УДК 681.3

В.В.Куликов, П.Д.Скобелев

ОБ ОДНОМ АЛГОРИТМЕ ПОСТРОЕНИЯ $LR(k)$ АНАЛИЗАТОРОВ

(г.Куйбышев)

В настоящее время широко используются и развиваются методы синтаксического анализа языков, основанные на технике $LR(k)$ анализа, предложенного в статье [1]. Так, дальнейшие исследования в этой области [2-7] позволяют представить $LR(k)$ анализатор в виде таблицы состояний автомата с магазинной памятью (МП-автомата), минимизировать число состояний данного автомата и сократить использование им магазинной памяти, что, в отдельных случаях, дает возможность заменить МП-автомат конечным автоматом и существенно ускорить анализ входного языка. Кроме того, метод $LR(k)$ анализа позволяет успешно обрабатывать языки высокого уровня (класс $LR(1)$ -грамматик порождает в точности детерминированные контекстно-свободные (КС) языки [8]), причем соответствующий анализатор обладает наилучшими характеристиками [2]. Это обеспечивает эффективность применения $LR(k)$ анализаторов, в частности, для построения транслирующих систем. Однако для подобных применений необходимы удобные на практике методы построения анализаторов. Остановимся на некоторых особенностях существующих методов построения $LR(k)$ анализаторов.

Построение синтаксических анализаторов указанного типа для больших k существенно затруднено, так как объем работ, выполняемый при построении $LR(k)$ анализатора, резко возрастает с увеличением размеров грамматики и ростом k. Поэтому обычно используются методы, основанные на построении $LR(0)$ анализатора. Однако, поскольку класс $LR(0)$ грамматик уже, чем обычно необходимо, в дальнейшем применяются специальные методы устранения конфликтных ситуаций в полученном $LR(0)$ анализаторе - таким образом класс $LR(0)$ грамматик расширяется (SLR [2], $LALR$ [3], [4] и т.д.). С другой стороны, существует общий алгоритм, применимый к любой КС- грамматике, позволяющий преобразовать $LR(k)$ грамматику при $k \neq 0$ в $LR(k-1)$ грамматику [9] но при этом могут быть потеряны символы, необходимые для интерпретации конструкций рассматриваемого языка. Таким образом, применение подобного метода позволяет уменьшить объем требуемых вычислений, сократить размеры получаемых анализаторов и описать достаточно широкий класс языков.

В настоящей статье приводится модификация алгоритма [4] построения программ $LR(k)$ анализаторов, обеспечивающая дополнительно:

проверку требуемых свойств (конфигурационности) грамматики в процессе построения анализатора;

ускоренную перестройку программы анализатора при изменении входного языка;

возможность работы с отдельными элементами (конструкциями) языка без изменения программы анализатора;

построение структурированного МП-автомата[10].

Введем некоторые обозначения и кратко рассмотрим построение анализатора по методу $LR(k)$ грамматик [4].

Пусть порождающая грамматика представлена в виде $G=(V_T, V_N, P, S)$, где V_T – конечный набор символов терминального алфавита, V_N – конечный набор символов нетерминального алфавита, $V_N \cap V_T = \phi$, S – начальный символ (аксиома) грамматики G, $S \in V_N$, P – конечный набор правил вывода вида $\varphi \to \psi$, где $\psi \in (V_N \cup V_T)^+$, $\varphi \in V_N$ и $|\varphi| = 1$ (КС-грамматика) [4].

Программа анализатора строится для данной грамматики путем построения множества конфигураций. Конфигурация записывается в виде $\varphi \to a \circ b$, где a и b могут быть порознь пустые, $a, b \in (V_N \cup V_T)^*$ (строчными буквами будем обозначать строки символов из объединенного алфавита $V = V_N \cup V_T$, прописными – символы из этого алфавита) $\circ$ – маркер, выделяющий очередной рассматриваемый из V в подстроке b. Каждому состоянию соответствует группа конфигураций Z_i. Из одного состояния Z_j возможен один или несколько переходов в другие состояния Z_i, множество строк, по которым возможен переход из Z_j в Z_i будем обозначать $W(Z_j, Z_i)$. Число переходов в состоянии Z_j определяется числом различных символов, стоящих за маркером в конфигурациях этого состояния. Конфигурацию вида $\varphi \to \circ \psi$ будем называть исходной, а $\varphi \to \psi_0 \circ$ – конечной. Конечная конфигурация указывает на необходимость замены (редукции) строки ψ на символ φ.

