку с собой. В данном случае получаемое значение есть строка знаков длиной в 64 байта и в каждом байте находится шестнадцатеричный код 0F. Процедура ADD производит сложение двух строк символов как двоичных чисел (см. пояснения к строке с номером 70).

В заключение следует сказать, что транслятор является экспериментом. Эффективность данного транслятора еще должна быть доказана. Однако некоторые выводы можно сделать уже сейчас. Программы, полученные с помощью данного транслятора, работают несколько медленнее, чем программы, написанные вручную на языке ПЛ/I. Использование транслятора требует некоторых навыков от программиста. С другой стороны, этап отладки значительно сокращается ввиду средств отладки, предоставляемых ЭВМ WANG 2200 B. Надо заметить, что, изменив семантические подпрограммы, можно получить транслятор на любой другой подходящий язык.

Достоинством является то, что без особых усилий можно удалить, заменить или вставить любые предложения или псевдокомментарии, допускаемые на языке BASIC.

Литература

1. Aho, A.V., Ulmann, J.D. The theory of parsing, translation and compiling. Vol. II. Compiling. USA, New Jersey, Prentice-Hall, 1973.

2. Option I. Matrix Rom. Reference manual. Wang laboratories Inc., 1974.

3. 2200 A/B BASIC. Programming manual. Wang laboratories Inc., 1974.

4. Saulenas, M. Wang BASIC-2 language. Reference manual. Wang laboratories Inc., 1976.

5. Кнут Д. Искусство программирования для ЭЦВМ. ТІ. Основные алгоритмы. М., "Мир", 1976.

6. Система "Ванг-2200 В". Руководство для пользователей. Рига, "Знание", 1974.

7. Вооглайд А.О., Томбак М.О. Система построения эффективных многопроходных трансляторов с LR(k)-семантикой. "Программирование", № 5, 1976.

8. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. М., "Мир", 1975.

9. Операционная система ДОС/ЕС. ПЛ/І. Пособие по языку Е ІО.І32.070 Дд.

ІО. Операционная система ДОС/ЕС. ПЛ/І. Описание языка. ЕІО.І32.072 Дд.

ІІ. Операционная система ДОС/ЕС. Руководство для программиста. Е ІО.І32.07І Дд.

12. Lee, J.A.N. The formal definition of the BASIC language.--The Computer Journal. Vol. 15, No 1, February 1972.

A. Gasenas, R. Paluoja

Translator BASIC-PL/1

Summary

In this paper a translator from BASIC to PL/1 is described.This translator enables us to speed up the program debugging. Translator is based on the compiler writing system made in the TPI computer centre.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 439 1978

УДК 519.271/272.

С.Тынисмяги-Геращенко

ОБ ИСПОЛЬЗОВАНИИ КРИТЕРИЕВ КАЧЕСТВА РЕГУЛИРОВАНИЯ

Системы автоматического регулирования с переменной структурой обладают рядом преимуществ ([1], [2]) перед обычными следящими системами. В связи с этим вопросы синтеза таких систем приобретают весьма важное значение. Одной из задач синтеза с переменной структурой является обеспечение устойчивого движения в плоскости скольжения и выбор параметров скольжения (параметров, определяющих положение гиперплоскости скольжения в фазовом пространстве), таким образом, чтобы некоторая оценка качества регулирования системы была наилучшей.

В данной работе решается задача о выборе оптимальных параметров скольжения. За оценку качества регулирования приняты интегральные критерии качества и степень устойчивости системы.

1. Постановка задачи

Как известно движение на плоскости скольжения описывается обыкновенным линейным дифференциальным уравнением $(n-1)$-го порядка

$$x^{(n-1)} + c_1(r)\,x^{(n-2)} + \ldots + c_{n-1}(r)\,x = 0, \qquad (1.1)$$

где x – регулируемая величина;

n – порядок системы регулирования;

$c_i(r)$ – коэффициенты, определяющие положение гиперплоскости скольжения в фазовом пространстве с координатами $(x_1, x_2, \ldots, x_n)$,

где

$$x_1 = x, \quad x_2 = \dot{x}, \quad \ldots \quad , x_n = x^{(n-1)}.$$

Известно ([I], [2]), что коэффициенты $c_i(r)$ имеют вид:

$$c_k(r) = r\,c_{k-1} + a_k, \quad k \in \overline{1, n-1}, \tag{I.2}$$

где a_k – постоянная величина;
r – параметр.

Возникает следующая задача: Выбрать значение r так, чтобы некоторый критерий качества принимал оптимальное значение.

В качестве критериев качества рассмотрим следующие критерии [3]:

I. Интегральные критерии.

а) обобщенная квадратичная мера

$$J_v = \int_0^\infty V\,dt, \tag{I.3}$$

где

$$V = \sum_{i=1}^{n-1} c_i x_i^2;$$

б)
$$J_{2t^2} = \int_0^\infty t^2 x^2(t)\,dt. \tag{I.4}$$

2. Степень устойчивости корней характеристического уравнения, соответствующего уравнению (I.I).

2. Обобщенная квадратичная мера качества переходного процесса

В общем виде вопрос о вычислении интеграла J_v решен А.А.Фельдбаумом [4]. Используем аналогичный прием.

Уравнение (I.I) запишем в виде системы дифференциальных уравнений:

$$\begin{aligned} \frac{dx_1}{dt} &= x_2, \\ &\cdots\cdots \\ \frac{dx_m}{dt} &= -c_m(r)\,x_1 - \ \cdots\ - c_1(r)\,x_m. \end{aligned} \tag{2.I}$$

Систему (2.I) перепишем в матричном виде:

$$\frac{dx}{dt} = Ax, \tag{2.2}$$

где A – матрица порядка $m \times m$;
x – столбец-вектор.

Воспользуемся матричной записью системы (2.2) и запишем J_ν в виде:

$$J_\nu = \int_0^\infty (x, Cx)\, dt, \qquad (2.3)$$

где C – матрица вида

$$C = \begin{pmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & c_m \end{pmatrix}$$

и (x, Cx) – скалярное произведение векторов x и Cx.

Если значения параметра r выбраны так, что движение в плоскости скольжения асимптотически устойчиво, то интеграл J_ν сходится, и, в силу непрерывности коэффициентов $c_i(r)$ как функций от параметра r, J_ν является функцией от r.

Найдем вариацию $J_\nu(r)$ при изменении параметра r:

$$\delta J_\nu(r) = 2\int_0^\infty (\delta_r x, Cx)\, dt, \qquad (2.4)$$

где

$$\delta_r x = \left(e^{A(r+\delta r)t} - e^{A(r)t}\right) x_0. \qquad (2.5)$$

Согласно работе Р.Беллмана [5],

$$\delta_r x = \delta r \int_0^t e^{A(r)(t-s)} \frac{dA}{dr} e^{As} x_0\, ds. \qquad (2.6)$$

Подставляя (2.6) в выражение для δJ_ν, получим:

$$\delta J_\nu(r) = 2\delta r \int_0^\infty \left(e^{At}\int_0^t e^{-As}\frac{dA}{dr}e^{As}x_0\, ds,\ Ce^{At}x_0\right) dt =$$

$$= 2\delta r \int_0^\infty \left(\int_0^t e^{-As}\frac{dA}{dr}e^{As}x_0\, ds,\ e^{A't}Ce^{At}x_0\right) dt =$$

$$= 2\delta r \left\{\left(\int_0^t e^{-As}\frac{dA}{dr}e^{As}x_0\, ds,\ e^{A't}Be^{At}x_0\right)\Big|_0^\infty - \right.$$

$$\left. - \int_0^\infty \left(e^{-At}\frac{dA}{dr}e^{At}x_0,\ e^{A't}Be^{At}x_0\right) dt \right\} =$$

$$= -2\delta r \int_0^{\infty} \left(x_0,\ e^{A't}\left(\frac{dA}{dr}\right)' B e^{At} x_0\right) dt, \qquad (2.7)$$

так как если движение асимптотически устойчиво, то

$$\lim_{t\to\infty} \left(\int_0^t e^{-A\delta} \frac{dA}{dr} e^{A\delta} x_0 \, d\delta,\ e^{A't} B e^{At} x_0 \right) dt = 0$$

и B – квадратная матрица порядка $m \times m$ такая, что $A'B + BA = C$. Матрица B будет матрицей определенно отрицательной квадратичной формы-функции Ляпунова для системы (2.I). В работе А.А.Фельдбаума [4] приведены формулы для нахождения матрицы B.

Для вычисления подинтегрального выражения в формуле (2.7) воспользуемся аналогичным приемом, а именно, будем искать матрицу M такую, что

$$A'M + MA = \left(\frac{dA}{dr}\right)' B. \qquad (2.8)$$

Тогда получим

$$dJ_v(r) = 2\delta r (x_0, Mx_0). \qquad (2.9)$$

Следовательно, экстремальные значения параметра r удовлетворяют уравнению

$$(x_0, Mx_0) = 0. \qquad (2.I0)$$

Для того, чтобы найти оптимальное значение r, необходимо найти вещественные корни уравнения (2.I0), взять те корни r_i', которые обеспечивают асимптотически устойчивое скольжение и сравнить между собой значения $J_v(r_i')$. Оптимальное значение параметра r^0 найдется из условия:

$$J(r^0) = \min_i J_v(r_i').$$

3. Второй интегральный критерий качества

Рассмотрим следующий критерий качества регулирования. Пусть

$$J_t(r) = J_{2t^2}(r) = \int_0^{\infty} t^2 x^2(r,t)\, dt.$$

По формуле Рэйли [4] для данного случая имеем

$$J_t(r) = \frac{1}{2\pi} \int_{c-j\omega}^{c+j\omega} \left| L(t \cdot x(t)) \right|^2_{p=c+j\omega} d\omega, \qquad (3.I)$$

где $L(x(t))$ – Лапласово изображение функции $x(t)$.

Так как

$$L(t\cdot x)=\frac{d}{dp}L(x),$$

то

$$J_t(r)=\frac{1}{\pi}\int_{c+0\cdot j\omega}^{c+\infty\cdot j\omega}\left|\frac{d}{dp}L(x)\right|^2_{p=c+j\omega}d\omega. \qquad (3.2)$$

Найдем $L(x)$ из дифференциального уравнения (I.I). Переходя к изображению Лапласа, вместо уравнения (I.I) получим:

$$(p^{n-1}+c_1(r)p^{n-2}+\ldots+c_{n-1}(r))\,x(p)=R(p,r,x_0), \qquad (3.3)$$

где $R(p, r, x_0)$ – полином, определенный начальными условиями.

Для простоты положим

$$x_0=x(0)\neq 0,\quad x_0^{(k)}=x^{(k)}(0)=0 \quad \text{при} \quad k>0.$$

Тогда

$$R(p,r,x_0)=p^{n-2}x_0+c_1(r)p^{n-3}x_0+\ldots+c_{n-2}(r)x_0,$$

следовательно,

$$x(p)=x_0\frac{p^{n-2}+c_1(r)p^{n-3}+\ldots+c_{n-2}(r)}{p^{n-1}+c_1(r)p^{n-2}+\ldots+c_{n-1}(r)}. \qquad (3.4)$$

Нетрудно заметить, что числитель и знаменатель выражения для $x(p)$ рассматриваются как функции от p и r, являются суммами отрезков геометрической прогрессии, поэтому, умножая числитель и знаменатель на $(p-r)$, получим:

$$x(p)=x_0\frac{f_{n-1}(p)-f_{n-1}(r)}{f_n(p)-f_n(r)}, \qquad (3.5)$$

где

$$f_{n-1}(p)=p^{n-1}+a_1p^{n-2}+\ldots+a_{n-1},$$

$$f_n(p)=pf_{n-1}(p).$$

Для нахождения интеграла $J_t(r)$ необходимо найти $\frac{dx(p)}{dp}$. При вычислении найдем, что

$$\frac{dx(p)}{dp}=x_0\frac{f_{n-1}(r)\left[\frac{df_n(p)}{dp}-r\frac{df_{n-1}(p)}{dp}\right]-f^2_{n-1}(p)}{[f_n(p)-f_n(r)]^2}. \qquad (3.6)$$

Итак, необходимо минимизировать интеграл вида

$$J_t(r)=X_0^2\int_{c-j\infty}^{c+j\infty}\left|\frac{f_{n-1}(r)\dfrac{df_n(p)}{dp}-\left[f_n r\dfrac{df_{n-1}(p)}{dp}+f_{n-1}^2(p)\right]}{[f_n(p)-f_n(r)]^2}\right|^2_{p=j\omega}d\omega=$$

$$=X_0^2\int_{c-j\infty}^{c+j\infty}\left\{f_{n-1}^2(r)\left|\frac{\dfrac{df_n(p)}{dp}}{[f_n(p)-f_n(r)]^2}\right|^2+\left|\frac{f_n(r)\dfrac{df_{n-1}(p)}{dp}+f_{n-1}^2(p)}{[f_n(p)-f_n(r)]^2}\right|^2-\right.$$

$$\left.-2f_{n-1}(r)\,\mathrm{Re}\,\frac{\dfrac{df_n(p)}{dp}\left[f_n(r)\dfrac{df_{n-1}(\bar p)}{dp}+f_{n-1}^2(\bar p)\right]}{[f_n(p)-f_n(r)]^2[f_n(\bar p)-f_n(r)]^2}\right\}_{p=j\omega}d\omega. \qquad (3.7)$$

Из полученной выше формулы следует, что

$$J_t(r)=f_{n-1}^2(r)J_1(r)-2f_{n-1}(r)J_2(r)+J_3(r),$$

т.е. имеем

$$J_t(r)=J_1(r)\left[f_{n-1}(r)-\frac{J_2(r)}{J_1(r)}\right]^2+J_3(r)-\frac{J_2^2(r)}{J_1(r)}\geqslant J_3(r)-\frac{J_2^2(r)}{J_1(r)}. \qquad (3.8)$$

Поэтому, если нужно уменьшить возможную нижнюю границу функции $J_t(r)$, то следует уменьшить величину

$$J_3(r)-\frac{J_2^2(r)}{J_1(r)}=\varphi(r).$$

Можно заметить, что достаточным условием экстремума $\varphi(r_1^\circ)$ является равенство

$$\frac{df_n(r_1^\circ)}{dr}=0. \qquad (3.9)$$

Кроме того, если считать $J_1(r)$, $J_2(r)$ и $J_3(r)$ постоянными и рассмотреть трехчлен

$$\varphi(r)=f_{n-1}^2(r)A-2f_{n-1}(r)B+C,$$

то достаточным условием того, что $\varphi(r)$ принимает экстремальное значение, является равенство

$$\frac{df_{n-1}(r_2^\circ)}{dr}=0. \qquad (3.10)$$

Вполне естественно поэтому предположить, что точка минимума функции $J_t(r)$ находится в интервале (r_1°, r_2°), целиком принадлежащем интервалу значений „r", обеспечивающих асимптотически устойчивое скольжение.

