5. Шатровский Л.И. Макропрограммирование в вычислительной среде ДОС/ЕС. - М.: Советское радио, 1979.
6. Инженерные расчеты на ЭВМ. Справочное пособие под редакцией проф., д-ра физ.-мат. наук Троицкого В.А. - Л.: Машиностроение, 1979.

УДК 681.3.06:5

В.В.Куликов, М.А.Шамашов

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ СИНТАКСИЧЕСКИХ АНАЛИЗАТОРОВ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ СИСТЕМ АВТОМАТИЗАЦИИ ЭКСПЕРИМЕНТА

Решение задач автоматизации экспериментальных исследований, как правило, требует использования специально разрабатываемого проблемно-ориентированного программного обеспечения (ПО). В частности, возникает необходимость построения трансляторов для языков управления экспериментом, реализуемых в диалоговом и пакетном режимах, и систем подготовки информационных массивов заданной структуры [1-3]. Подобные задачи возникают и при построении моделей специализированных УВМ, входящих в состав систем автоматизации эксперимента [4, 5]. Большая стоимость ПО требует создания инструментальных систем автоматизации программирования (ИСАП).

В предлагаемой статье рассматривается один из подходов автоматизации процесса построения синтаксических анализаторов для трансляторов (САТ) проблемно-ориентированных языков - одной из подсистем ИСАП.

Разработанная система автоматизации проектирования САТ базируется на уже известном наборе команд магазинного преобразователя (МП-преобразователя) [6]. Исходная КС-грамматика задана в виде пятерки $G=(V_N, V_T, S, P, \#)$, где V_T и V_N - конечные алфавиты основных и вспомогательных символов, $V_N \cap V_T = O$, S - начальный символ грамматики, $S \in V_N$, P - множество правил вывода вида

$$R_i(\psi_i \to \varphi_{ij} \mid i=\overline{1,n};\ j=\overline{1,m_i};\ |\psi_i|=1) \in P.$$

По исходной КС-грамматике строится МП-преобразователь:

$$A = (Q, V, \Gamma, W, \delta, q_0, z_0, F),$$

где $Q = \vec{Q} \cup \hat{Q}$ - конечное множество состояний автомата;

$\vec{Q}$ - множество состояний, из которых возможен переход по очередному символу $A_i \in V_T \cup V_N$ грамматики $G = (V_N, V_T, S, P, \#)$;

$\hat{Q}$ - множество состояний, в которых производится свертка;

V - конечный входной алфавит;

Γ - конечный алфавит магазинных символов;

W - конечный выходной алфавит;

δ - отображение $(Q \times (V \cup \{e\} \times \Gamma) \to P(Q \times \Gamma^* \times W^*)$;

$q_0 \in Q$ - начальное состояние МП-преобразователя;

$z_0 \in \Gamma$ - начальный символ магазина;

$F \subseteq Q$ - заключительное состояние МП-преобразователя.

Отображение δ :

$$(q, x, I) \to \{(q_i, W_i, \gamma_i)\} \quad (i = \overline{1, r},\ q, q_i \in Q,\ x \in V,\ I \in \Gamma,\ \gamma_i \in \Gamma^*)$$

интерпретируется следующим образом: МП-преобразователь, находясь в состоянии q, считывает со входной ленты символ $a \in (V_N \cup V_T)$. При этом

1. Если $q = \vec{q}$, в магазин заносится индекс I состояния, в которое переходит МП-преобразователь. На выходную ленту заносится строка символов W_i.

2. Если $q = \hat{q}$, то из магазина удаляется k символов ($k = |\varphi_{ij}|$ для $P_i : \varphi_i \to \varphi_{ij}$, читающая головка на входной ленте сдвинется на k позиций влево и на входную ленту записывается символ $A_i \equiv \varphi_i$.

Множество $Q \times V \times \Gamma$ выполняет функцию программы МП-преобразователя и моделируется тремя типами команд: сравнить, свернуть, перейти [7]. Каждая из функций этих команд в ИСАП расширена. Так, в качестве элементов сравнения и свертки можно использовать многосимвольные конструкции. Элементы сравнения могут быть набором символов для непосредственного сравнения, именами синтермов, т.е. обозначениями для целой группы терминалов (термов), либо именами подпрограмм синтаксического анализа. Последнее позволяет разбить общую программу синтаксического анализа на ряд подпрограмм в соот-

ветствии с уровнями исходной грамматики и использовать эти подпрограммы как стандартные для построения анализаторов с различных языков программирования.

Кроме этих трех команд, в состав ИСАП введен ряд дополнительных команд, реализующих отладочные и сервисные функции и стандартные соглашения о связях между подпрограммами анализатора (в частности, обеспечивающих рекурсивные обращения к этим подпрограммам). По выполнении каждой команды сравнения и свертки подсистема дает возможность выполнить практически неограниченный набор семантических подпрограмм, написанных пользователем. Это позволяет совмещать синтаксическую фазу с фазой непосредственной генерации кодов.

