



## 第6章

# *LR* 分析法

---

# 移进-归约过程

## ■ 单词序列 **aaab** 的分析

文法  $G(S)$  :

(1)  $S \rightarrow AB$

(2)  $A \rightarrow aA$

(3)  $A \rightarrow \varepsilon$

(4)  $B \rightarrow b$

(5)  $B \rightarrow bB$

# 自底向上优先分析与自顶向下技术相比

## – 功能较强大

原因在于**推导**和**归约**过程有如下差别：推导时仅观察可推导出的输入串的一部分，而归约时可归约的输入串整体已全部出现

## – 利于出错处理

输入符号查看后才被移进

## – 构造较复杂

手工构造有难度

但存在很好的自动构造技术  
(如 Yacc 工具采用 LALR 分析技术)

# LR(k)

由左向右地  
处理输入串

规范推导的逆过程

向右看k个符号

如：LR(0)、SLR(1)、LR(1)、LALR(1)

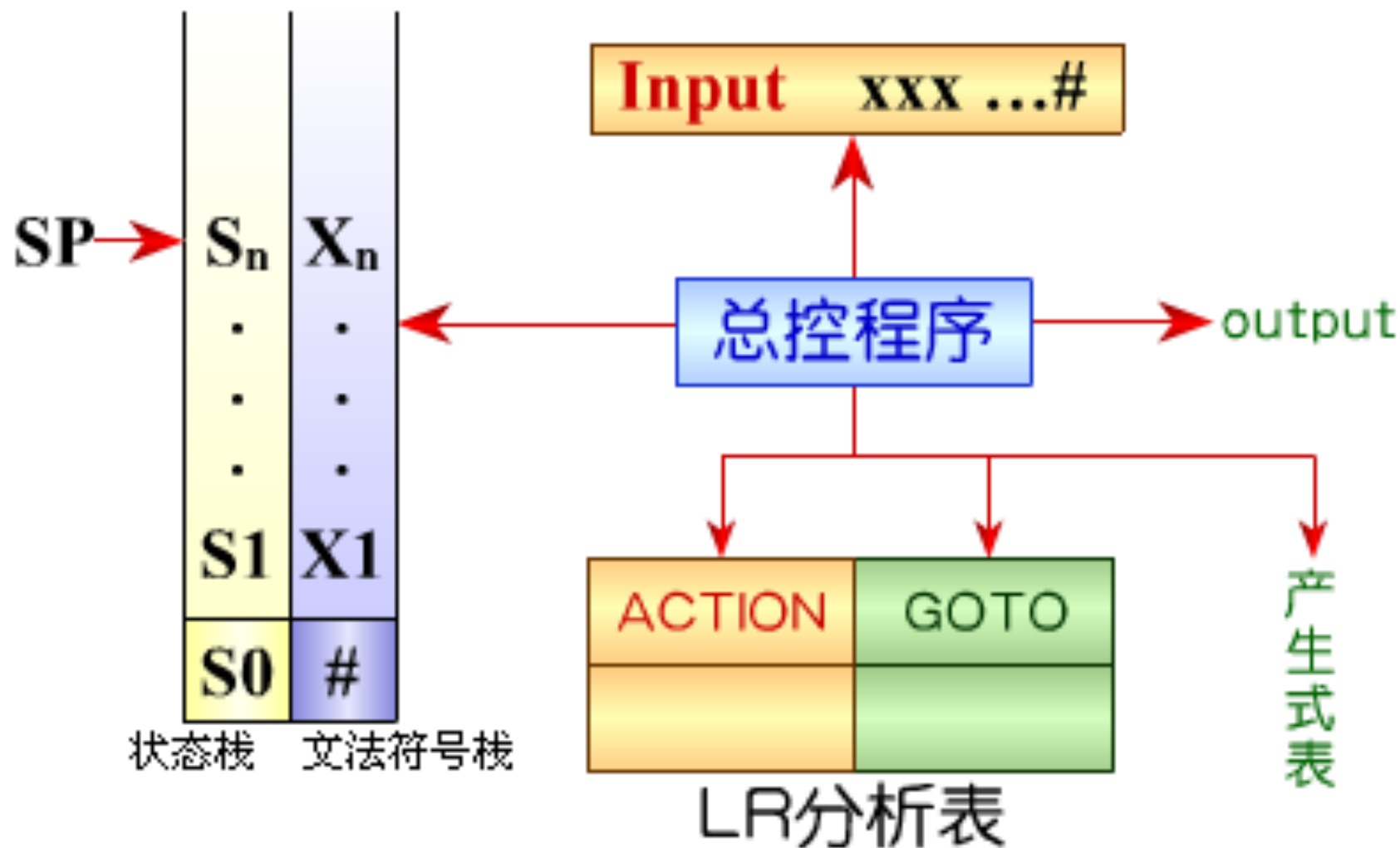
## 6.1 LR分析概述

*LR分析器模型*

*LR分析表*

*LR分析算法*

# LR分析器模型



# LR分析表

	ACTION						GOTO		
	a	c	e	b	d	#	S	A	B
0	$S_2$						1		
1						acc			
2				$S_4$				3	
3		$S_5$		$S_6$					
4	$r_2$	$r_2$	$r_2$	$r_2$	$r_2$	$r_2$			
5					$S_8$				7
6	$r_3$	$r_3$	$r_3$	$r_3$	$r_3$	$r_3$			
7			$S_9$						
8	$r_4$	$r_4$	$r_4$	$r_4$	$r_4$	$r_4$			
9	$r_1$	$r_1$	$r_1$	$r_1$	$r_1$	$r_1$			

**ACTION** [i,e]=
  $\left\{ \begin{array}{l} S_j: \text{移进, } e \text{ 和 } j \text{ 进栈} \\ r_k: \text{用产生式 } (k)A \rightarrow X_{m-r+1}X_{m-r+2}\dots X_m \text{ 规约, 且 } S=\text{GOTO}[S',A] \text{ 进栈} \\ \text{acc: 接受} \\ \text{出错} \end{array} \right.$