Пусть из Z_j в Z_i можно перейти по символу A. Тогда часть конфигураций состояния Z_i будет прямо выводиться из конфигураций состояния Z_j по формуле

$$Z_i^* = \{\varphi \to xA \circ y \mid (\varphi \to x \circ Ay) \in Z_j \wedge A \in W(Z_j, Z_i)\}, \qquad (1)$$

где Z_i^* – начальная группа конфигураций состояния Z_i

Полная группа конфигураций состояния Z_i определяется следующим образом:

$$Z_i = Z_i^* \cup \{A \to \circ\psi \mid (y \to x \cdot Ay) \in Z_i\}, \qquad (2)$$

т.е. добавлением исходных конфигураций для очередных символов.

Определение множества состояний начинается с начального состояния Z_1, в которое включаются конфигурации вида $S \to \circ\psi$, а затем, применяя правило (2), определяют полную группу конфигураций. Описанный процесс приводит к построению конечного числа групп конфигураций, которые могут быть заданы в виде программы МП-автомата [2,4].

Рассмотрим теперь подробнее работу модифицированного алгоритма (краткое описание и замечания по использованию данного алгоритма – в [II]).

Будем строить анализатор в процессе левого вывода из аксиомы грамматики G. Так же, как и в работе [6], для удобства описания процедуры построения анализатора поставим каждому нетерминальному символу, встреченному в процессе вывода, во взаимно-однозначное соответствие новые символы, составляющие алфавит помеченных символов V_N', причем будем обозначать символ из этого алфавита, соответствующий данному $A \in V_N$, как $\bar{A}$ в случае, если во множестве начальных состояний Z присутствует первое состояние Z_n^A анализатора языка, порождаемого данным нетерминалом, и $\underline{A}$ – в противном случае. В общем случае будем обозначать символы алфавита $V_N' \cup V_T$ как A', а строки, состоящие из символов данного алфавита – как a'.

. В процессе вывода будем последовательно применять правила подграмматики, определяемой рассматриваемым $A \in V_N$, причем переход к следующему правилу будет осуществляться в случае, если текущее правило порождает цепочку символов из множества $(V_N' \cup V_T)^+$. Такой переход будем фиксировать в выводе символом " | ".

Приостанов вывода происходит, когда все правила подграмматики, т.е.правила, левая часть которых представляет собой указанный символ, будут использованы:

$$S \Rightarrow^* a' A b \Rightarrow a' c_1' \mid c_2' \mid \ldots c_n' \mid b, \qquad (3)$$

где $\{A \to c_i \mid i = 1, n\} \in P$ – множество правил подграмматики A

Приостанов вывода означает, что для данного $A \in V_N'$ может быть построен анализатор, т.е. в правилах подграмматики A имеются либо терминальные символы, либо нетерминальные символы, для которых построены соответствующие анализаторы, либо такие нетерминальные символы, для которых анализатор не может быть получен на данном этапе

(отложенные символы). Анализатор строится подобно виду (I),(2),однако, исходные конфигурации не порождаются, а происходит перезапись уже имеющихся начальных состояний в текущее состояние, определяемое начальной конфигурацией, т.е.

$$z_i^0 = \{A \to x'B'\circ y' \mid (A \to x'\circ B'y') \in z_j \wedge B \in W(z_j, z_i)\},$$

$$z_i = z_i^0 \cup z_\kappa^B,$$

если $B \in V_N'$ (тогда $z_\kappa^B \in z$) или $z_i = z_i^0$, если $B \in V_N'$ (отложенный символ) или $B \in V_T$.