4. Степень устойчивости

Рассмотрим в качестве критерия качества регулирования степень устойчивости корней характеристического уравнения. Как известно, в этом случае необходимо выбирать параметры так (в данном случае параметр r), чтобы добиться величины $\min_i \mathrm{Re}[-\lambda_i(r)]$, где $\lambda_i(r)$ – корень характеристического уравнения $(i \in \overline{1,n-1})$. Известно, что характеристическое уравнение, соответствующее дифференциальному уравнению скольжения, можно привести к виду:

$$f_n(\lambda) - f_n(r) = 0. \qquad (4.1)$$

Уравнение (4.1) отличается от характеристического уравнения тем, что содержит один лишний корень $\lambda_n = r$.

Предположим, что найдены из уравнения (4.1) все корни λ_i как функции $s = f_n(r)$. Тогда подставляя $\lambda_i(s)$ в уравнение (4.1) и дифференцируя его по s, получим:

$$\frac{df_n(\lambda_i)}{d\lambda_i(s)} \cdot \frac{d\lambda_i(s)}{ds} = 1,$$

или

$$\frac{d\lambda_i(s)}{ds} = \frac{1}{\varphi_n(\lambda_i)}, \qquad (4.2)$$

где

$$\varphi_n(\lambda_i) = n\lambda_i^{n-1} + (n-1)a_1\lambda_i^{n-2} + \ldots + a_{n-1}.$$

Выберем s так, чтобы получить

$$\max_s \min_{i \in \overline{1,n}} \mathrm{Re}[-\lambda_i(s)].$$

Могут представиться две возможности:

1) максимум достигается вдоль одной ветви кривой $\lambda_{i'}(s)$. Тогда, дифференцируя вдоль этой ветви $\lambda_{i'}(s)$, получим, что минимум по s достигается в точках s, соответствующих условию

$$\frac{d\mathrm{Re}[-\lambda_i(s)]}{ds} = 0$$

или

$$\varphi_n(\lambda) + \varphi_n(\bar{\lambda}) = 0. \qquad (4.3)$$

Решая уравнение (4.3) совместно с (4.1), найдем множество значений, обеспечивающих $\max_s \min_i \mathrm{Re}[-\lambda_i(s)]$.

2) $\max_s \min_i \mathrm{Re}[-\lambda_i(s)]$ достигается в тех точках, где нарушается единственность решения дифференциального уравнения (4.2), т.е. $\frac{d\lambda}{ds} = \infty$. Но в таких точках вы-

полняется уравнение

$$\varphi_n(\lambda_i) = 0, \qquad (4.4)$$

которое означает, что уравнение (4.I) имеет кратный корень. Поэтому ясно, что условие (4.4) будет достаточным условием для выполнения равенства (4.3).

Итак, можно сформулировать теорему.

Теорема. Значения r°, обеспечивающие экстремум степени устойчивости, являются корнями уравнения

$$D\left(f_n(\lambda) - f_n(r),\ \frac{df_n(\lambda)}{d\lambda}\right) = 0.$$

В работе [2] рассмотрена система третьего порядка при

$$n = 3,\quad a_1 = a_2 = a_3 = 0,$$

$$\dot{x}_1 = x_2,$$

$$\dot{x}_2 = x_3,$$

$$\dot{x}_3 = u,$$

$$u = -\psi x_1 - \delta u, \qquad \varphi = \begin{cases} \alpha & \text{при } x_1\sigma > 0, \\ \beta & \text{при } x_1\sigma < 0, \end{cases}$$

где $\sigma = c_1x_1 + c_2x_2 + x_3$.

Коэффициенты c_1, c_2 уравнения плоскости скольжения выбирались с помощью интегрального критерия качества и "оценки степени устойчивости" скользящих движений. Результаты по этому примеру [2]согласуются с излагаемыми в данной статье. Причем из корней уравнения $dJ_t(r)/dr = 0$ нужно выбрать те, которые обеспечивают устойчивость движения в скользящем режиме.

И еще один пример к пункту 4.

Пример.

$$n = 3,\ a_1 = 2,\ a_2 = 1.$$

$$\lambda(\lambda^2 + 2\lambda + 1) - \delta = 0,$$

$$3\lambda^2 + 4\lambda + 1 = 0,\quad \lambda_1^\circ = -1,\quad \lambda_2^\circ = -\tfrac{1}{3},$$

$$\delta_1 = -1(1 - 2 + 1) = 0,\ \text{т.е. } r_1^\circ = 0 \text{ и}$$

$$D\left(f_n(\lambda) - f_n(r),\ \frac{df_n(\lambda)}{d\lambda}\right) =$$

$$= \lambda^2 + (2+r)\lambda + (r^2 + 2r + 1) = 0,$$

$$\lambda_{1,2}(r=0) = \lambda^0_{1,2} = -1.$$

Литература

1. Барбашин Е.А. Введение в теорию устойчивости. М., "Наука", 1967.

2. Уткин В.И. Скользящие режимы и их применение в системах с переменной структурой. М., "Наука", 1974.

3. Брайсон А., Хо Ю-Ши. Прикладная теория оптимального управления. М., "Мир", 1972, гл.14.

4. Фельдбаум А.А. Основы теории оптимальных автоматических систем. М., "Наука", 1966.

5. Беллман Р. Введение в теорию матриц. М., "Наука", 1969.

S. Tõnismägi-Gerashchenko

On Use of the Criterions for Control Performance

Summary

Integral criterion of quality and stability degree of the system of differential equations are used for choice of the parameters defining the position of sliding giperplane in the phase space.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 439 1978

УДК 517.5

Ю.М.Гиршович

ОПТИМАЛЬНОЕ ВОССТАНОВЛЕНИЕ И ДИФФЕРЕНЦИРОВАНИЕ ФУНКЦИЙ

Обозначим через W^rL_q множество всех функций $f(x)$, имеющих на $[0, 1]$ абсолютно непрерывную производную порядка $r-1$ и удовлетворяющих условию

$$\|f^{(r)}\|_q = \|f^{(r)}(t)\|_{L_q(0,1)} \le 1.$$

Пусть задано натуральное число m, $0 \le m \le r-1$. Нас интересует приближенное вычисление значения $f^{(m)}(u)$ по известным значениям $f(x_1), \dots, f(x_n)$, где $u, x_1, \dots, x_n \in [0,1]$. В соответствии с результатами [1] мы будем рассматривать только линейные формулы приближения

$$f^{(m)}(u) = \sum_{k=1}^{n} A_k f(x_k) + r(f; u; x; A). \qquad (1)$$

Величина

$$r(u; x; A) = \sup_{f \in W^rL_q} r(f; u; x; A)$$

называется точной оценкой ошибки формулы (1) на множестве W^rL_q. При фиксированных $u, x_1, \dots, x_n \in [0,1]$ назовем формулу (1) оптимальной с фиксированными узлами на множестве W^rL_q, если коэффициенты $A_1, \dots, A_n$ доставляют величине $r(u; x; A)$ наименьшее значение, которое обозначим $R(u; x)$. В теореме 2 мы дадим условие, необходимое и достаточное для оптимальности коэффициентов $A_1, \dots, A_n$.

Формулу (1) назовем наилучшей в E-норме на множестве W^rL_q, если ее узлы $x_1, \dots, x_n$ доставляют наименьшее значение величине

$$R(x) = \|R(\cdot; x)\|, \qquad (2)$$

где $\|\cdot\|$ – норма в некотором пространстве E функций, определенных на $[0, 1]$. Коэффициенты наилучшей формулы вычисляются по построенным узлам так же, как и в первом случае. Мы дадим решение этой задачи, когда $m=0$, $r=1$, $1<q\leqslant\infty$, а норма в (2) есть $L_p(0,1)$ – норма при $1\leqslant p\leqslant\infty$.

Сформулированные выше задачи для различных множеств функций рассматривались, например, в работах [2–5].

I. Рассмотрим задачу построения оптимальных формул (I) с фиксированными узлами с помощью методов, использованных в [6–8] для построения оптимальных квадратурных формул.

Обозначим через Q множество всех формул (I) с конечным значением точной оценки ошибки на множестве W^rL_q, а через S – множество всех кусочно-полиномиальных функций (сплайнов) вида

$$s(t)=\frac{(t-u)_+^{r-m-1}}{(r-m-1)!}-\sum_{k=1}^{n}\frac{A_k(t-x_k)_+^{r-1}}{(r-1)!},$$

удовлетворяющих условиям

$$s^{(j)}(1)=0\quad(j=0,\dots,r-1). \tag{3}$$

Здесь $v_+^k=v^k$ при $v\geqslant 0$, $v_+^k=0$ при $v<0$. Отметим, что функции из S тождественно равны нулю при $t<\min\{x_1,u\}$ и при $t>\max\{x_n,u\}$.

Теорема I. Существует взаимно однозначное соответствие между множествами Q и S причем для любой формулы из Q и функции $f(x)\in W^rL_q$ выполнены соотношения

$$r(f;u;x;A)=(-1)^r\int_0^1 s(t)f^{(r)}(t)\,dt, \tag{4}$$

$$A_k=s^{(r-j-1)}(x_k-0)-s^{(r-j-1)}(x_k+0)\ (k=1,\dots,n), \tag{5}$$

$$r(u;x;A)=\|s\|_{q'}\quad\left(\frac{1}{q}+\frac{1}{q'}=1\right), \tag{6}$$

где $s(t)$ – соответствующая этой формуле функция из S.

Доказательство. Пусть задана формула (I) с конечным значением величины $r(u; x; A)$ на множестве $W^r L_q$. Из [9] следует, что эта формула точна для многочленов степени $\leqslant r-1$. Используя этот факт и представление

$$f(x) = \pi(x) + \frac{1}{(r-1)!}\int_0^1 (x-t)_+^{r-1} f^{(r)}(t)\,dt$$

для $f(x) \in W^r L_q$, где $\pi(x)$ – многочлен степени $\leqslant r-1$, получаем из (I)

$$r(f; u; x; A) = \int_0^1 f^{(r)}(t)\left[\frac{(u-t)_+^{r-m-1}}{(r-m-1)!} - \sum_{k=1}^{n} \frac{A_k (x_k - t)_+^{r-1}}{(r-1)!}\right] dt.$$

Обозначим

$$s(t) = (-1)^r \left[\frac{(u-t)_+^{r-m-1}}{(r-m-1)!} - \sum_{k=1}^{n} \frac{A_k (x_k - t)_+^{r-1}}{(r-1)!}\right].$$

Из равенства $x_+^k + (-1)^k (-x)_+^k = x^k$ и условия точности формулы (I) для многочлена $\pi(x) = (x-t)^{r-1}$ при любом t получаем, что

$$s(t) = \frac{(t-u)_+^{r-m-1}}{(r-m-1)!} - \sum_{k=1}^{n} \frac{A_k (t-x_k)_+^{r-1}}{(r-1)!}, \qquad (7)$$

причем условие (3) для этой формулы выполнено. Следовательно, $s(t) \in S$. Этим равенство (4) доказано. Равенство (5) получается непосредственно из (7).

Обратно, если задана функция $s(t) \in S$, то с помощью интегрирования по частям, как и в [6, 7], можно показать, что этой функции соответствует формула (I) с коэффициентами (5) и ошибкой (4).

Равенство (6) доказывается, как и в [6, 7] построением экстремальной функции или последовательности функций.

Теорема доказана.

Таким образом, построение оптимальной формулы (I) с фиксированными узлами сводится к задаче оптимальной кусочно-полиномиальной аппроксимации функции $(t-u)_+^{r-m-1}$ в метрике $L_{q'}(0,1)$.

