Глава 4. Восходящий синтаксический анализ

4.6. *LR*-таблицы разбора

4.6.3. *LALR*(1)-грамматики

Рассмотренный *SLR*(1)-метод недостаточно мощный, поскольку он не учитывает левый контекст, необходимый в общем случае для принятия решения о действиях синтаксического анализатора для разрешения конфликтов. В ряде случаев левый контекст играет важную роль при решении вопроса о том, можно ли считать заданный символ действительно символом-следователем, хотя в смысле *SLR*(1)-метода он и является возможным символом-следователем (входит в множество, определяемое функцией *Follow*).

Левый контекст учитывается при построении таблицы разбора так называемым *LR*(1)-методом с предпросмотром (Look Ahead), или *LALR*(1)-методом. Грамматика, для которой все конфликты в таблице разбора разрешаются при использовании данного метода, относится к подклассу *LALR*(1)-грамматик.

В *LALR*(1)-методе, чтобы учитывать левый контекст, состояние *LALR*(1)-автомата должно содержать дополнительную информацию. Это достигается с помощью предпросмотра. Поэтому в этом методе используется *LR*(1)-пункт вида $[A \to \alpha \bullet \beta, a]$, где терминал a является предпросмотром пункта. Множество таких терминалов всегда является подмножеством множества, определяемого функцией *Follow*.

Метод построения *LALR*(1)-таблицы разбора, по сути, тот же, что и *SLR*(1)-метод. Отличия связаны с определением предпросмотров пунктов и их использованием для разрешения конфликтов. Предпросмотр не играет никакой роли в пункте вида $[A \to \alpha \bullet \beta, a]$, где $\beta \neq \varepsilon$, а важен для пункта вида $[A \to \alpha \bullet, a]$, когда выполняется свертка для продукции $A \to \alpha$, если очередным входным символом является терминал a.

Предпросмотры определяются при выполнении операции замыкания. Пусть имеется $LR(1)$-пункт $[A \to \alpha \bullet B\beta, a]$, $B \in V_N$. Тогда должна существовать правосторонняя схема вывода

$$S \overset{*}{\Rightarrow} \delta A a x \Rightarrow \delta \alpha B \beta a x, \text{ где } \delta \in (V_T \cup V_N)^*, x \in V_T^*.$$

Предположим, что $\beta a x$ генерирует терминальную строку by, $b \in V_T$, $y \in V_T^*$. Тогда для любой продукции вида $B \to \gamma$, $\gamma \in (V_T \cup V_N)^*$, можно получить схему вывода

$$S \overset{*}{\Rightarrow} \delta \alpha B b y \Rightarrow \delta \alpha \gamma b y,$$

т. е. пункт $[B \to \bullet\gamma, b]$ должен войти в это же состояние. Необходимо обратить внимание на то, что символ b может быть первым терминалом, выводимым из β; возможно также, что β может генерировать пустую строку ε, в этом случае $b = a$. Следовательно, символ b может быть любым терминалом из множества $First(\beta a)$, где $First$ – функция, подробно описанная при рассмотрении $LL(1)$-грамматик.

Таким образом, операция замыкания заключается в следующем. Если пункт вида $[A \to \alpha \bullet B\beta, a]$ входит в некоторое состояние, то все пункты вида $[B \to \bullet\gamma, b]$ для всех терминалов b из $First(\beta a)$ включаются в это же состояние. Процесс продолжается рекурсивно до тех пор, пока в состояние не будут включены все возможные пункты.

Множество первых элементов *LR*(1)-пунктов, входящих в одно состояние, называется *ядром* состояния. *LALR*(1)-метод не допускает разных состояний с одинаковыми ядрами, даже если предпросмотры пунктов отличаются. Поэтому если грамматика дает более двух состояний с одним и тем же ядром, то эти состояния должны быть объединены в одно. В остальных случаях выполнение перехода из состояния в состояние-преемник не влияет на предпросмотр пункта.

Рассмотрим *LALR*(1)-метод на примере грамматики с продукциями

1) $S \to AaBb$

2) $A \to C$

3) $A \to db$

4) $B \to C$

5) $C \to Ccd$

6) $C \to d$

Для сравнения с *SLR*(1)-методом для всех нетерминалов определим значения функции *Follow*:

$Follow(S) = \{\perp\}$,

$Follow(A) = \{a\}$,

$Follow(B) = \{b\}$,

$Follow(C) = \{a, b, c\}$.

Базовым пунктом для исходного состояния I_0 $LALR(1)$-автомата является начальный пункт $[S' \to \bullet S\bot, \varepsilon]$, в котором предпросмотром является пустая строка ε, так как за S' не может следовать никакой символ. Поскольку маркерная точка стоит перед нетерминалом S, а $First(\bot) = \{\bot\}$, необходимо добавить пункт $[S \to \bullet AaBb, \bot]$. Продолжим вычисление замыкания. Поскольку $First(aBb\bot) = \{a\}$, добавляются пункты $[A \to \bullet C, a]$ и $[A \to \bullet db, a]$. Маркерная точка стоит перед нетерминалом C, $First(a) = \{a\}$, поэтому добавляются $[C \to \bullet Ccd, a]$ и $[C \to \bullet d, a]$. Опять маркерная точка перед нетерминалом C, $First(cda) = \{c\}$, добавляются пункты $[C \to \bullet Ccd, c]$ и $[C \to \bullet d, c]$. Больше никакие новые пункты добавить нельзя, операция замыкания для состояния I_0 завершена. Отметим, что в одном состоянии могут быть $LR(1)$-пункты с одинаковыми первыми элементами, но с разными предпросмотрами, например, $[C \to \bullet d, a]$ и $[C \to \bullet d, c]$. Для удобства записи предпросмотры пунктов будем представлять как множества, а пункты с одинаковыми первыми элементами – как один пункт (хотя формально это разные пункты), предпросмотром которого является объединение множеств их предпросмотров, например, пункты $[C \to \bullet d, a]$ и $[C \to \bullet d, c]$ будут записываться как $[C \to \bullet d, \{a, c\}]$.