В процессе анализа исходной строки подсистема может вырабатывать размеченную строку, в которой представлены все нетерминальные элементы, получаемые в процессе анализа исходной строки, с указанием номеров вхождения метасимвола (нетерминала) в терминальную строку. Это дает возможность полностью разделить фазы синтаксического анализа и собственной трансляции текста, что в ряде случаев бывает весьма удобным.

Рассмотрим отдельно каждую из основных команд анализатора.

I. Команда "Сравнить и перейти" - CAB *(Compare And Branch)* содержит набор ключевых параметров. С помощью параметра *T =* одна команда CAB определяет произвольное число конструкций сравнения. Параметром *LAB =* команда CAB задает метку, идентифицирующую состояние МП-автомата данной подпрограммы анализатора, на которую следует передать управление в случае совпадения конструкции сравнения из CAB и поля рабочего стека. Здесь совпадение понимается в более широком смысле, чем просто непосредственное равенство литералов. Для каждой конструкции характерен следующий порядок сравнения:

а) конструкция интерпретируется как обычный набор символов для непосредственного сравнения. В частности, здесь же производится сравнение с системными синтермами "A" и "9" - любая буква латинского алфавита или любая цифра;

б) если конструкции не совпадают, осуществляется анализ синтермов, заданных пользователем с помощью команды анализатора *SINTERM*, например, *OPER SINTERM (+, -, *, /, **)*;

в) при несовпадении конструкций *(в пп. а и б)* конструкция сравнения интерпретируется как внешнее имя (имя подпрограммы) и

анализируется его наличие в таблице, задаваемой командой анализатора *EXTERNAL*.

Если конструкция сравнения не совпадает ни с одним именем таблицы, либо подпрограмма с данным именем не найдена, весь процесс сравнения повторяется для следующей конструкции (если их несколько).

При несовпадении элемента входной строки ни с одной из конструкций сравнения производится последовательный переход на следующую команду подпрограммы анализатора.

При совпадении конструкций перед переходом на метку, указанную параметром *LAB* = , может выполнятся неограниченный набор семантических подпрограмм, определяемых необязательным параметром *SEM* = . Метка из *LAB* заносится в магазин адресов, а суммарная длина проанализированной части входного стека - в магазин длины.

Общий формат команд CAB имеет вид:

[< метка >] CAB T = (< список конструкций сравнения >),
LAB = < метка очередной команды анализатора >,
[, *SEM* = (< список семантических подпрограмм >)]

< метка > = < буква > | < цифра > | < метка > < буква > | < метка > < цифра >
< список конструкций сравнения >:: = < конструкция сравнения > |
< список конструкций сравнения > , < конструкция сравнения >
< конструкция сравнения >:: = < набор символов *ДКОИ* > | '(' | ')' | ""
< набор символов ДКОИ >:: = любой символ ДКОИ, кроме указанных отдельно.

Список семантических подпрограмм определяет набор имен подпрограмм через запятую.

Из определения видно, что метка может быть цифрой, что удобно для последовательной нумерации состояний МП-автомата.

Рассмотрим пример записи команды CAB:

100 CAB T = *(BEGIN, '(', "", PROC)*,
SEM = (BEGPROC), LAB = 120

2. Команда "Свернуть" - *RDN(ReDutiON)* имеет формат:

[< метка >] *RDN VAR* = < конструкция свертки >,
NUM = < число сворачиваемых конструкций >,
[, *SEM* = (< список семантических подпрограмм >.

По этой команде магазин адресов и длины сокращается на число элементов, заданных в *NUM*, в просмотренную часть стека за-

носится конструкция свертки, выполняется набор семантических подпрограмм (если он задан), и управление передается на команду с меткой из головы магазина.

3. Команда "Перейти" - *JMP(JuMP)* имеет формат:

$$[<\text{метка}>]\quad JMP\ \left\{\begin{matrix}<\text{метка}>\\,<\text{имя}>\end{matrix}\right\}.$$

Эта команда осуществляет переход либо на команду подпрограммы анализатора данного уровня (переход по "метке"), либо на поименованную команду из оставшейся части исходного модуля, содержащего данную подпрограмму анализатора (переход по "имени").

Дело в том, что подпрограммы анализатора могут включать,кроме специальных команд анализатора, любые конструкции на языке АССЕМБЛЕРА. Запрещается только использовать знак @ в качестве начальных символов внешних адресных констант и точек входа.

Кроме перечисленных выше команд, в программах анализатора должны использоваться следующие команды:

BEGIN - "Инициализировать анализатор",
NEXT - "Определить подпрограмму анализатора",
RDART - "Возврат из подпрограммы анализатора",
FINISH - "Определить режим отладки анализатора",
ERROR - "Сообщить об ошибках и обработать их".