Теорема 2. Для того, чтобы при заданных узлах $x_1, \ldots, x_n \in [0,1]$ функция $\bar{s}(t)$ наименее уклонялась от нуля в

метрике $L_{q'}(0,1)^{(1<q\le\infty)}$ среди всех функций из S, необходимо и достаточно, чтобы существовала функция $\bar{M}(t)$, удовлетворяющая условиям

$$\bar{M}^{(r)}(t) = |\bar{s}(t)|^{q'-1}\operatorname{sign}\bar{s}(t),$$

$$\bar{M}(x_k) = 0 \quad (k=1,\dots,n).$$

При этом

$$\int_0^1 |\bar{s}(t)|^{q'}dt = (-1)^r\int_0^1 \bar{M}(t)\,dt.$$

При $1<q_1<\infty$ функция $\bar{s}(t)$ единственна.

Доказательство этой теоремы аналогично доказательству теоремы I из работы [8], поэтому мы его опускаем.

Отметим, что при $q_1=2$ условия теоремы образуют совместно с условиями (3) линейную систему уравнений относительно неизвестных $A_1,\dots,A_n$.

Результаты теорем I и 2 могут быть распространены как на формулы, так и на множества функций более общего вида, рассмотренные в работах [6–8].

2. Рассмотрим задачу нахождения узлов и коэффициентов наилучшей в $L_p(0,1)$ - норме на множестве W^rL_q формулы (I) с $m=0$ при $1<q_1\le\infty$, $1\le p\le\infty$.

Теорема 3. Наилучшая в $L_p(0,1)$ - норме на множестве W^rL_q формула (I) с $m=0$ при $1<q_1\le\infty, 1\le p\le\infty$ имеет узлы

$$x_k=\frac{k-1+\omega}{n-1+2\omega} \quad (k=1,\dots,n), \tag{8}$$

ошибку

$$R(x)=\frac{1}{(\theta+1)^{\frac{1}{p}}}\left(\frac{\omega}{n-1+2\omega}\right)^{\frac{1}{q'}},$$

где

$$\frac{1}{q}+\frac{1}{q'}=1, \quad \theta=\frac{p}{q'} \quad (\theta=\infty \text{ при } p=\infty),$$

$$\omega=(\theta+1)^{\frac{1}{\theta}}\left\|\frac{t(1-t)}{\left\{(1-t)^{q-1}+t^{q-1}\right\}^{\frac{1}{q-1}}}\right\| \quad (\omega=\tfrac{1}{2} \text{ при } q_1=\infty).$$

Наилучшая формула имеет коэффициенты

I) при $1<q_1<\infty$

$$A_1=1, \quad A_2=\dots=A_n=0 \quad \text{при } u\in[0,x_1],$$

$$A_1=\ldots=A_{n-1}=0,\ A_n=1 \quad \text{при} \quad u\in[x_n,1],$$

$$A_1=\ldots=A_{m-1}=A_{m+2}=\ldots=A_n=0,$$

$$A_m=\frac{(x_{m+1}-u)^{q-1}}{(x_{m+1}-u)^{q-1}+(u-x_m)^{q-1}},\quad A_{m+1}=\frac{(u-x_m)^{q-1}}{(x_{m+1}-u)^{q-1}+(u-x_m)^{q-1}}$$

при $u\in[x_m,x_{m+1}]$ $(m=1,\ldots,n-1)$;

2) при $q=\infty$

$$A_1=1,\quad A_2=\ldots=A_n=0 \quad \text{при} \quad u\in[0,\frac{x_1+x_2}{2}),$$

$$A_1=\ldots=A_{n-1}=0,\ A_n=1 \quad \text{при} \quad u\in(\frac{x_{n-1}+x_n}{2},1],$$

$$A_1=\ldots=A_{m-1}=A_{m+1}=\ldots=A_n=0,\quad A_m=1$$

при

$$u\in\left(\frac{x_{m-1}+x_m}{2},\ \frac{x_m+x_{m+1}}{2}\right)\quad (m=1,\ldots,n-1),$$

$$A_1=\ldots=A_{m-1}=A_{m+2}=\ldots=A_n=0,\quad A_m+A_{m+1}=1,\quad 0\le A_m,\ A_{m+1}\le 1$$

при

$$u=\frac{x_m+x_{m+1}}{2}\quad (m=1,\ldots,n-1).$$

Доказательство. I. Найдем коэффициенты оптимальной на $W^r L_q$ формулы (I) с фиксированными узлами $0<x_1<\ldots<x_n<1$ в зависимости от $u\in[0,1]$. При доказательстве можем считать, что $u\neq x_m$ $(m=1,\ldots,n)$. Пусть заданы формула (I), точная для постоянных, с $m=0$ и соответствующая функция $s(t)\in S$, обозначим

$$B_i=\sum_{k=1}^{i}A_k,\quad B_0=0. \tag{8}$$

Из условия точности формулы (I) для постоянных получаем, что $B_n=1$.

При $u\in(x_m,x_{m+1})$ $(m=1,\ldots,n-1)$ имеем

$$s(t)=\begin{cases} 0, & t<x_1,\\ -\sum_{k=1}^{i}A_k=-B_i, & t\in(x_i,x_{i+1})\cap(0,u)\ (i=1,\ldots,m),\\ 1-\sum_{k=1}^{i}A_k=1-B_i, & t\in(x_i,x_{i+1})\cap(u,1)\ (i=m,\ldots,n-1),\end{cases}$$

$$\left| 1 - \sum_{k=1}^{n} A_k = 0 \quad , \quad t > x_n \right. .$$

Следовательно, в этом случае для $\forall s(t) \in S$ **имеем неравенства**

$$\|s\|_{q'}^{q'} = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} |s(t)|^{q'} dt = \sum_{i=1}^{m-1} |B_i|^{q'} (x_{i+1} - x_i) +$$

$$+ |B_m|^{q'} (u - x_m) + |1 - B_m|^{q'} (x_{m+1} - u) + \sum_{i=m+1}^{n-1} |1 - B_i|^{q'} (x_{i+1} - x_i) \geq$$

$$\geq |B_m|^{q'} (u - x_m) + |1 - B_m|^{q'} (x_{m+1} - u), \qquad (9)$$

причем равенство здесь достигается при

$$B_1 = \ldots = B_{m-1} = 0, \quad B_{m+1} = \ldots = B_n = 1. \qquad (10)$$

Правая часть (9) достигает наименьшего значения, равного

$$\begin{cases} \dfrac{(u - x_m)(x_{m+1} - u)}{\left[(x_{m+1} - u)^{q-1} + (u - x_m)^{q-1}\right]^{\frac{1}{q-1}}}, & 1 < q < \infty \\ \min\{u - x_m;\ x_{m+1} - u\}, & q = \infty \end{cases} \qquad (11)$$

при

$$B_m = \begin{cases} \dfrac{(x_{m+1} - u)^{q-1}}{(x_{m+1} - u)^{q-1} + (u - x_m)^{q-1}}, & 1 < q < \infty \\ 1, \quad u \in \left(x_m, \dfrac{x_m + x_{m+1}}{2}\right), & q = \infty \\ 0, \quad u \in \left(\dfrac{x_m + x_{m+1}}{2}, x_{m+1}\right), & q = \infty \end{cases} \qquad (12)$$

(при $q = \infty$ **и** $u = \dfrac{x_m + x_{m+1}}{2}$ $B_m \in [0,1]$ **– любое).**

Аналогично, при $u \in [0, x_1)$ **для** $\forall s(t) \in S$ **имеет место неравенство**

$$\|s\|_{q'}^{q'} = x_1 - u + \sum_{i=1}^{n-1} |1 - B_i|^{q'} (x_{i+1} - x_i) \geq x_1 - u, \qquad (13)$$

причем равенство достигается при

$$B_1 = \ldots = B_{n-1} = 1 \tag{14}$$

а при $u \in (x_n, 1]$ — неравенство

$$\|s\|_{q'}^{q'} = \sum_{i=1}^{n-1} |B_i|^{q'} (x_{i+1} - x_i) + u - x_n \geq u - x_n, \tag{15}$$

причем равенство достигается при

$$B_1 = \ldots = B_{n-1} = 0. \tag{16}$$

Из соотношений (8), (10), (12), (14), (16) получаем значения коэффициентов $A_1, \ldots, A_n$.

2. Найдем теперь при $1 < q_{/} < \infty$ узлы наилучшей в $L_p(0,1)$ норме формулы (1). Из (11), (13), (15) получаем, что

$$R(u;x) = \begin{cases} \dfrac{[(u-x_m)(x_{m+1}-u)]^{\frac{1}{q'}}}{[(x_{m+1}-u)^{q-1} + (u-x_m)^{q-1}]^{\frac{1}{q}}}, & u \in (x_m, x_{m+1}) \ (m = 1, \ldots, n-1), \\ (x_1 - u)^{\frac{1}{q'}}, & u \in [0, x_1) \\ (x_n - u)^{\frac{1}{q'}}, & u \in (x_n, 1]. \end{cases} \tag{17}$$

Поэтому при $1 \leq p < \infty$ имеем

$$\begin{aligned} (R(x))^p &= \int_0^1 |R(u;x)|^p du = \int_0^1 |x_1 - u|^\theta du + \\ &+ \sum_{m=1}^{n-1} \int_{x_m}^{x_{m+1}} \frac{[(u-x_m)(x_{m+1}-u)]^\theta}{[(x_{m+1}-u)^{q-1} + (u-x_m)^{q-1}]^{\frac{\theta}{q-1}}} du + \int_{x_n}^1 |u - x_n|^\theta du = \\ &= \frac{1}{\theta+1} \left\{ x_1^{\theta+1} + \omega^\theta \sum_{m=1}^{n-1} (x_{m+1} - x_m)^{\theta+1} + (1 - x_n)^{\theta+1} \right\}. \end{aligned} \tag{18}$$

Известными методами получаем, что правая часть (18) принимает наименьшее значение, равное

$$\frac{1}{\theta+1} \cdot \left(\frac{\omega}{n-1+2\omega} \right)^\theta$$

при узлах

$$x_k = \frac{k-1+\omega}{n-1+2\omega} \qquad (k = 1, \ldots, n).$$

Аналогично при $p=\infty$ из (I7) получаем равенства

$$\max_{u\in[0,x_1)}|R(u;x)|=x_1^{\frac{1}{q'}},$$

$$\max_{u\in(x_m,x_{m+1})}|R(u;x)|=\tfrac{1}{2}(x_{m+1}-x_m)^{\frac{1}{q'}}\quad(m=1,\dots,n-1),$$

$$\max_{u\in(x_n,1]}|R(u;x)|=(1-x_n)^{\frac{1}{q'}},$$

поэтому

$$\|R(\cdot;x)\|_\infty=\max\Big\{x_1^{\frac{1}{q'}},\tfrac{1}{2}(x_2-x_1)^{\frac{1}{q'}},\dots,\tfrac{1}{2}(x_n-x_{n-1})^{\frac{1}{q'}},$$

$$(1-x_n)^{\frac{1}{q'}}\Big\}\geqslant\Big(\frac{\omega}{n-1+2\omega}\Big)^{\frac{1}{q'}},$$

а равенство здесь достигается при узлах

$$x_k=\frac{k-1+\omega}{n-1+2\omega}\qquad(k=1,\dots,n),$$

где $\omega=2^{-q'}$. Этим для $1<q<\infty$ теорема доказана.

3. Рассмотрим ту же задачу при $q=\infty$. Из (II), (I3), (I5) получаем, что в этом случае

$$R(u;x)=\begin{cases}x_1-u, & u\in[0,x_1)\\ u-x_m, & u\in(x_m,\frac{x_m+x_{m+1}}{2})\\ x_{m+1}-u, & u\in(\frac{x_m+x_{m+1}}{2},x_{m+1})\\ u-x_n, & u\in(x_n,1].\end{cases}\qquad(m=1,\dots,n-1)$$

Поэтому при $1\leqslant p<\infty$ имеем неравенства

$$(R(x))^p=\int_0^1|R(u;x)|^p du=\int_0^{x_1}(x_1-u)^p du+$$

$$+\sum_{m=1}^{n-1}\Big[\int_{x_m}^{\frac{x_m+x_{m+1}}{2}}(u-x_m)^p du+\int_{\frac{x_m+x_{m+1}}{2}}^{x_{m+1}}(x_{m+1}-u)^p du\Big]+\int_{x_n}^1(u-x_n)^p du=$$

$$=\frac{1}{\theta+1}\Big[x_1^{\theta+1}+\frac{1}{2^\theta}\sum_{m=1}^{n-1}(x_{m+1}-x_m)^{\theta+1}+(1-x_n)^{\theta+1}\Big]\geqslant$$

$$\geqslant \frac{1}{\theta+1}\cdot\left(\frac{1}{2n}\right)^{\theta},$$

причем равенство здесь достигается при узлах

$$x_k = \frac{k-\frac{1}{2}}{n} \qquad (k=1,\dots,n). \tag{I9}$$

Аналогично доказывается, что при $p=\infty$ справедливо неравенство

$$R(x) = \sup_{u\in[0,1]} |R(u;x)| \geqslant \frac{1}{2n},$$

которое обращается в равенство при узлах (I9).

Этим теорема полностью доказана.

Отметим, что при $q=\infty$ построенная наилучшая формула (I) выглядит особенно просто. Эта формула имеет узлы (I9), а в качестве приближенного значения $f(u)$ принимается значение функции в ближайшем к u узле x_k.