Кроме того, на первое состояние $z_i^A = z_i$, определяемое начальной конфигурацией, отсутствуют переходы из других состояний z_j программы анализатора, т.е. $W(z_j, z_i) = \phi$.

Построенное таким образом состояние z_i^0 включается во множество z, при этом, по возможности, осуществляется доопределение отложенных символов, в частности, если некоторое состояние z_κ было построено с учетом отложенного символа A, происходит перезапись z_i^A в z_κ.

На этапе построения промежуточного анализатора может быть выполнена частичная оптимизация программы анализатора (приводится соотношение, позволяющее в процессе вывода оценить число состояний анализатора при такой оптимизации [II]).

После того, как анализатор построен и доопределены возможные символы, вывод обращается, т.е. правая часть правила рассмотренной подграмматики заменяется определяющим нетерминальным символом, теперь $\bar{A}$ (обращенный вывод обозначим $=\!=>$):

$$\alpha' c_1' \mid c_2' \mid \ldots c_n' \mid \beta =\!=> \alpha' \bar{A} \beta,$$

после чего приостановленный вывод продолжается до тех пор, пока не будет достигнута аксиома грамматики.

Однако $\angle R(\kappa)$ анализ осуществим, если среди множества состояний анализатора нет неоднозначных, т.е. таких [4], в которых содержится более, чем одна, конечная конфигурация, либо истинно выражение

$$\exists((\varphi_1 \to \circ M\beta),\ (\varphi \to \varphi_0)) \in z_i \wedge (x\varphi \in W(z_1, z_i)) \wedge$$

$$\wedge (S =>^+ x\varphi y =>_x \psi y =>^+ x\varphi M u).$$

Грамматики, обладающие указанным свойством, названы к о н ф и г у р а ц и о н н ы м и [7], здесь же приводится алгоритм, позволяю-

щий для класса КС-грамматик распознать свойство конфигурационности. Для этого требуется построить программу анализатора (МП-автомата), после чего выделить и проверить состояния, в которых возможна неоднозначность – это приводит к многократным просмотрам программы анализатора. Аналогичные методы [8] требуют построения и проверки матриц соответствующих отношений между символами исходной грамматики.

Покажем, как свойство конфигурационности может быть проверено в предлагаемом алгоритме. Пусть, для определенности, состояние, в котором наряду с конечной конфигурацией содержатся конфигурации, задающие переходы в другие состояния, единственно в каждой подграмматике. Назовем такое состояние заключительным, обозначив как $\hat{z}^{A}_{m}$, и зафиксируем его, подобно тому, как фиксируется начальное состояние z^{A}_{K}. Если в текущем состоянии имеется $\bar{A} \in V_N'$, то очередные символы перехода в заключительном состоянии $\hat{z}^{A}_{K}$ должны быть сопоставлены с первым символом цепочки в (3),(4), тогда условие нарушения конфигурационности принимает вид (наличие нескольких конечных конфигураций в одном состоянии проверяется на этапе формирования состояния)

$$\exists B \in W(\hat{z}^{A}_{K}, z_n) \wedge (B = BC,\ A \in W(z_i, z_j),\ \bar{A} \in V_N').$$

В общем случае, если таких заключительных состояний несколько, данное условие должно быть проверено для каждого из них при отсутствии отложенных символов. В процессе вывода за рассматриваемым нетерминальным символом появляются все возможные варианты цепочки B, и каждый раз проверяется данное условие (за исключением случая пустой цепочки), таким образом в процессе вывода проверяются только те нетерминальные символы, для которых это условие может быть нарушено.

Пример. Рассмотрим сокращенный вариант грамматики языка диалога, разработанного для проведения физико-технического эксперимента [12]:

$G_1 = (V_T, V_N, P, S)$, $P = \{S \to z;, z \to z, O|O, O \to GNKV, K \to (R),$
$R \to R, Q|Q, Q \to u|x|y, V \to =T/M\}$, $V_T = (;, ,, (,), u, x, y,$
$G, N, =, T, M)$, $V_N = (S, z, O, K, R, Q, V)$.