Состояния-преемники определяются так же, как и в ранее описанных методах, т. е. по первым элементам пунктов. *LALR*(1)-автомат для рассматриваемой грамматики представлен в табл. 4.10 (символы свертки указаны в столбце «символ перехода»), а *LALR*(1)-таблица разбора – в табл. 4.11.

Таблица 4.10

LALR(1)-автомат

Состояние	Пункты	Символ перехода	Состояние-преемник	Свертка
I_0	$S' \to \bullet S\bot, \{\varepsilon\}$	S	*stop*	
	$S \to \bullet AaBb, \{\bot\}$	A	I_1	
	$A \to \bullet C, \{a\}$	C	I_2	
	$C \to \bullet Ccd, \{a, c\}$			
	$A \to \bullet db, \{a\}$	d	I_3	
	$C \to \bullet d, \{a, c\}$			
I_1	$S \to A\bullet aBb, \{\bot\}$	a	I_4	
I_2	$A \to C\bullet, \{a\}$	a		$R2$
	$C \to C\bullet cd, \{a, c\}$	c	I_5	
I_3	$A \to d\bullet b, \{a\}$	b	I_6	
	$C \to d\bullet, \{a, c\}$	a, c		$R6$
I_4	$S \to Aa\bullet Bb, \{\bot\}$	B	I_7	
	$B \to \bullet C, \{b\}$	C	I_8	
	$C \to \bullet Ccd, \{b, c\}$			
	$C \to \bullet d, \{b, c\}$	d	I_9	
I_5	$C \to Cc\bullet d, \{a, b, c\}$	d	I_{10}	
I_6	$A \to db\bullet, \{a\}$	a		$R3$
I_7	$S \to AaB\bullet b, \{\bot\}$	b	I_{11}	
I_8	$B \to C\bullet, \{b\}$	b		$R4$
	$C \to C\bullet cd, \{b, c\}$	c	I_5	
I_9	$C \to d\bullet, \{b, c\}$	b, c		$R6$
I_{10}	$C \to Ccd\bullet, \{a, b, c\}$	a, b, c		$R5$
I_{11}	$S \to AaBb\bullet, \{\bot\}$	$\bot$		$R1$

1) $S \to AaBb$
2) $A \to C$
3) $A \to db$
4) $B \to C$
5) $C \to Ccd$
6) $C \to d$

$Follow(S) = \{\bot\}$,
$Follow(A) = \{a\}$,
$Follow(B) = \{b\}$,
$Follow(C) = \{a, b, c\}$.

Таблица 4.11

LALR(1)-таблица разбора

Номер состояния	*S*	*A*	*B*	*C*	*a*	*b*	*c*	*d*	⊥
0	*stop*	*S*1		*S*2				*S*3	
1					*S*4				
2					*R*2		*S*5		
3					*R*6	*S*6	*R*6		
4			*S*7	*S*8				*S*9	
5								*S*10	
6					*R*3				
7						*S*11			
8						*R*4	*S*5		
9						*R*6	*R*6		
10					*R*5	*R*5	*R*5		
11									*R*1

Следует обратить внимание на то, что для состояния I_2 в соответствии с пунктом $[C \to C\bullet cd, \{a, c\}]$ по символу c образуется преемник с базовым пунктом $[C \to Cc\bullet d, \{a, c\}]$, а для состояния I_8 в соответствии с $[C \to C\bullet cd, \{b, c\}]$ – преемник с базовым пунктом $[C \to Cc\bullet d, \{b, c\}]$. Ядра этих состояний-преемников совпадают, поэтому согласно требованию *LALR*(1)-метода они слиты в одно состояние I_5, а предпросмотры их пунктов объединены в единое множество, т. е. получилось состояние с пунктом $[C \to Cc\bullet d, \{a, b, c\}]$.

Все конфликты для рассматриваемой грамматики успешно разрешены, неадекватных состояний нет, следовательно, она является *LALR*(1)-грамматикой. Эта грамматика не обладает признаком *LR*(0), так как нельзя, например, поместить все элементы свертки *R*2 в каждый столбец для состояния I_2. Не обладает она также и признаком *SLR*(1), так как в противном случае в состоянии I_3, где выполняется свертка для продукции $C \to d$, можно было бы занести элемент свертки *R*6 в столбец, соответствующий терминалу *b*, поскольку $Follow(C) = \{a, b, c\}$. Однако это привело бы к конфликту с уже имеющимся в этой позиции элементом переноса *S*6, т. е. *LALR*(1)-метод устанавливает, что в состоянии I_3 действительными символами-следователями являются только терминалы *a* и *c*, а символ *b* не может быть символом-следователем. По аналогичной причине в состоянии I_9 в столбце, соответствующем терминалу *a*, нет элемента свертки *R*6, поскольку в этом состоянии действительными символами-следователями могут быть только терминалы *b* и *c*.

LALR(1)-метод важен с практической точки зрения, поскольку для стандартных синтаксических конструкций языков программирования там, где не приводит к успеху *SLR*(1)-метод, обычно справляется *LALR*(1)-метод, и достаточно редко приходится применять наиболее общий *LR*(1)-метод.