Литература

I. Бахвалов Н.С. Об оптимальности линейных методов приближения операторов на выпуклых классах функций. Ж. выч. мат. и мат. физ., I97I, II, § 4, с. I0I4–I0I8.

2. Осипенко К.Ю. Оптимальная интерполяция аналитических функций. – Мат.заметки, I972, I2, вып. 4, с. 465–476.

3. Боянов Б.Д. Наилучшие методы интерполирования для некоторых классов дифференцируемых функций. – Мат. заметки, I977, I7, вып. 4, с. 5II–524.

4. Micchelli, C.A., Rivlin, T.J., Winograd, S. The optimal recovery of smooth functions. – Numer. Math., 1976, 26, N. 2, 191–200.

5. Bojanov, B.D., Chernogorov, V.G. An optimal interpolation formula. – J. Appr. Theory, 1977, 20, N. 3, 264–274.

6. Levin, M., Girshovich, V. Optimal quadrature formulae for sets of functions satisfying boundary conditions.-Изв.АН ЭССР,Физ.-Матем.,1975,24,№3,p.264-269.

7. Гиршович Ю.М., Левин М.И. Моносплайны и квадратурные формулы на множествах функций с заданными краевыми условиями. - "Труды Таллинск. политехн. ин-та" , Мат. и теор. мех., 1976, № 393, с. 21-30.

8. Гиршович Ю.М. Оптимальные квадратурные формулы с фиксированными узлами.-Изв. АН ЭССР, Физ.-Мат., 1976, 25, № 2, с. 201-204.

9. Левин М. Одно свойство наилучших квадратурных формул.-Изв. АН ЭССР, Физ.-Мат., 1971, 20, № 1, с. 90-91.

Y. Girshovich

Optimal Recovery and Differentiation of Functions

Summary

The problem of constructing the optimal formulas (1) is considered. Theorem 2 gives necessary and sufficient condition in order that the formula (1) is optimal one for the set $W^r L_q (1<q\leq\infty)$ with prescribed nodes.

Both the nodes and the coefficients of the optimal formula (1) with m=0 in $L_p(0,1)$ - metric for the set $W^1 L_q$ $(1<q\leq\infty;\ 1\leq p\leq\infty)$ are derived in theorem 3.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 439 1978

УДК 51:801

А.Р.Вадер, А.О.Вооглайд

ОПИСАНИЕ МЕТАЯЗЫКА СИСТЕМЫ ПОСТРОЕНИЯ ТРАНСЛЯТОРОВ

Введение

В вычислительном центре Таллинского политехнического института в настоящее время разрабатывается новая версия системы построения трансляторов (СПТ) [13] для ЭВМ ЕС.

Данная система позволяет генерировать транслятор со схемой синтаксически управляемой трансляции, дополненный элементами атрибутных грамматик [2, 8]. Транслятор составляемой СПТ состоит из следующих логических частей:

1) лексический анализатор,

2) синтаксический анализатор,

3) семантический анализатор.

Лексический анализ проводится аналогично [16].

Методом синтаксического анализа служат методы, базирующиеся на грамматиках предшествования с ограниченным контекстом, используемые в технике "каскада" [11].

Результатом синтаксического анализа является дерево синтаксиса [2]. Для облегчения второго прохода сематического анализа можно во время первого прохода построить на синтаксическое дерево дополнительные структуры. Другими словами – можно провести первичный семантический анализ, используя при этом сам процесс создания синтаксического дерева, который является своеобразным алгоритмом прохождения дерева. Точнее первичный семантический анализ представляет собой разбиение вершин дерева на классы эквивалентности, причем элементы одного класса могут быть перечислены или (и) объединены в линейный список.

После генерирования синтаксического дерева обрабатываются создаваемые списки и удаляются далее ненужные поддеревья. Во время обработки списка инициализируются такие семантические атрибуты, значение которых можно назначать при однократном прохождении поддерева. Например, информация переносится от описаний к описываемым объектам.

В течение второго прохождения генерируются связанные вершинами наименования семантических действий. Этот процесс можно связать с активизацией семантических действий, т.е. с генерацией текста базового языка. Базовым языком используется PL/I или ASSEMBLER.

Понятия семантического равенства анализаторов [I2] и мощность допустимых классов грамматик [I4] позволяют соединить семантические действия с правилами метаязыка и не опасаться, что анализатор не отражает исходную структуру синтаксического дерева.

В системе существует возможность замены вида резервированных слов во время работы анализатора таким образом, что семантика сохраняется. Это снимает ограничение, что пользователь не может использовать резервированные слова объектного языка в качестве резервированных слов метаязыка.

При создании данной системы частично использован практический опыт коллег Государственного университета в Хельсинки [I0].

I. Основные понятия

Обозначим через ϑ^* множество всех слов в алфавите ϑ и $\vartheta^+ = \vartheta^* / \{\lambda\}$, где λ – пустое слово. Если $x \in \vartheta^*$, то $|x|$ – длина слова x. Контекстно-свободная грамматика (КСГ) – это упорядоченная четверка $G = (\vartheta_N, \vartheta_T, P, S)$, где ϑ_N – конечный алфавит нетерминальных символов, т.е. множество понятий, ϑ_T – конечный алфавит терминальных символов, т.е. множество элементов объектного языка, $\vartheta_T \cap \vartheta_N = \emptyset$, P – множество правил подстановки вида $\{A \to x : A \in \vartheta_N,\ x \in (\vartheta_N \cup \vartheta_T)^+\}$ и S – начальный символ, т.е. аксиома.

Обозначим $\vartheta = \vartheta_N \cup \vartheta_T$. Пусть $x \in \vartheta^+$ и $y \in \vartheta^*$. Слово y непосредственно канонически выводимо из слова $x (x \Rightarrow y)$, если $x = uAv$, $y = uzv$ и $A \to z \in P$ $(u, x \in \vartheta^*, v \in \vartheta_T^*)$.

Слово y канонически выводимо из слова $x (x \overset{*}{\Rightarrow} y)$, если существует такая последовательность слов $z_0, z_1, \ldots, z_k$ $(z_i \in \vartheta^*, 0 \leqslant i \leqslant k)$., что $x = z_0$, $y = z_k$ и $z_i \Rightarrow z_{i+1} (0 \leqslant i < k)$.

Последовательность $z_0, z_1, \ldots, z_k$ называется каноническим выводом y из x.

Язык $\mathcal{L}$, определяемый грамматикой G, $\mathcal{L}(G)$, есть множество слов, выводимых в G из S и состоит только из терминалов: $\mathcal{L}(G) = \{x : S \overset{*}{\Rightarrow} x;\ x \in \vartheta_T^*\}$.

Пусть $G = (\vartheta_N, \vartheta_T, P, S)$ - КСГ, тогда определим следующие бинарные отношения на множестве $\vartheta \times \vartheta$.

$$\lambda = \{(A,B) : A \to By \in P,\ y \in \vartheta^*\},\ \lambda_T = \lambda \cap (\vartheta \times \vartheta_T),$$
$$\rho = \{(A,B) : B \to xA \in P,\ x \in \vartheta^*\},$$
$$\alpha = \{(A,B) : C \to xABy \in P;\ x, y \in \vartheta^*\},$$
$$\lessdot = \alpha\lambda^+,$$
$$\doteq = \alpha,$$
$$\gtrdot = (\rho^+ \alpha \lambda_T^*).$$

Отношения $\lessdot, \doteq, \gtrdot$ называются отношениями предшествования.

КГ-грамматика G называется грамматикой предшествования, если

1) P не содержит правил с пустой правой частью (КСГ G λ - свободная);

2) отношения $\lessdot, \doteq, \gtrdot$ попарно непересекающиеся [9].

Имеет место следующая теорема (см. [1]). Если G - грамматика предшествования и

$$S \overset{*}{\Rightarrow} X_1 X_2 \ldots X_k A T_1 \ldots T_p \Rightarrow X_1 X_2 \ldots X_k Y_1 \ldots Y_m T_1 \ldots T_p \text{ в } G,$$

то имеют место следующие отношения предшествования и только они:

1) $X_i \lessdot X_{i+1} \vee X_i \doteq X_{i+1}$ $1 \leqslant i < k$,

2) $X_k \lessdot Y_1$,

3) $Y_j \doteq Y_{j+1}$ $1 \leqslant j \leqslant m$,

4) $Y_m \Rightarrow T_1$.

Канонический вывод $z_0, z_1, \ldots, z_k$ можно представить в виде $p_1, p_2, \ldots, p_k$, где $p_i \in P$ – правило, при помощи которого z_i непосредственно выводимо из z_{i-1}. Пусть правило p_i имеет вид $A_i \to x_i$. Вводим сокращенное обозначение $(A_i \to x)_{i=1}^{k}$ для вывода $p_1, \ldots, p_k$. Пусть $H \subseteq P$ – множество таких правил, которые имеют семантическое значение. Пусть $D = (A_i \to x_i)_{i=1}^{k}$ – канонический вывод грамматики $G = (\vartheta_N, \vartheta_T, P, S)$, H – разреженным выводом называется кортеж $D_H = (A_i \to x_i : A_i \to x_i \in H)_{i=1}^{k}$ [4, с. 681]. Пусть T_D – дерево вывода, которое соответствует выводу D [4, с. 682]. Пусть $Q \subseteq \vartheta_T$ – такое подмножество ϑ_T, элементы которого имеют семантическое значение, т.е. ϑ_T / Q содержит такие терминальные символы, содержание которых выражено в структуре дерева вывода (скобки, разделители) или содержание которых совпадает с семантикой соответствующих правил (символы операции, ключевые слова). H,Q – разреженным деревом вывода D (обозначаем $T_D^{H,Q}$) называем дерево, которое получается из дерева вывода T_D следующим образом:

1) вершинами дерева $T_D^{H,Q}$ являются все вершины дерева T_D, метки которых принадлежат множеству $H \cup Q$;

2) в дереве $T_D^{H,Q}$ соединены X и Y дугой тогда и только тогда, когда в дереве T_D существует путь $z_0, \ldots, z_l$ от вершины X до вершины Y такой, что $z_i \in P/H$. $(1 \leqslant i \leqslant l-1$, $z_0 = X, z_l = Y)$.

2. Описание метаязыка

Язык описания составляемого метаязыка. Ни один язык невозможно описать его собственными средствами. Для этой цели используется метаязык [16]. В связи с тем, что в данном случае объектным языком (описываемым языком) служит метаязык, в целях выделения и конкретизации его следует писать с большой буквы.

С целью более подробного описания Метаязыка используется метаязык и обозначения Джексона [5]. Так, метаязык составляемого языка состоит из слов русского языка, эле-

ментов Метаязыка (см. с. 63–65) и символов {}, [ ], ::=, & и |, которые называются знаками метаязыка.

Синтаксической единицей метаязыка называется произвольное сочетание из элементов метаязыка, которое записывают в фигурных или квадратных скобках.

Символ ::= используется для обозначения "допустим"; символом & обозначается "и"; символом | обозначается "или" и при помощи него записываются альтернативные варианты определяемого объекта; квадратные скобки [ ] соответствуют необязательной синтаксической единице или же нескольким единицам; фигурные скобки { } используются для составления синтаксических единиц.

В квадратных и фигурных скобках может существовать и "многоэтажная" коллекция.

Если такая коллекция в квадратных скобках, то это обозначает, что выбирается один из "этажей" или опускается вся конструкция. Например, конструкции $[\begin{smallmatrix}A\\B\end{smallmatrix}]$ и [A|B] эквивалентные и обозначают, что на данном месте должен быть или символ A или B, или это место остается пустым.

Если "многоэтажная" коллекция в фигурных скобках, то это обозначает выбор одного "этажа". Например, конструкции $\{\begin{smallmatrix}A\\B\end{smallmatrix}\}$ и {A|B} эквивалентные и на данном месте должен быть или символ A или B.

Использование обозначения Джексона для описания составляемого метаязыка. При помощи обозначения Джексона можно определить правила синтаксиса метаязыка, используя

1) элементарную операцию,

2) последовательность,

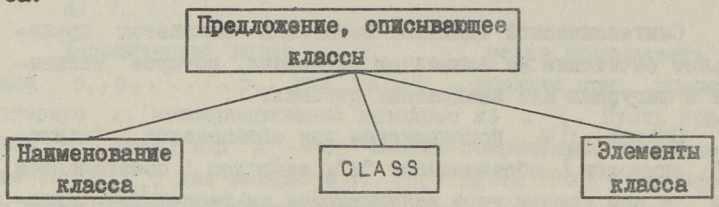
3) выборку,

4) повторение.

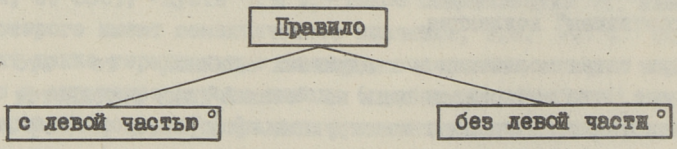
Обозначение элементарной операции ☐ используется для обозначения обязательного элемента в программе Метаязыка. Например, SIMPLE указывает на то, что в данном месте программы Метаязыка должно быть ключевое слово SIMPLE.