Вывод для аксиомы S имеет вид: $\underline{S} \Rightarrow \underline{z}; \Rightarrow \underline{z}, \underline{Q}; \Rightarrow \underline{z}, GN\underline{K}V; \Rightarrow$
$\Rightarrow \underline{z}, GN(\underline{R})V; \Rightarrow \underline{z}, GN(\underline{R}, Q)V; \Rightarrow \underline{z}, GN(\underline{R}, u|x|y|)V; = \Rightarrow \underline{z}, GN(\underline{R},$
$\bar{Q})V; \Rightarrow \underline{z}, GN(\underline{R}, \bar{Q}|\bar{Q}|)V; = \Rightarrow \underline{z}, GN(\bar{R})V; \Rightarrow \underline{z}, GN(\bar{R})/V; = \Rightarrow \underline{z},$
$GNRV; \Rightarrow \underline{z}, GN\bar{R}\underline{V}; \Rightarrow \underline{z}, GNK = T; \Rightarrow \underline{z}, GNK = T/M|; = \Rightarrow \underline{z}, GNR\bar{V}; \Rightarrow$
$\Rightarrow \underline{z}, GNK\bar{V}|; = \Rightarrow \underline{z}, \bar{O}; \Rightarrow \underline{z}, \bar{O}|\bar{O}|; = \Rightarrow \bar{z}; \Rightarrow \bar{z}; | = \Rightarrow \bar{S} \Rightarrow \bar{S}|.$

Программа анализатора, построенная в процессе вывода, показана на рисунке (а). Фигурными скобками отмечены начальные состояния для каждой подграмматики. Наклонная черта обозначает появление нового состояния, которое образуется в результате объединения переходов по одинаковым символам.

Пусть исходная грамматика модифицируется, например, конструкция V приобретает вид $\{V \to A/B\} \in P$. В этом случае необходимо, во-

а

№ сост.	команды		
1	СР	X	2
	СР	У	2
	СР	Ц	2
2	ПМ	a	1
3	СР	Ц	2
	СР	X	2
	СР	У	2
	СР	a	4
	СР	R	5
4	ПМ	R	1
5	СР	.	6
6	СР	Ц	2
	СР	X	2
	СР	У	2
	СР	a	7
7	ПМ	R	3
8	СР	(	9
9	СР	Ц	2
	СР	X	2
	СР	У	2
	СР	a	4
	СР	R	5/12
	СР	R	10/12
10	СР	)	11
11	ПМ	K	3
12	СР	,	8
	СР	)	11
13	СР	=	14
	СР	M	15
14	СР	T	16
15	ПМ	V	1
16	ПМ	V	2
17	СР	G	18

№ сост.	команды		
18	СР	N	19
19	СР	(	9
	СР	K	20
20	СР	=	14
	СР	M	15
	СР	V	21
21	ПМ	O	4
22	СР	a	18
	СР	O	23
	СР	Z	24
23	ПМ	Z	1
24	СР	)	25
25	СР	a	18
	СР	O	26
26	ПМ	Z	3
27	СР	G	18
	СР	O	23
	СР	Z	24/30
	СР	Z	28/30
28	СР	;	29
29	ПМ	S	2
30	СР	)	25
	СР	;	29

б

№ сост	команды		
13	СР	A	14
	СР	B	14
14	ПМ	V	1
20	СР	A	14
	СР	B	14
	СР	V	21

Р и с. Таблица состояний анализатора, построенная для грамматики G_1 (а) и состояний, заменяемых при модификации грамматики (б)

первых, заменить анализатор соответствующей подграмматики, во-вторых, модифицировать те состояния, в которых символ V является символом перехода. Для определения этих состояний требуется дополнительный вывод, в процессе которого, однако, построение (модификация) анализатора осуществляется только в том случае, когда рассматриваемый символ V выводится из определяющего символа подграмматики и присутствует в начальном состоянии анализатора, построенного для определяющего символа, либо символ V содержится в правой части одного из правил текущей грамматики. При этом нумерация состояний позволяет непосредственно определить номер состояния, требующего модификации. Так, для символа V новый анализатор строится, начиная с состояния I3 (рисунок (б)), а модифицируется состояние 20.