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcde#	移进	0		
2)	#a	bbcd#e#	移进	02		
3)	#a <b>b</b>	bcde#	归约( $A \rightarrow b$ )	024		
4)	#aA	bcde#	移进	023		
5)	#a <b>Ab</b>	cde#	归约( $A \rightarrow Ab$ )	0236		
6)	#aA	cde#	移进	023		
7)	#aAc	de#	移进	0235		
8)	#aAc <b>d</b>	e#	归约( $B \rightarrow d$ )	02358		
9)	#aAcB	e#	移进	02357		
10)	#aAc <b>Be</b>	#	归约( $S \rightarrow aAcBe$ )	023579		
11)	#S	#	接受	01		

## 对输入串abbcde#的LR分析过程

## 文法 $G[S]$ :

(1)  $S \rightarrow aAcBe$

**(2)  $A \rightarrow b$**

(3)  $A \rightarrow Ab$

**(4)  $B \rightarrow d$**

[illegible]



# LR分析算法

置ip指向输入串w的第一个符号

令S为栈顶状态

a是ip指向的符号

loop

if ACTION[S,a]= $S_j$  then

PUSH j,a

/\*进栈

ip 前进

/\*指向下一输入符号

else if ACTION[S,a]= $r_k$

/\*第k条产生式为 $A \rightarrow \beta$

pop  $|\beta|$  项

令当前栈顶状态为 $S'$

push GOTO[ $S'$ ,A]和A

/\*进栈

else if ACTION[s,a]=acc

return /\*成功

else

error

end loop

## ■ LR分析法特征

- ❑ 符号栈中的符号是规范句型的前缀且不含句柄以后的任何符号(活前缀)。当它含有句柄时，称之为可规约前缀。
- ❑ 分析决策依据：栈顶状态和现行输入符号
- ❑ 存在识别活前缀的DFA