Последовательность означает понятие "и". Например, предложение, которое определяет классы, состоит из наимено-

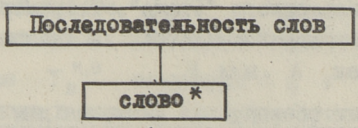
вания класса, ключевого слова CLASS и элементов класса:



Выборка обозначает понятие "или", для обозначения которого используется нолик в верхнем правом углу блока: [ ]° . Например, правило в Метаязыке может быть с правой частью или без нее:



Повторение обозначается " * ". Пусть дано повторение и повторяющимся элементом является слово:

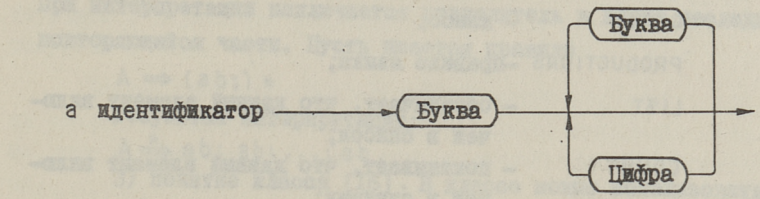


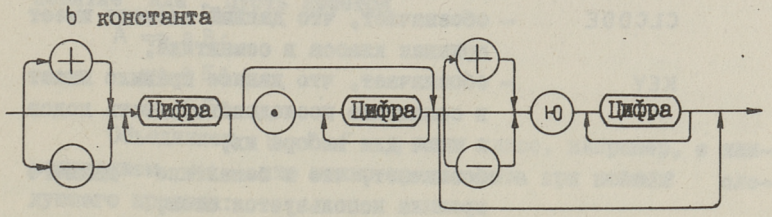
Необходимо учитывать требование, что элементы одного уровня последовательности должны иметь обозначения однотипного характера.

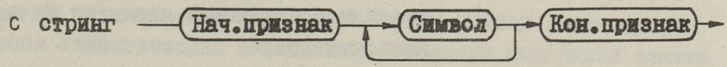
Описание синтаксиса. Описание лексического анализатора. Здесь используется лексический анализатор с фиксированными лексемами.

Опираясь на опыт вычислительного центра Эстонского радио, мы отказались от автоматически генерируемого лексического анализатора [6]. Причиной отказа является цель уменьшить количество возможных ошибок со стороны пользователя. В данном случае пользователь не должен описывать весь конечный автомат – лексический анализатор.

Лексемы зафиксированы следующим образом:







Если пользователю нужны специальные лексемы, то фиксированные можно заменить на уровне модулей.

Элементы Метаязыка. Элементами Метаязыка являются следующие терминалы и понятия.

Терминалы:

I) I - идентификатор Метаязыка, который обозначает резервированные слова в объектном языке и нетерминальные элементы в правилах Метаязыка относительно объектного языка;

2) C - постоянная Метаязыка, которая обозначает числовые постоянные в объектном языке и в правилах семантики Метаязыка;

3) ключевые слова, которыми являются идентификаторы, имеющие в языке специальное значение. В Метаязыке используются следующие ключевые слова:

GRAMMAR - грамматика;

TERMINALS - резервированные слова;

SIMPLE - простой разделитель;

CONNECT - составной разделитель;

SUBGRAMMAR – подграмматики;

GLASS – класс;

PRODUCTIONS – правило языка;

LIST – обозначает, что данный элемент включен в список;

COUNT – показывает, что данный элемент включен в счетчик;

CLCODE – обозначает, что данный элемент имеет признак класса в семантике;

KEY – обозначает, что данное правило имеет в семантике последовательность кодов и ключ для выбора их;

STACK – обозначает, что в семантике данного правила используется стек;

STARTSY – обозначает, что левая часть данного правила подстановки является начальным символом;

$\bar{I}$ – идентификатор объектного языка;

$\bar{C}$ – числовая константа объектного языка;

$\bar{S}$ – стринг объектного языка;

PREORDER – обозначает алгоритм прохождения поддерева в последовательности PREORDER [7].

Понятия:

1) символ " $\Rightarrow$ " отделяет в формуле Бэкуса левую часть от правой;

2) сочетание символов $(\ )*$, которое интерпретируется в Метаязыке как звездочка Клини [17], что означает повторение.

Пусть имеется правило

$A \Rightarrow a(b+c)*$,

тогда сочетание $(b+c)*$ интерпретируется как повторение, т.е. $b+c$ в программе может быть любое число раз, но непременно хотя бы один раз.

Если необходимо разделить повторяющееся сочетание разделителем, то следует сделать следующее:

$(\{$ повторяющаяся часть $\}\ E)*$,

где E – разделитель.

При интерпретации исключается разделитель в конце последней повторяющейся части. Пусть имеется правило

$A \Rightarrow (ab;)*$

Результат интерпретации:

$A \overset{*}{\Rightarrow} ab; ab; \ldots; ab$.

3) понятие класса [I6]. В классе можно использовать понятие "или". Пусть имеется

$A \Rightarrow aBc$

$\Rightarrow aEc$

$\Rightarrow aFc$

Объединяем B, E, F в один класс, например, с наименованием KL, что можно представить при помощи следующего правила:

$A \Rightarrow aKLc$.

В случае, если класс описан, при помощи наименования класса семантический анализатор Метаязыка совершает замены автоматически;

4) разделитель конца физической записи для того, чтобы отделить одно правило от другого.

Семантика. Каждый элемент или правило объектного языка может иметь семантику. Семантика определяет множества H, Q и синтаксическое дерево $T_D^{H,Q}$. Если терминал или правило имеет семантику, то они автоматически включаются в синтаксическое дерево $T_D^{H,Q}$. Семантика может иметь следующие формы:

1) Сем 1 :: = код 1 & дс;

2) Сем 2 :: = CLCODE & к & код1 & дс;

3) Сем 3 :: = код0 & дс;

4) Сем 4 :: = STARTSY & код0 & дс;

где

$$код0 ::= \left\{ \begin{array}{l} код \\ \\ код3 \end{array} \right\}$$

$$код ::= \left\{ \begin{array}{l} код2 \,\&\, код1 \\ \\ ко \end{array} \right\} \qquad код2 ::= \left\{ \begin{array}{l} ST \\ \\ PR \\ \\ ST \,\&\, PR \end{array} \right\}$$

$$\text{КО} ::= \left\{ \begin{matrix} \text{код 1} \\ \\ \text{I} \end{matrix} \right\}$$

код 1 ::= C

ST ::= STACK & тип стека

тип стека ::= C

PR ::= PREORDER

(по умолчанию выбирают последовательность, аналогичную PREORDER, но корень поддерева проходят два раза).

$$\text{АС} ::= \left[\begin{matrix} \{\text{LIST} \,\&\, n_1\} \mid \{\text{COUNT} \,\&\, n_2\} \\ \{\text{LIST} \,\&\, n_1\} \,\&\, \{\text{COUNT} \,\&\, n_2\} \end{matrix} \right]$$

n_1 ::= номер списка,

n_2 ::= номер счетчика,

k ::= признак класса.

Признак класса ::= 0 | I | 2 | ... | 3 I .

CLCODE, STARTSY, STACK, LIST, COUNT, PREORDER – ключевые слова,

I, C – элементы Метаязыка,

код 1 – простой код, т.е. наименование семантической программы,

код 2 – ведущий код, т.е. наименование семантической программы, которое является составным именем, полученным из последовательности меток вершин с кодом код 2 и кода данной вершины.

Код 2 определяет алгоритм прохождения поддерева. Стек можно использовать, например, для обработки вложенных условных операторов.

код 3 ::= KEY & арифметическое выражение & пк,

пк ::= пк & код,

пк ::= код,

где пк – последовательность кодов.

Арифметическим выражением является стандартное арифметическое выражение, результат вычисления которого преобразуется в целое число.

Данный результат вычисления выражения назначит код из последовательности кодов. Если результат выражения больше, чем количество кодов, то обработка вершины пропускается.

Элементами арифметического выражения могут быть:

I) глобальные переменные, которые имеют значения или которым они присваиваются во время первичного семантического анализа,

2) коды из ближайшего окружения данной вершины. Ближайшим окружением считается сама вершина, главная и подчиненные вершины на расстоянии I.

Семантика разделителей. Составные разделители могут иметь семантику в виде СемI.

Простые разделители могут иметь семантику в видах СемI или Сем2.

Если простой разделитель имеет семантику Сем2, то входящий в неё признак класса интерпретируется следующим образом:

I - разделитель, который игнорируется;
2 - простой разделитель;
3 - разделитель, с которого начинается хотя бы один составной разделитель;
4 - разделитель, используемый при анализе ошибок;
5 - разделитель стринга;
6 - разделитель комментария;
7 - разделитель строки;
8 - разделитель уровня;
9 - разделитель препроцессора.

Если значение признака класса принадлежит отрезку [I0,3I], то разделитель является простым и используется попарно (например, скобки), или этот разделитель связан с понятием конкретного языка.

Семантика резервированных слов. Резервированные слова могут иметь семантику в видах СемI и Сем2.

Если резервированное слово имеет семантику Сем2, то оно является словом, которое не имеет значения при определении языка, но делает язык более обозримым. Такое слово пропускается при лексическом анализе.

Семантика правил. С каждым правилом можно связывать семантику, которая переносится на ту продукцию КС-грамматики, левой частью которой является соответствующий левой части правила нетерминал. Для сокращения времени анализа вычисляется соответственно семантике ключ, который назначает генерацию синтаксического дерева $T_D^{H,Q}$ и первичный семантический анализ.

Правило может иметь семантику в видах Сем3 или Сем4. Вершине может соответствовать имя семантической подпрограммы. Имя дается само по себе или сгенерированным из последовательности кодов типа 2 и типа I или вычисленным из ближайшего окружения и глобальных параметров. В конкретных случаях при вычислении с вершиной можно связать имя пустого действия.

Если правило имеет в семантике ключевое слово STARTSY, то прекращается составление синтаксического дерева программы, т.е. левая часть правила считается начальным символом. Следовательно, начальным символом может быть несколько разных понятий.

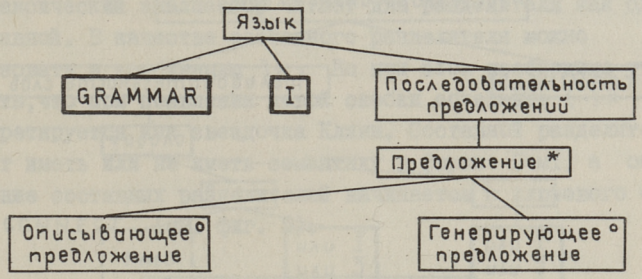
3. Правила описания объектного языка

Для синтаксического анализа объектного языка необходимо определить его символы и правила при помощи Метаязыка. Правила определения приведены на фиг. I – I4. Элементы, выделенные жирным шрифтом, подставляются в синтаксическое дерево $T_D^{H,Q}$. Также эти элементы указывают пользователю на необходимые при составлении программ в Метаязыке элементы.

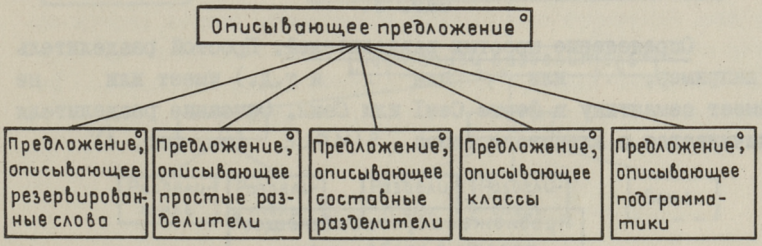
Язык состоит из ключевого слова GRAMMAR, наименования грамматики и последовательности предложений. Предложения могут быть описывающими или генерирующими и могут располагаться в программе в любом порядке (см. фиг. I).

При помощи определяющих предложений можно описать резервированные слова, простые и составные разделители, классы

и подграмматики (см. фиг. 2)



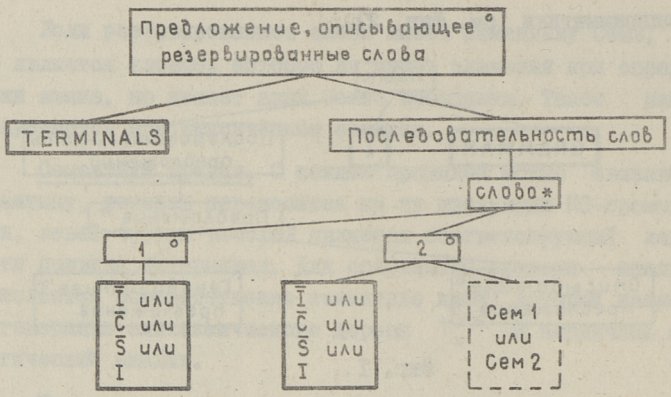
Фиг. I.



Фиг. 2.

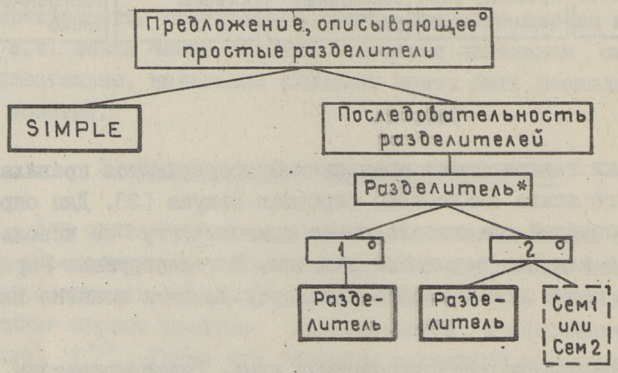
При помощи генерирующих предложений составляются правила объектного языка аналогично формулам Бэкуса [3]. Для определения понятий объектного языка символы < > не используются, так как все терминалы описаны. В генерирующих предложениях можно использовать звездочку Клини и понятие класса.