Принимая состояние z_{27}^{S} за начальное (z_1) и перенумеровав состояния, получим программу анализатора, совпадающую с построенной по методу [4].

Литература

1. *Knuth D.E. On the translation of languages from left to right. – Inf. Control, 1965, No 8, p. 607-639.* (Русский перевод: Кнут Д.Е. О переводе (трансляции) языков слева направо. – В сб.: Языки и автоматы. М.: Мир, I975, с. 9-42).

2. *De Remer F.L. Simple LR(k) grammers. – Comm. ACM, 1971, V. 14. No. 7, p. 453-460.*

3. *Korenjak A.J. A practical method for constructing LR(k) processors. – Comm. ACM, 1969, v. 12, No. 11, p. 613-623.*

4. Неданов И.Е., Шумей А.С. О расширении класса грамматик $LR(0)$. – Автоматика и телемеханика, I973, № II, с. I50-I54.

5. Шумей А.С., Зонис В.С. О синтаксическом анализе по однозначным грамматикам. – Программирование, I975, № 3, с. I23-I30.

6. Зонис В.С., Шумей А.С. Оптимизация синтаксического анализатора, построенного по методу $LR(k)$ грамматики. – Программирование, I976, № 2, с. 3-9.

7. Шумей А.С. Ускоренный синтаксический анализ символьных программ. – Автоматика и телемеханика, I976, № 5, с. I58-I64.

8. Ахо А.В., Ульман Д.Д. Теория синтаксического анализа, перевода и компиляции. Т. 2. – М.: Мир, I978. – 488 с.

9. Курочкин В.М. Об одном алгоритме преобразования $LR(k)$ грамматик. – В кн.: Труды Всесоюзного симпозиума по методам реализации новых алгоритмических языков. Ч. I, Новосибирск, 1975, с. 212-213.

10. Виттих В.А., Куликов В.В. Рекурсивное представление алгоритмов обработки и структур данных в АСНИ. – В кн.:Труды Всесоюзной конференции по теории кодирования и передачи информации. Куйбышев, 1981, с. 32-34.

11. Скобелев П.О. Об использовании синтаксически ориентированных методов трансляции для систем автоматизации программирования. – В сб.: Автоматизация научных исследований. Куйбышев, 1982, с.120-125.

12. Скобелев П.О. Вопросы реализации диалоговой подсистемы "ДИСТЕХ" в системе автоматизации физико-технического эксперимента. -Деп. рукопись, 1982.

УДК 518.62

Б.М.Шумилов

О ЧИСЛЕННОМ ОТЫСКАНИИ РАЗРЫВОВ ПРОИЗВОДНОЙ В ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

(г.Томск)

При конструировании обводов самолетов и кузовов автомобилей, при обработке медико-биологических и других экспериментальных данных большую информационную ценность имеет знание местоположения точек разрыва некоторой производной. Так, в самолетостроении разрыв второй производной линии обвода приводит к срыву обтекающего потока и развитию турбулентности; при обработке медико-биологических данных, например электрокардиограмм, точки разрыва первой производной (изломы) связаны с характерными точками, имеющими диагностическую ценность. Функции с разрывами производных возникают также в задаче отслеживания траекторий реактивных аппаратов с импульсным или разрывным управлением тягой и в задаче оконтуривания изображений, полученных методом вычислительной томографии. Общим для всех перечисленных примеров является то,что в качестве объекта исследования предлагается некоторая пространственная кривая, либо временная зависимость, по виду которой требуется вынести суждение о месте расположения точек разрыва некоторой производной. В элементарных случаях это может быть сделано визуально, либо путем применения методов физического моделирования, например, в само-