## 6.2 LR(0)分析

*活前缀*

*识别活前缀的有限自动机*

*LR(0)分析表*

## 定义6.1-活前缀

- 设  $S' \xRightarrow{*}_R \alpha A \omega \xRightarrow{R} \alpha \beta \omega$  是文法  $G$  中的一个规范推导，如果符号串  $\gamma$  是  $\alpha \beta$  的前缀，则称是  $G$  的一个活前缀。
- $S'$  是对原文法扩充 ( $S' \rightarrow S$ ) 增加的非终结符，以使  $S'$  不出现在任何产生式的右部，

# 可归前缀与活前缀

输入串 **abbcde**

文法  $G[S]$ :

(1)  $S' \rightarrow S[0]$

(1)  $S \rightarrow aAcBe[1]$

(2)  $A \rightarrow b[2]$

(3)  $A \rightarrow Ab[3]$

(4)  $B \rightarrow d[4]$

$S' \Rightarrow S[0]$

$\Rightarrow aAcBe[1]$

$\Rightarrow aAcd[4]e[1]$

$\Rightarrow aAb[3]cd[4]e[1]$

$\Rightarrow ab[2]b[3]cd[4]e[1]$

可归前缀

$S[0]$

$ab[2]$

$aAb[3]$

$aAcd[4]$

$aAcBe[1]$

活前缀

$\epsilon, S$

$\epsilon, a, ab$

$\epsilon, a, aA, aAb$

$\epsilon, a, aA, aAc, aAcd$

$\epsilon, a, aA, aAc, aAcB, aAcBe$

## 6. 2. 2识别活前缀的有限自动机

### ■ LR分析需要构造识别活前缀的有穷自动机

- 我们可以文法的终结符和非终结符都看成有穷自动机的输入符号，每次把一个符号进栈看成已识别过了该符号，同时状态进行转换，当识别到可归前缀时，相当于在栈中形成句柄，认为达到了识别句柄的终态。