Определение резервированных слов. Резервированным словом являются идентификатор объектного языка ($\overline{I}$), постоянная объектного языка ($\overline{C}$), стринг объектного языка ($\overline{S}$) или любое слово объектного языка, длина которого может быть произвольной, для опознавания же используются только 8 символов. Резервированные слова могут быть без семантики или иметь форму Сем1 или Сем2. Описание резервированных слов начинается с ключевого слова TERMINALS (см. фиг. 3).



Фиг. 3.

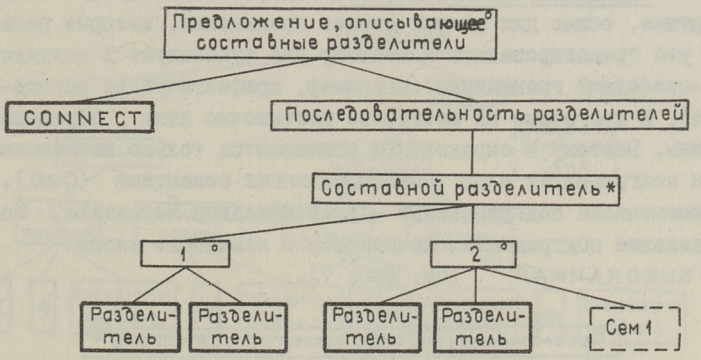
Определение простых разделителей. Простой разделитель (например, "." или "," или ":" и т.д.) имеет или не имеет семантику в форме Сем1 или Сем2. Описание разделителя начинается с ключевого слова SIMPLE (см. фиг. 4).



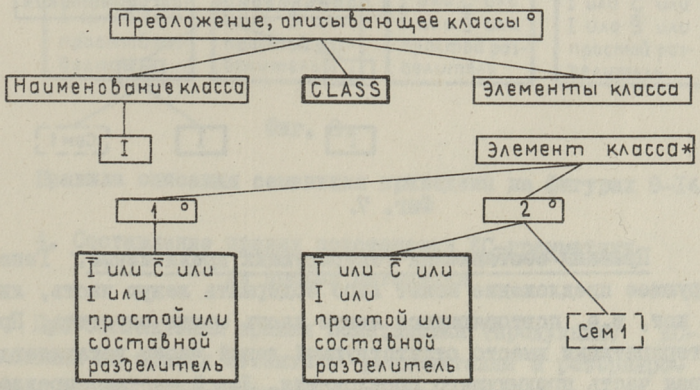
Фиг. 4.

Определение составных разделителей. Для описания языка достаточно кодированных простых разделителей и попарных сочетаний этих разделителей. Поэтому составной разделитель состоит из двух простых, при этом не исключена возможность использования их отдельно. Если компоненты составного разделителя описаны как простые, то пользователь должен учесть,

что если в одном правиле эти разделители находятся рядом, то лексический анализатор читает два разделителя как один составной. В качестве составного разделителя можно использовать и комбинацию)* . Но при этом необходимо учитывать, что при появлении левой скобки сочетание ( )* интерпретируется как звездочка Клини. Составной разделитель может иметь или не иметь семантику в форме СемI, а определение составных разделителей начинается с ключевого слова CONNECT (см. фиг. 5).



Фиг. 5.

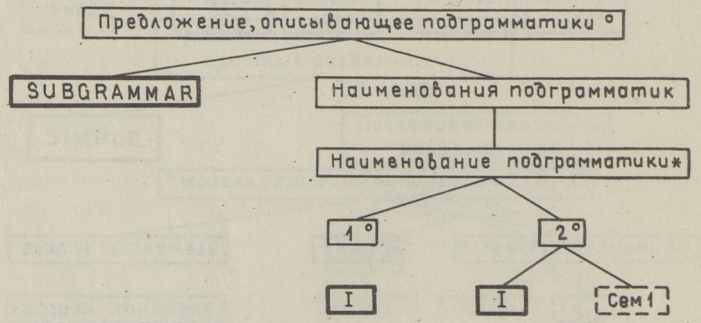


Фиг. 6.

Определение классов. Определение класса состоит из наименования класса, ключевого слова CLASS и из элементов класса. Наименованием класса может быть любой идентификатор Метаязыка.

Элементом класса может быть идентификатор или постоянная объектного языка, идентификатор Метаязыка или любой разделитель. Количество элементов не ограничено, они могут иметь семантику в форме СемI (см. фиг. 6).

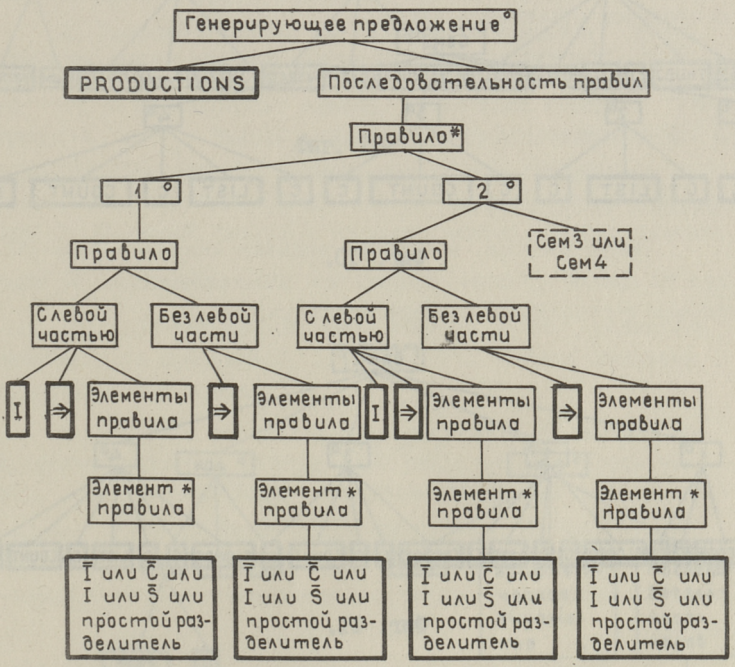
Определение подграмматик. Подграмматики [I5] – стандартные, общие для многих языков грамматики, которые раньше уже транслировались и поэтому уже существуют в контекстно-свободной грамматике (например, арифметические выражения). В программе на Метаязыке достаточно лишь обращения к ним. Поэтому в определении описываются только наименования подграмматик и при необходимости их семантика (СемI). Наименование подграмматики – идентификатор Метаязыка, определение подграмматик начинается с ключевого слова SUBGRAMMAR (см. фиг. 7).



Фиг. 7.

Правила составления генерирующих предложений. Генерирующее предложение может либо содержать левую часть, либо нет, т.е. повторяющуюся левую часть можно опустить. При интерпретации вместо отсутствующей левой части вставляется левая часть предыдущего предложения. Левой частью предложения может быть ровно один нетерминальный элемент, который может обозначать наименование класса. Правая часть пред-

ложения может содержать любое количество терминальных и нетерминальных элементов. Допускается использование классов, подграмматик, звездочки Клини. Недопустимо использование постоянных Метаязыка. Определение предложения начинается с ключевого слова PRODUCTIONS и может содержать семантику вида Сем3 или Сем4 (см. фиг. 8).

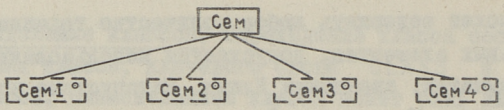


Фиг. 8.

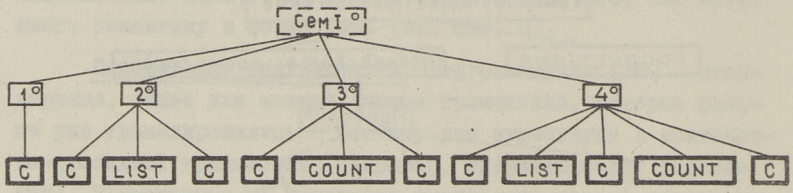
Правила описания семантики приведены на фигурах 9–14.

4. Составление правил подстановки КС-грамматики из правил метаязыка

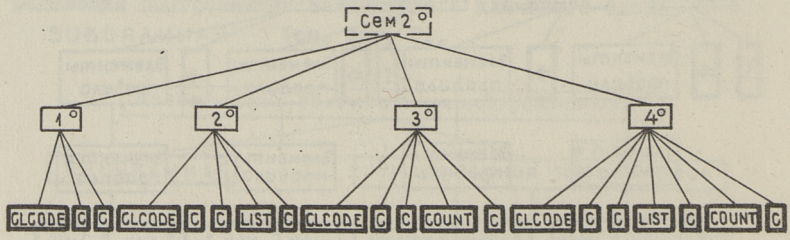
При составлении правил подстановки (продукции) КС-грамматики из правил Метаязыка разделителям и резервированным словам присваиваются автоматически внутренние коды, которые учитываются при синтаксическом анализе, если пользователь не назначит специальный код. Каждому понятию



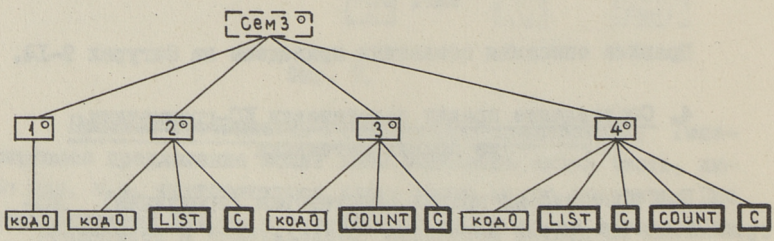
Фиг. 9.



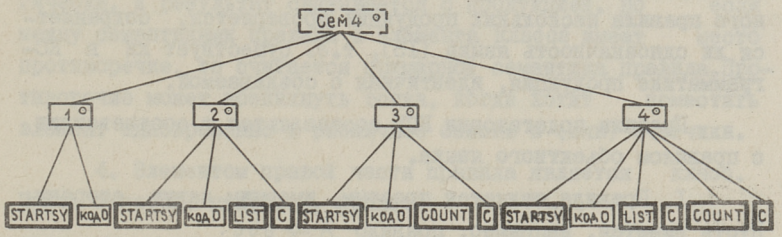
Фиг. IO.



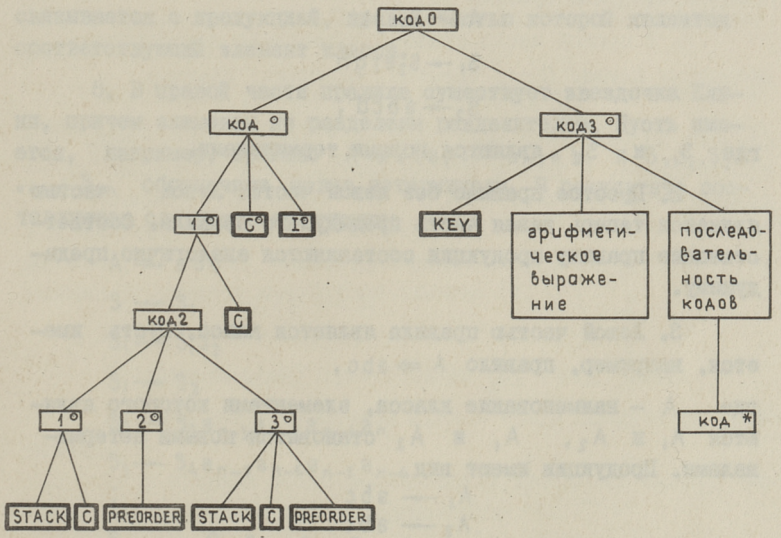
Фиг. II.



Фиг. I2.


Сем4°
1°
2°
3°
4°
STARTSY
КОД0
STARTSY
КОД0
LIST
C
STARTSY
КОД0
COUNT
C
STARTSY
КОД0
LIST
C
COUNT
C

Фиг. I3.


КОД0
КОД°
КОД3°
1°
C°
I°
KEY
арифметическое выражение
последовательность кодов
КОД2
C
1°
2°
3°
КОД *
STACK
C
PREORDER
STACK
C
PREORDER

Фиг. I4.

ставится в соответствие нетерминал и пользуются новыми нетерминалами в случае, если одному правилу Метаязыка соответствует несколько продукций КСГ. При составлении из одного правила нескольких продукций проверяется, сохраняется ли однозначность языка [I5], т.е. существует ли в КС-грамматике продукция, идентичная с создаваемой.

Правила подстановки КСГ составляются в соответствии с правилом объектного языка.

I. Правило является простым, имеющим левую сторону. Пусть имеется, например, правило $A \Rightarrow abc$.

В КСГ составляется продукция $A \rightarrow abc$.

Код левой стороны сохраняется до следующего правила, имеющего левую сторону.

Пусть имеется, например, правило $A \Rightarrow abcdefghijkl$, где на правой стороне больше, чем пять элементов. В таком случае составляются следующие продукции:

$$A \rightarrow S_1 ijkl$$

$$S_1 \rightarrow S_2 efgh$$

$$S_2 \rightarrow abcd,$$

где S_1 и S_2 являются новыми терминалами.