Для проверки выбранного метода и реализованной системы использовался анализатор арифметических выражений [8]. Сгенерированная программа анализатора состоит из 46 состояний МП-автомата(см. статью Гамин П.В., Куликов В.В. Генерация программ синтаксического анализатора. Наст. сборник). Алгоритм анализа по этой грамматике значительно проще известных методов [8], так как не требует никаких дополнительных сведений. При изменении грамматик изменяется только программа (таблица состояний), что делает анализатор синтаксически ориентированным.

В качестве примера, иллюстрирующего запись анализатора в предлагаемой системе, рассмотрим подпрограмму анализа идентификатора, входящую в состав анализатора арифметических выражений.

```
IDENTIF  NEXT
I        CAB     T= ИДЕНТИФ, LAB=2
         CAB     T= A, LAB=3, SEM=ASSEMBL
         JMP     5
2        CAB     T=(A,9), LAB=4, SEM=ASSEMBL
         RDART   SEM=TABLE
3        RDN     NUM=1, VAR=ИДЕНТИФ
         RDN     NUM=2, VAR=ИДЕНТИФ
4
5        ERROR   C1='НЕВЕРНА ЗАПИСЬ ИДЕНТИФИКАТОРА'
         FINISH  1, ALL
         END
```

Здесь семантическая процедура *ASSEMBL* осуществляет сборку идентификатора в рабочей области, а процедура *TABLE* заносит собранный идентификатор в таблицу идентификаторов и ставит ему в соответствие адрес области памяти ЭВМ.

Все рассмотренные команды анализатора реализованы с использованием макроассемблера ДОС ЕС [9] на машинах серии ЕС ЭВМ и М-4030.

Макрокоманды анализатора содержат наборы констант и обращения к подпрограммам, реализующим функции конкретной команды. Объем памяти, требуемой этим подпрограммам, составляет 10 К. Общий объем памяти, требуемый для транслятора, зависит от числа команд и объема семантических подпрограмм.

Кроме рассмотренной выше подсистемы ИСАП, автоматизирующей процесс проектирования синтаксических анализаторов для трансляторов, в настоящее время разработана подсистема, генерирующая подпрограммы анализатора МП-автомата на основе описания языка в нормальной форме Бэкуса [10].

В дальнейшем предполагается расширение этой системы за счет включения в нее типового набора семантических процедур транслятора. Кроме того, эта система может быть расширена за счет включения в нее ряда элементов технологического комплекса программиста *RTK* [11], имеющего большой набор стандартных процедур трансляции и удобные средства для работы с файлами на внешних накопителях.

В заключение следует отметить, что на базе рассмотренных команд при определенной доработке можно строить не только анализаторы, но и генераторы контрольных тестов для них и генераторы информационных массивов заданной структуры [2 , 3].

Литература

I. Куликов В.В. О структуре данных в системе автоматизированной обработки информации. - В сб.: Автоматизация экспериментальных исследований. Вып. 9, Куйбышев, КуАИ, 1977.

2. Куликов В.В., Трещев С.И. Моделирование стохастических информационных массивов заданной структуры. УП Всесоюзное совещание по теории и методам математического моделирования. Тезисы докладов. - М.: Наука, 1978.

3. Куликов В.В., Трещев С.И. Моделирование информационных массивов. - В сб.: Автоматизация экспериментальных исследований. Вып. 10, Куйбышев, КуАИ, 1979.

4. Будячевский И.А., Коварцев А.Н., Кораблин М.А., Шамашов М.А. Двухэтапная схема интерпретации специализированных УВМ. - УСиМ , 1978, № 3.

5. Коварцев А.Н., Кораблин М.А., Шамашов М.А. Имитационное моделирование системы автоматизации эксперимента с использованием эмуляторов полной конфигурации. - УСиМ , 1979, № 4.

6. Виттих В.А., Куликов В.В. Структурный подход к разработке систем программного обеспечения автоматизированных систем обработки телеметрической информации. I Всесоюзная конференция по технологии программирования. Тезисы докладов. - Киев, 1978.

7. Шумей А.С., Зонис В.С. О синтаксическом анализе по однозначным грамматикам. - Программирование, 1975, № 3.

8. Ингерман П. Синтаксически ориентированный транслятор. - М.: Мир, 1969.

9. Программирование на языке Ассемблера ЕС ЭВМ. - М.: Статистика, 1975.

10. Гамин П.В., Куликов В.В., Шамашов М.А. Система автоматизации проектирования синтаксических анализаторов. Всесоюзное совещание по автоматизации проектирования трансляторов. Тезисы докладов. Таллин, 1980.

11. Вельбицкий И.В., Ходаков В.Н., Шолмов Л.И. Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6. - М.: Статистика, 1980.