活前缀

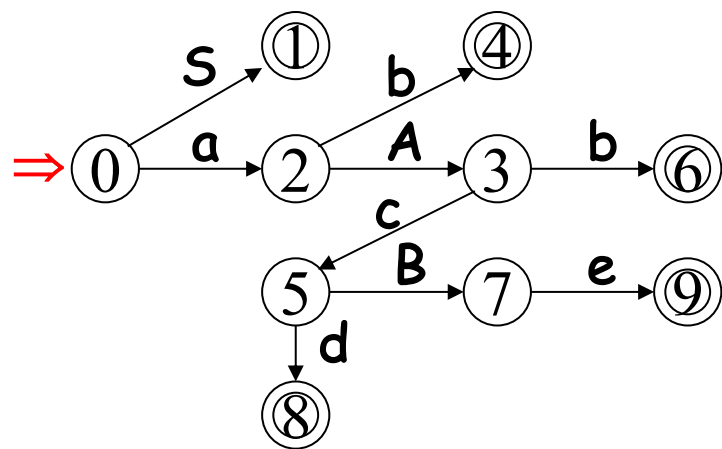
$\epsilon, S$

$\epsilon, a, ab$

$\epsilon, a, aA, aAb$

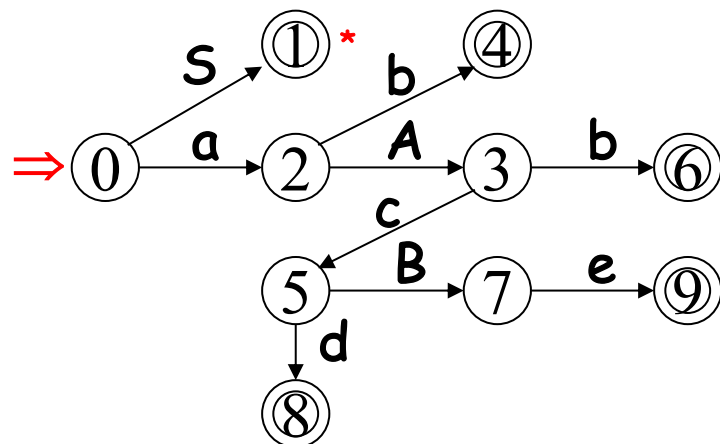
$\epsilon, a, aA, aAc, aAcd$

$\epsilon, a, aA, aAc, aAcB, aAcBe$



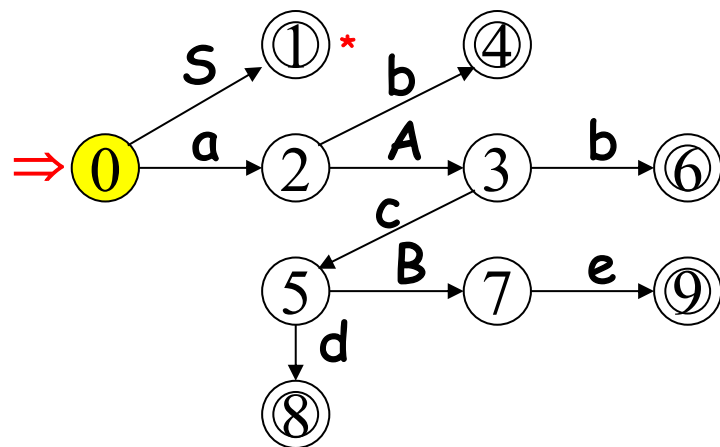
# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进 0	$S_2$		
2)	#a	bbcede#	移进 02	$S_4$		
3)	#ab	bcde#	归约( $A \rightarrow b$ ) 024	$r_2$	3	
4)	#aA	bcde#	移进 023	$S_6$		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ ) 0236	$r_3$	3	
6)	#aA	cde#	移进 023	$S_5$		
7)	#aAc	de#	移进 0235	$S_8$		
8)	#aAcd	e#	归约( $B \rightarrow d$ ) 02358	$r_4$	7	
9)	#aAcB	e#	移进 02357	$S_9$		
10)	#aAcBe	#	归约( $S \rightarrow aAcBe$ ) 023579	$r_1$	1	
11)	#S	#	接受 01	acc		



# 对输入串abbcede的LR分析过程

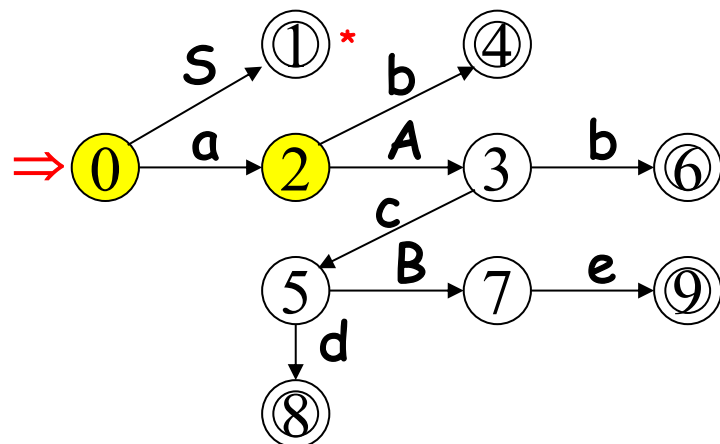
步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进	0		