2. Простое правило без левой части. Левой частью является теперь левая часть предыдущего правила. Соответственные правилу продукции составляются аналогично предыдущему.

3. Левой частью правила является класс. Пусть имеется, например, правило $A \Rightarrow abc$,

где A - наименование класса, элементами которого являются A_1 и A_2, A_1 и A_2 становятся новыми нетерминалами. Продукции имеют вид

$$A_1 \rightarrow abc$$
$$A_2 \rightarrow abc.$$

Если элемент класса не имеет семантики, а правило имеет, то с продукцией, соответствующей данному элементу класса, связывается семантика правила. Если правило не имеет семантики, а элемент класса имеет, то с продукцией связывается семантика элемента класса. Если и правило и

элемент класса имеют семантику, то семантики слагаются логически, и результат связывается с продукцией. Но если между семантиками правила и элемента класса имеет место противоречие, то считается примарной семантика правила. Противоречие может возникнуть тогда, когда хотят поместить элемент одновременно в различные списки и (или) счетчики.

4. Элементом правой части правила является класс. Пусть имеется, например, правило $A \Rightarrow aKLb$,

где KL – наименование класса, элементами которого являются A_1 и A_2. Вместо наименования класса пользуются новым нетерминалом S. Первой продукцией является $A \rightarrow aSb$. Затем составляются продукции, левой частью которых является S и правой частью – соответствующий элемент класса:

$$S \rightarrow A_1 \quad \text{и} \quad S \rightarrow A_2.$$

Если элемент класса имеет семантику, то семантика связывается с продукцией, правой частью которой является соответствующий элемент класса.

5. В правой части правила существует звездочка Клини, причем элементы не разделены разделителем. Пусть имеется, например, правило $A \Rightarrow a(a_1 \dots a_n)*b$ и $S, S_1, S_2, \dots, S_m$ обозначают новые нетерминалы. В результате составляются следующие продукции:

$$A \rightarrow aSb$$
$$S \rightarrow S_1$$
$$S_1 \rightarrow S_1 S_2$$
$$S_1 \rightarrow S_2$$
$$S_2 \rightarrow S_3 a_{n-3} a_{n-2} a_{n-1} a_n$$
$$S_3 \rightarrow S_4 a_{n-7} a_{n-6} a_{n-5} a_{n-4}$$
$$\vdots$$
$$S_{m-1} \rightarrow S_m a_{i+1} a_{i+2} a_{i+3} a_{i+4}$$
$$S_m \rightarrow a_1 \dots a_i,$$

где $i = \begin{cases} \text{modulo } [\frac{n}{4}], & i \neq 0 \\ 4 \end{cases}$

С целью предотвращения конфликтов предшествования произведена проверка и никаких конфликтов не существует.

6. В правой части правила существует звездочка Клини, причем повторяющиеся элементы разделены разделителем. Пусть имеется, например, правило $A \Rightarrow a(a_1 \dots a_n E)*b$,

где E – обозначает разделитель. В результате составляются следующие продукции:

$$A \rightarrow aSb$$
$$S \rightarrow S_1$$
$$S_1 \rightarrow S_2 S_3$$
$$S_2 \rightarrow S_1 E$$
$$S_1 \rightarrow S_3$$
$$S_3 \rightarrow S_4 a_{n-3} a_{n-2} a_{n-1} a_n$$
$$S_4 \rightarrow S_5 a_{n-7} a_{n-6} a_{n-5} a_{n-4}$$
$$\vdots$$

Противоречия с требованиями грамматики предшествования не возникает. Проверка производится аналогично описанной в предыдущем пункте.

7. В правой части правила существует многократная звездочка Клини. Пусть имеется, например, правило

$$A \Rightarrow a(b(c_1 \dots c_n E) * E_1) * a.$$

Сначала составляются продукции внутреннего повторения, потом внешнего. В данном случае составляются следующие продукции:

$$S \rightarrow S_1 \qquad S_5 \rightarrow S_7$$
$$S_1 \rightarrow S_2 S_3 \qquad S_7 \rightarrow bS$$
$$S_2 \rightarrow S_1 E \qquad A \rightarrow aS_4 a$$
$$S_1 \rightarrow S_3$$
$$S_3 \rightarrow c_1 \dots c_n$$
$$S_4 \rightarrow S_5$$
$$S_5 \rightarrow S_6 S_7$$
$$S_6 \rightarrow S_5 E_1$$

Проверено, что конфликтов предшествования не возникает.

8. В правой части правила в повторяющемся элементе существует класс. Пусть имеется, например, правило

$$A \Rightarrow d(aKLb) * d;$$

где элементами класса КС являются A_I и A_2. Сначала составляются продукции для понятия класса, затем для повторения. В данном случае составляются следующие продукции:

$S \rightarrow A_1$ $\quad$ $S_2 \rightarrow S_3$

$S \rightarrow A_2$ $\quad$ $S_3 \rightarrow aSb$

$S_1 \rightarrow S_2$ $\quad$ $A \rightarrow dS_1d$

$S_2 \rightarrow S_2S_3$

Противоречия с требованиями грамматики предшествования не возникает.

9. В правой части правила существует наименование подграмматики, которой является КСГ, полученная из Мета-анализатора. Для того, чтобы внутренние коды данной грамматики (обозначим это через SG) и подграмматики (обозначим это через AG) были идентичными, производится переименование кодов AG, соответственно кодам SG. Терминалу грамматики AG присваивается код идентичного с ним терминала в грамматике SG. При нетерминалах проводится переименование кодов.

Внутренний код наименования грамматики AG в грамматике SG соответствует тому нетерминалу, который является начальным символом подграмматики.

Правила грамматики AG и их семантика переносятся в грамматику SG, причем с правилом, левой частью которого является начальный символ подграмматики, не связывается в семантике признак начального символа (STARTSY).

В итоге можно сказать что:

I) правило может существовать без левой части;

2) количество элементов правой части правила не ограничено;

3) в правиле можно использовать звездочку Клини, понятие класса и стандартные грамматики;

4) левой частью правила может быть наименование класса;

5) повторение может существовать внутри повторения;

6) внутри повторения можно использовать понятие класса;

7) наименования подграмматики можно использовать в любом месте в правой части правила.

Литература

1. A h o, A.V., D e n n i n g, P.I., U l l m a n, J.D. Weak and mixed strategy precedence parsing. - J.ACM, 1972, 19,2,p.226...243.

2. A h o, A.V., U l l m a n, J.D. The theory of parsing. Translation and Compiling. II: Compiling. USA, New Jersey, Prentice - Hall, 1973.

3. F l o y d, R.W. Syntactic analysis and operator precedence,-JACM, 10 (July 1963), p. 316-327.

4. G r a y, J.N., H a r r i s o n, M.A. On the covering and reduction problems for context-free grammars. - J.Assoc.Comput.Machinery, 1972, 19,4,p.675...698.

5. J a c k s o n, M.A. Principles of program design. London, A.P.I.C Studies in Data Processing, 1975.

6. J o n s o n, W.L., P o r t e r, J.H., A c k l e y, S.I., R o s s, D.T. Automatic generation of efficient lexical processors using finite state techniques. CASM,11.Dec. 1968.

7. K n u t h, D.E. The art of computer programming, vol.1. Addison Wesley, 1968.

8. L e w i s, P.M., R o s e n k r a n t z, D.J., S t e a r n s, R.E., - Attributed Translations. Journal of Computer and System Sciences,9,4 (Dec. 1974), p.279...307.

9. W i r t h, N., W e b e r, H. EULER - a generalization of ALGOL and its formal definition,pt.1. - Comm. ACM, 1966, 9,No.1.,p.13...23.

10. R ä i h ä, K.-J. Suunnitelma matakääntäjän semanttiseksi osaksi. Sarja C. Arkistonumero 1976/24 tekninen raportti. Helsingin yliopisto.

II. В о о г л а й д А.О., Т о м б а к М.О. О проблемах редуцирования в грамматиках предшествования. – "Тр. Таллинск. политехн. ин-та", 1975, № 386, с. 23–37.

I2. В о о г л а й д А.О. Семантическое равенство распознавателей, работающих на грамматике LR(k) и грамматике предшествования C(I/I) ограниченным каноническим контекстом. – "Тр.Таллинск. политехн. ин-та", 1976, № 4II, с.39–55.

I3. В о о г л а й д А.О., Т о м б а к М.О. Система построения эффективных многопроходных трансляторов с LR(k) семантикой. – "Программирование", 1976, № 5, с. 28–38.

I4. В о о г л а й д А.О. Расширенный анализатор предшествования со смешенной стратегией. – "Тр. Таллинск. политехн. ин-та", 1977, № 423, с. 23–4I.

I5. Г и н з б у р г С. Математическая теория контекстно-свободных языков. М., "Мир", 1970.

I6. Г р и с Д. Конструирование компиляторов для цифровых вычислительных машин. М., "Мир", 1975.

I7. Э р л и Д. Эффективный алгоритм анализа контекстно-свободных языков. БКС. Языки и автоматы. М.,"Мир", 1975.

A. Vader, A. Vooglaid

Compiler Writing System's Metalanguage for ES Computers

Summary

The description of metalanguage used in compiler writing system for ES computers is presented. The rules for generating context-free grammar from the object language description are given.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 439 1978

УДК 51:801

А.О.Вооглайд, Х.Х.Рохтла

ОРГАНИЗАЦИЯ ОБРАБОТКИ ОШИБОК В СИСТЕМЕ ПОСТРОЕНИЯ ТРАНСЛЯТОРОВ

Введение

В вычислительном центре Таллинского политехнического института создан новый вариант системы построения трансляторов (СПТ) для ЭВМ ЕС, являющийся логическим продолжением предыдущей СПТ [15]. Данная статья написана прежде всего по заказу пользователей СПТ. Дело в том, что опыт нескольких лет работы изложен в статье [13] в виде ряда требований пользователей СПТ. Большинство из этих вопросов уже решены и реализованы в новой СПТ. Например, проблемы стандартизации и независимости семантических подпрограмм, возможности расширения контекстно-свободной грамматики при помощи "звездочки" Клини и использования более мощного механизма разбиения на классы эквивалентности для первичного семантического анализа и подграмматик в метаязыке([12, 13, 15] и статья на странице 57 в настоящем сборнике).

Среди названных в статье [13] проблем есть еще две незатронутые нами, как правило, возникавшие в виде вопросов со стороны пользователей на рабочих семинарах новой СПТ. Первая из них затрагивает проблемы синтаксических ошибок. Интересно отметить, что как в большинстве ранее известных СПТ, так и в нашей предыдущей системе, данная проблема считалась второстепенной. С точки зрения же пользователя она является первостепенной, особенно, если учитывать принцип "безопасного программирования" Тейтельмана [11].

В данной статье дается общее представление о новом комплексном методе диагностики и исправления синтаксических ошибок. Учитывая объем алгоритмов и необходимость получения статистических материалов из имеющегося программного обеспечения, подробное изложение описания алгоритмов приводится в отдельной статье. Статистический материал необходим для рекомендаций пользователю по выбору мощности диагностического аппарата и определения цены исправления синтаксических ошибок в зависимости от создаваемого языка. Второй вопрос, волнующий пользователя, это отсутствие обзора общения с системой. Самое короткое и простое решение заключается в описании обработки ошибок в ходе генерации транслятора и в рассмотрении требований к обработке ошибок в трансляторе.

В данной статье и дается ответ на два вышеуказанных, ранее не затрагиваемых вопроса. Основные понятия, используемые в статье, можно найти в [I4].

I. Обработка ошибок во время генерации транслятора

Ошибки, появляющиеся при работе с СПТ, можно разделить на два класса:

I) ошибки в генерации транслятора,

2) ошибки в работе транслятора.

Рассмотрим, во-первых, ошибки первого класса.

Генерирование транслятора начинается с работы метатранслятора. Пользователь описает синтаксис и семантику декларируемого языка в метаязыке, метатранслятор переведет это описание в КС-грамматику и свяжет семантику с правилами этой грамматики. При создании метатранслятора использовали метод бутстрапинга, т.е. метатранслятор был создан при помощи той самой СПТ. Обработка ошибок во время метатрансляции происходит так же, как и во время работы создаваемых трансляторов. Это будет рассмотрено ниже.

Второй шаг генерации транслятора – это преобразование данной грамматики в грамматику предшествования. Доказано, что любая грамматика G может быть преобразована в грамма-

тику предшествования G' так, что $\mathcal{L}(G')=\mathcal{L}(G)-\{\lambda\}$ [8, I6].

С точки зрения практики наиболее эффективны два метода такого преобразования (т.е. устранения конфликтов предшествования), данные в работах [8 и I8]. В СПТ имеется возможность выбирать, какой из этих методов использовать. Так как преобразование предшествования всегда возможно, никаких сообщений об ошибках не выдается.