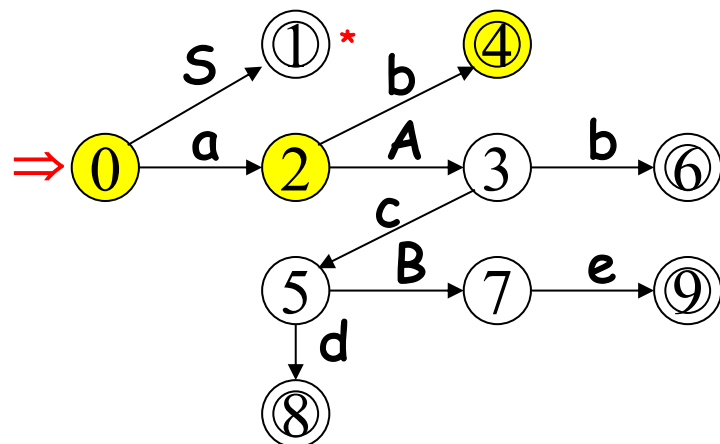
# 对输入串abcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abcde#	移进	0		
2)	#a	bcde#	移进	02		



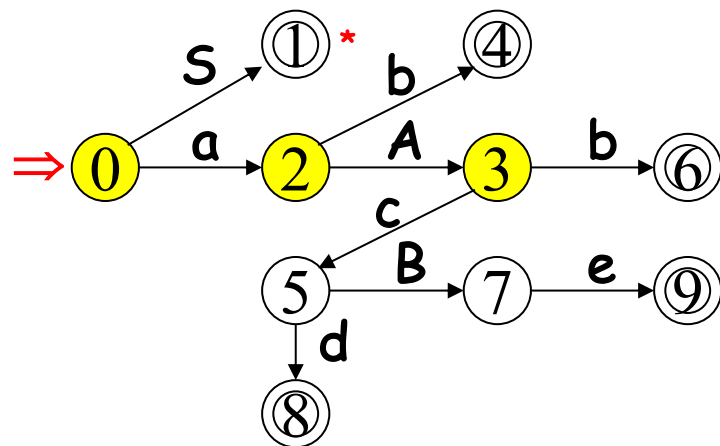
# 对输入串abcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abcde#	移进	0		
2)	#a	bbcde#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024		



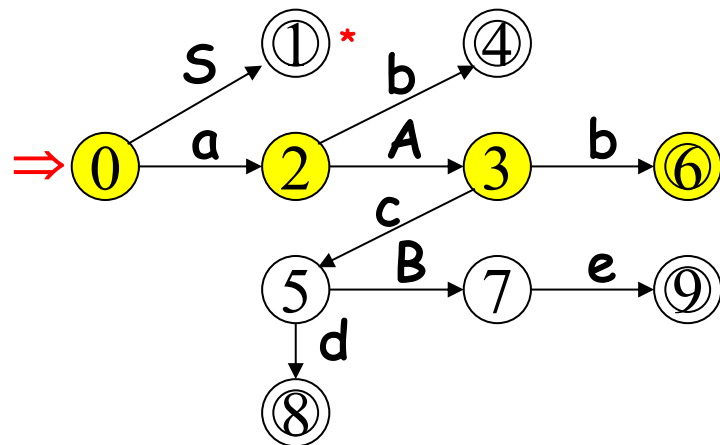
# 对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcde#	移进	0		
2)	#a	bbcde#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024	3	
4)	#aA	bcde#	移进	023		



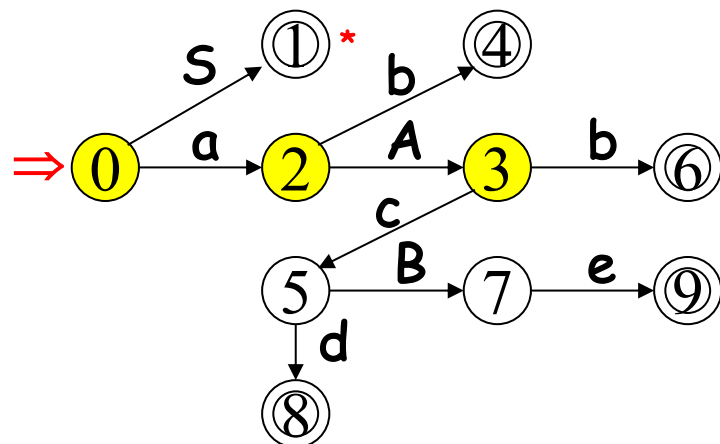
# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进	0		
2)	#a	bbcde#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024	3	
4)	#aA	bcde#	移进	023		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ )	0236	3	



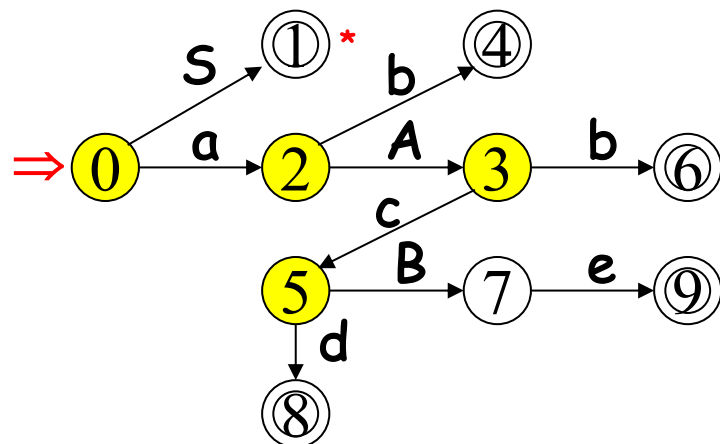
# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进	0		
2)	#a	bbcde#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024	3	
4)	#aA	bcde#	移进	023		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ )	0236	3	
6)	#aA	cde#	移进	023		



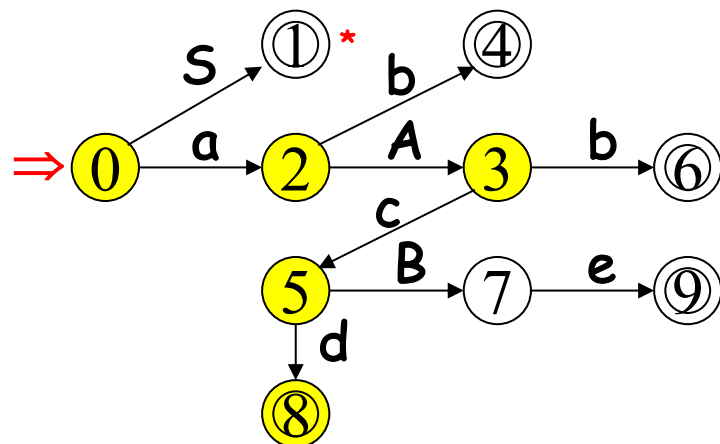
# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进	0		
2)	#a	bbcde#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024	3	
4)	#aA	bcde#	移进	023		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ )	0236	3	
6)	#aA	cde#	移进	023		
7)	#aAc	de#	移进	0235		



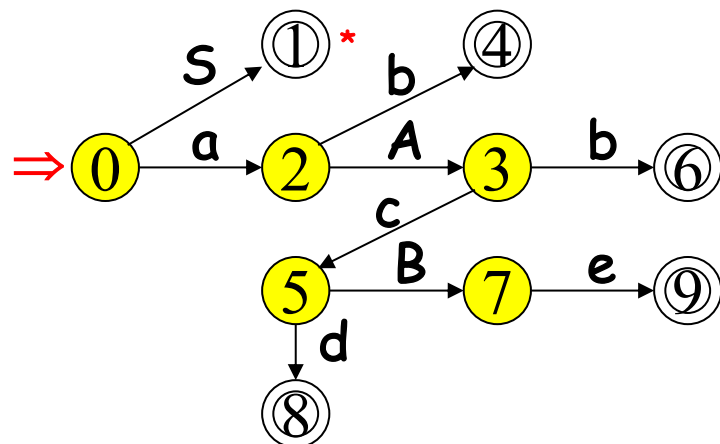
# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进	0		
2)	#a	bbcede#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024	3	
4)	#aA	bcde#	移进	023		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ )	0236	3	
6)	#aA	cde#	移进	023		
7)	#aAc	de#	移进	0235		
8)	#aAcd	e#	归约( $B \rightarrow d$ )	02358	7	