Следующий шаг – это устранение конфликтов редуцирования [I2]. Если в данной грамматике находится конфликт редуцирования (A,B,x) такой, что $C_A^{1,1} \cap C_B^{1,1} \neq \phi$ и $C_A^{\#,1} \cap C_B^{\#,1} = \phi$, то можно выдавать графы контекста ζ_A и ζ_B [I2]. По этим графам возможно преобразование левого контекста $(\#,0) \Rightarrow (1,0)$. Также имеется возможность преобразовать этот контекст автоматически. Обычно в грамматике находятся и такие конфликты редуцирования, что $C_A^{\#,1} \cap C_B^{\#,1} \neq \phi$. Практика показывает, что примерно 90% из таких конфликтов обусловлены неоднозначностью грамматики, т.е. это ошибки пользователя, допущенные им во время описания синтаксиса языка. Остальные конфликты либо разрешимы с помощью LR(к), где $к > 1$, либо нет. Так как не существует алгоритма для решения проблем LR(к) и проблем однозначности грамматик [6, I6], то автоматическое устранение таких конфликтов невозможно. Имеющиеся ЭВМ не позволят использовать приема, когда системе дается какое-то определенное время для преобразования грамматики в (1/1) -редуцируемую, предполагая, что грамматика является LR(1)-, LR(2)- и т.д. -редуцируемой [9]. Таким образом, устранение конфликтов редуцирования частично приходится производить вручную. Поэтому предпочитается эту работу производить на ЭВМ WANG 2200 B, так как возможна непосредственная работа с этой ЭВМ при помощи дисплея. В ВЦ ТПИ осуществлена прямая связь между ЭВМ "Минск-32" и WANG 2200 B и связь между ЭВМ "Минск-32" и ЕС-I020 через магнитную ленту. Таблицы грамматики находятся в памяти "Минск-32", а на WANG 2200 B создано математическое обеспечение, дающее нам следующие возможности:

I) расположения правил подстановки по заданному признаку;

2) выдачи графа контекста нетерминала $A(\zeta_A)$;

3) при конфликте редуцирования (A,B,x), если $C_A^{\#,0} \cap C_B^{\#,0} = \phi$, нахождения правил подстановки

$$p_1^A \dots p_{n_A}^A,$$
$$p_1^B \dots p_{n_B}^B,$$

дающих первый различный односимвольный контекст слева от основы для нетерминалов A и B;

4) переноса к основе левого контекста, найденного по таким правилам подстановки, что устраняет конфликт редуцирования;

5) выдачи AND/OR графа грамматики в виде дерева [5].

После устранения рассмотренных конфликтов осуществляется связь между ЭВМ WANG ⇒ "Минск-32" ⇒ ЕС и с помощью Конструктора [I5] получаем зависимые от создаваемого языка части транслятора.

2. Требования, предъявляемые к обработке синтаксических ошибок

Обработка синтаксических ошибок в создаваемых трансляторах должна соответствовать следующим требованиям.

I. Локализация всех появляющихся ошибок – т.е. достижение такого положения, когда анализ можно продолжать и после принятых мер вторичные ошибки не появляются. Для выполнения этой задачи часто используется метод "паники" [4], при котором часть текста, стоящая между какими-нибудь разделителями и в которой содержится ошибка, извлекают из анализа. Недостаток такого метода состоит в том, что в дальнейшем он может вызвать вторичные ошибки.

2. Автоматическое исправление простых ошибок – т.е. система должна решать, можно ли исправить данную ошибку и если можно, то найдется ли исправление, требующее изменений анализируемого текста только в заданных пределах. И даже тогда, когда исправление не совпадает с замыслом пользователя, результат все-таки положительный, так как ошибка была локализована и данное исправление может помочь пользователю найти действительную причину его ошибки.

3. После окончания обработки очередной ошибки система по мере возможности должна продолжать построение син-

таксического дерева, так как наличие этого дерева дает возможность проводить частичный семантический анализ уже в синтаксическом анализаторе. Система должна решать, в каком случае построение синтаксического дерева возможно; в каком нет и в случае невозможности продолжения этой работы блокировать ее.

4. Выдача как можно подробной диагностической информации по каждой ошибке. Нужны сообщения о том:

а) что была обнаружена ошибка;

б) в чем состоит ошибка;

в) что предпринято для устранения ошибки.

Особенно важны подробные сообщения об ошибке тогда, когда проводится только локализация ошибки. В этом случае пользователю надо точно знать, в какой именно конструкции была ошибка.

5. Время обработки ошибок не должно слишком удлинять время анализа.

6. Пользователю должна быть дана возможность выбрать, какие ресурсы он хочет использовать при обработке ошибок:

а) достаточно ли ему локализации ошибок, или он хочет использовать исправление ошибок, требующее большего объема времени и памяти машины;

б) как подробно и в каких понятиях пользователь хочет получить информацию об ошибках.

Надо учитывать то обстоятельство, что системно-ориентированные языки в СПТ создают в расчете на пользователя, не имеющего специальной подготовки по ЭВМ и программированию. Ему не будут понятны сообщения об ошибке, данные во внутрисистемных понятиях. Хотя бы во время ознакомления с системой и в период обучения языку пользователь нуждается в сообщениях, данных в своих понятиях (понятиях метаязыка). В дальнейшем, когда пользователь уже ознакомится с особенностями системы и языка, он может ограничиваться информацией менее подробной и данной в понятиях самой системы. Таким образом, СПТ принимает на себя в некоторой степени даже функцию обучения пользователя.

3. Обработка ошибок во время трансляции

В работе трансляторов появляются ошибки на трех этапах:

1) в ходе лексического анализа – ошибки в лексемах;

2) в ходе синтаксического анализа – синтаксические ошибки;

3) в ходе семантического анализа – семантические ошибки.

Исследуем подробнее эти типы ошибок и реакцию на них в рассматриваемой СПТ.

1. При лексическом анализе (сканировании) обычно используются две тактики:

а) формы лексем зафиксированы заранее,

б) формы лексем определяет пользователь.

Вторая возможность наиболее гибка и удобна для пользователя, но недостаток ее состоит в большой вероятности появления ошибок. Пользователь, не имеющий специальной подготовки, может дать разным лексемам определения, которые частично совпадают и т.д. Используя фиксированные лексемы можно избежать этих ошибок. В данной системе определены четыре класса лексем: идентификатор, константа, стринг и разделитель. Лексический анализ обнаруживает ошибки в константах и стринги недозволенной длины. В обоих случаях синтаксическому анализатору передается правильный код этих лексем и выдается сообщение об ошибке. В первом случае лексический анализ продолжается так, как будто ошибки не было, во втором случае продолжается анализ с первого разделителя в ошибочном стринге.

2. Исправить все синтаксические ошибки формальными методами невозможно, так как исправление может быть определено семантикой. Если даже исправления возможны, не всегда целесообразно их делать. Исправления, требующие существенных изменений в начальном тексте, делать не целесообразно, так как маловероятно, что результат совпадает с замыслом пользователя. Чтобы избежать это, надо с каждым исправлением связать какую-нибудь цену, по которой было бы

возможно определить наилучший вариант исправления и решить, целесообразно ли его осуществление. Независимо от того, сделали исправление или нет, должна быть обеспечена локализация ошибки.

В настоящее время пока еще ни один универсальный метод исправления синтаксических ошибок не нашел всеобщего признания и использования. Первые методы были либо зависимы от языка, либо были направлены только на локализацию ошибок. Можно согласиться с О.Лекармом [I7] (который имеет большой практический опыт по использованию СПТ), что самым многообещающим среди наиболее новых методов [см. 2,3,I0] является метод Грехема и Родеса (метод ГР) [4]. Этот метод критиковали как подходящий только для использования в анализе с использованием метода предшествования [I0], но данный анализатор отвечает именно этому требованию.

Метод ГР можно разделить на два этапа: на фазу "конденсации" и фазу исправления. Задача фазы конденсации - собрать контекст, окружающий место ошибки. Делается попытка продолжать синтаксический анализ налево и направо от места ошибки. В фазе исправления пытаются изменять сконденсированный стек в окружении ошибки так, чтобы ошибочная ситуация была исправлена и стек содержал символьную цепочку, которая могла бы появиться при анализе предложения данного языка.

В использованном в СПТ методе сохранены главные принципы метода ГР. Но использование каскадного метода редуцирования потребовало некоторого приспособления метода ГР, благодаря чему расширены возможности нахождения ошибок и определения места появления ошибки. Контроль контекста при редуцировании помогает избегать неправильных исправлений. Усовершенствован метод "паники", используемый Грехемом и Родесом, если исправление не удалось - конструкция, в которой содержится ошибка, не извлекается из анализа, а заменяется специальным понятием, позволяющим продолжать анализ. Орфографические ошибки в резервированных словах, появляющиеся относительно часто и легко поддающиеся исправлению, исправляются отдельно от других ошибок. Все эти изменения и новшества плюс реализация метода исправления синтаксических ошибок для значительно большего класса грамма-

тик, чем в работе Грехема и Родеса, позволяют в СПТ эффективно исправлять или локализовать все синтаксические ошибки. Предпосылкой для локализации ошибок является заданная пользователем связь между понятиями и терминалами.

В сообщениях об ошибках пока дается информация о том, появление какого символа породило ошибку и какое сделано исправление. Если исправление не удалось, пользователю сообщается, в какой конструкции содержалась ошибка. Оставлена также возможность использования в сообщениях об ошибке понятий метаязыка.

Соответствующие программы для ЭВМ ЕС-1020 находятся в стадии внедрения. Так по предварительным результатам, при использовании грамматики с 27 правилами исправление ошибки занимает в среднем 14 секунд.

3. Трансляторы, генерированные данной СПТ, работают по синтаксически ориентированной схеме [1], снабженной элементами атрибутной грамматики [7]. Семантическая обработка осуществляется на двух этапах.

а) Первичная семантическая обработка [13], где вычисляют значения глобальных атрибутов. Ошибки анализируют и сообщения об ошибках выдают соответствующими семантическими программами СПТ.

б) Вычисление названия программ, связанных с вершинами дерева синтаксиса, и активизация соответствующих программ. То есть из дерева синтаксиса генерируется текст базового языка, где базовым языком является либо

- Ассемблер,
- PL/I,
- Ассемблер и PL/I.

В этом случае генерация сообщений об ошибках частично осуществляется при помощи семантических атрибутов и вычисляемых кодов, а частично становится заботой программного обеспечения ЭВМ.

Таким образом, охвачены все аспекты общения пользователя с системой и даны ответы на поставленные им вопросы. В заключение можно подчеркнуть, что пользователю безразлично, по каким принципам реализована СПТ, его больше интересуют ее возможности.

Литература

1. Aho, A.V., Ullman, J.D. The Theory of Parsing, Translation and Compiling. Vol. 2 - Compiling. USA, New Jersey, Prentice-Hall, 1973.

2. Ciesinger, J. Generating Error Recovery in a Compiler Generating System. Fachtagung über Programmiersprachen. Springer-Verlag. Berlin-Heidelberg-New York, 1976, p. 185...193.

3. Feyock, S., Lazarus, P. Syntax-directed Correction of Syntax Errors.- Software-Practice and Experience, 1976, Vol. 6, p. 207...219.

4. Graham, S.L., Rhodes, S.P. Practical Syntactic Error Recovery.- Comm. ACM, 1975, 18, N 11, p. 639...650.

5. Hall, P.A.V. Equivalence between AND/OR Graphs and Context-free Grammars.- Comm. ACM, 1973, 16, N 7, p. 444...445.

6. Knuth, D.E. On the Translation of Languages from Left to Right.- Inform. Contr., 1965, 8, p. 607...639.

7. Lewis, P.M., Rosenkrantz, D.J., Stearns, R.E. Attributed Translations.- Journal of Computer and System Sciences, 1974, 9, p. 279...307.

8. McAfee, J., Presser, L. An Algorithm for the Design of Simple Precedence Grammars.- J. ACM, 1972, 19, N 3, p. 385...395.

9. Mickunas, M.D. On the Complete Covering Probleme for LR(k)-Grammars.- J. ACM, 1976, 23, N 1, p.17...30.

10. Sippu, S., Soisalon-Soininen, E. On Defining Error Recovery in Context-free Parsing. Lecture Notes in Computer Science, 51, Springer-Verlag. Berlin-Heidelberg-New York, 2,1977, p. 492...503.

II. Бауэр Ф., Гоос Г. Информатика. М., "Мир", 1976.

I2. Вооглайд А.О. Семантическое равенство распознавателей, работающих на грамматике LR(к) и грамматике предшествования с (1/1) ограниченным контекстом.- "Тр. Таллинск. политехн. ин-та", 1976, № 4II, с. 39-56.

I3. Вооглайд А.О. Расширенный анализатор предшествования со смешанной стратегией.-"Тр. Таллинск. политехн. ин-та", 1977, № 423, с. 23-4I.

I4. Вооглайд А.О., Томбак М.О. О проблемах редуцирования в грамматиках предшествования. - "Тр. Таллинск. политехн. ин-та", 1975, № 386, с. 23-38.

I5. Вооглайд А.О., Томбак М.О. Система построения эффективных многопроходных транслеторов с LR(к) семантикой. "Программирование", 1976, № 5, с. 28-38.

I6. Гинзбург С. Математическая теория контекстно-свободных языков. М., "Мир", 1970.

I7. Лекарм О. Практичность и переносимость системы построения трансляторов.-"Труды всесоюзного симпозиума по методам реализации новых алгоритмических языков", часть I, Новосибирск, 1975.

I8. Томбак М.О. Об устранении конфликтов предшествования.-"Тр. Тартуского ГУ", 1976, 37, с. 60-9I.

A. Vooglaid, H. Rohtla

Error Processing in a Compiler Writing System

Summary

This paper describes all the types of errors that can arise when working with a compiler writing system and presents corresponding system reactions.