# 对输入串abbcede的LR分析过程

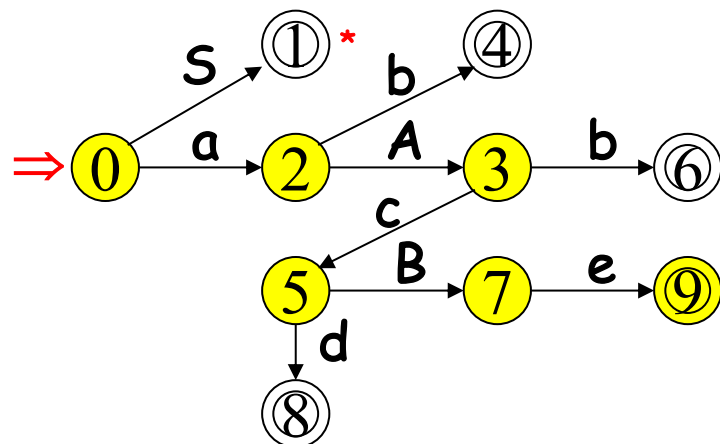
步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进	0		
2)	#a	bbcede#	移进	02		
3)	#ab	bcde#	归约( $A \rightarrow b$ )	024	3	
4)	#aA	bcde#	移进	023		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ )	0236	3	
6)	#aA	cde#	移进	023		
7)	#aAc	de#	移进	0235		
8)	#aAcd	e#	归约( $B \rightarrow d$ )	02358	7	
9)	#aAcB	e#	移进	02357		





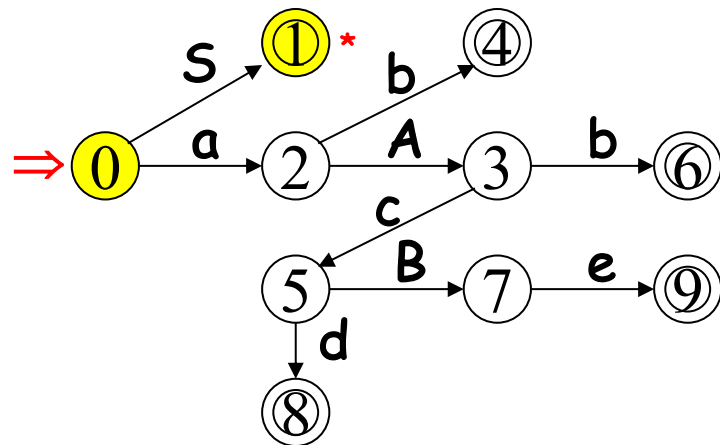
# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进 0	$S_2$		
2)	#a	bbcede#	移进 02	$S_4$		
3)	#ab	bcde#	归约( $A \rightarrow b$ ) 024	$r_2$	3	
4)	#aA	bcde#	移进 023	$S_6$		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ ) 0236	$r_3$	3	
6)	#aA	cde#	移进 023	$S_5$		
7)	#aAc	de#	移进 0235	$S_8$		
8)	#aAcd	e#	归约( $B \rightarrow d$ ) 02358	$r_4$	7	
9)	#aAcB	e#	移进 02357	$S_9$		
10)	#aAcBe	#	归约( $S \rightarrow aAcBe$ ) 023579	$r_1$		



# 对输入串abbcede的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcede#	移进 0	$S_2$		
2)	#a	bbcede#	移进 02	$S_4$		
3)	#ab	bcde#	归约( $A \rightarrow b$ ) 024	$r_2$	3	
4)	#aA	bcde#	移进 023	$S_6$		
5)	#aAb	cde#	归约( $A \rightarrow Ab$ ) 0236	$r_3$	3	
6)	#aA	cde#	移进 023	$S_5$		
7)	#aAc	de#	移进 0235	$S_8$		
8)	#aAcd	e#	归约( $B \rightarrow d$ ) 02358	$r_4$	7	
9)	#aAcB	e#	移进 02357	$S_9$		
10)	#aAcBe	#	归约( $S \rightarrow aAcBe$ ) 023579	$r_1$	1	
11)	#S	#	接受 01	acc		



# 构造识别可归前缀的有限自动机

## ■ 活前缀和句柄的关系

□ 活前缀 $\alpha$ 不含有句柄的任何符号

可记作:  $A \rightarrow \cdot \beta$

□ 活前缀 $\alpha\beta_1$ 含有句柄的部分符号

可记作:  $A \rightarrow \beta_1 \cdot \beta_2$

□ 活前缀 $\alpha\beta$ 已含有句柄的全部符号

可记作:  $A \rightarrow \beta \cdot$

- 文法的每一个产生式  $A \rightarrow XY$  定义有下面三个项目：

$A \rightarrow \bullet XY$

$A \rightarrow X \bullet Y$

$A \rightarrow XY \bullet$

对于  $A \rightarrow \varepsilon$  的项目只有  $A \rightarrow \bullet$

以上项目称作 **LR(0)项目**。

■ 文法  $G'[S']$ :

(0)  $S' \rightarrow S$

(1)  $S \rightarrow aAcBe$

(2)  $A \rightarrow b$

(3)  $A \rightarrow Ab$

(4)  $B \rightarrow d$

$S' \rightarrow \bullet S$

$S' \rightarrow S \bullet$

$S \rightarrow \bullet aAcBe$

$S \rightarrow a \bullet AcBe$

$S \rightarrow aA \bullet cBe$

$S \rightarrow aAc \bullet Be$

$S \rightarrow aAcB \bullet e$

$S \rightarrow aAcBe \bullet$

$A \rightarrow \bullet b$

$A \rightarrow b \bullet$

$A \rightarrow \bullet Ab$

$A \rightarrow A \bullet b$

$A \rightarrow Ab \bullet$

$B \rightarrow \bullet d$

$B \rightarrow d \bullet$

- **LR(0)**项目分为以下几种:

设**a**是终结符, **B**是非终结符,  $\alpha, \beta \in V^*$

**移进**项目, 形如  $\mathbf{A} \rightarrow \alpha \bullet \mathbf{a} \beta$

**待约**项目, 形如  $\mathbf{A} \rightarrow \alpha \bullet \mathbf{B} \beta$

**归约**项目, 形如  $\mathbf{A} \rightarrow \alpha \bullet$  或  $\mathbf{A} \rightarrow \bullet$

**接受**项目, 形如  $\mathbf{S}' \rightarrow \mathbf{S} \bullet$

## ■ 识别活前缀的DFA

**DFA = (K = {项目集规范族},  
 $\Sigma = V_T \cup V_N \cup \{S'\},$   
 $M = \{Go(I, X)\},$   
 $S = \text{Closure}\{S' \rightarrow \cdot S\},$   
 $Z = \{\text{项目集规范族}\})$**

## ■ 项目集闭包

设 $I$ 是文法 $G$ 的一个LR(0)项目集合, 则  
**Closure( $I$ )**是用下面三条规则构造的项目集:

1.  $I$ 中每一个项目都属于**Closure( $I$ )**;
2. 若项目 $A \rightarrow \alpha \cdot B \beta \in \text{Closure}(I)$ 且 $B \rightarrow \eta \in P$ , 则将 $B \rightarrow \cdot \eta$ 加进**Closure( $I$ )**中;
3. 重复执行(2)直到**Closure( $I$ )**不再增大为止。

$I_0 = \text{Closure}(S' \rightarrow \cdot S):$

$S' \rightarrow \cdot S$

$S \rightarrow \cdot aAcBe$

$I_1 = \text{Closure}(S \rightarrow a \cdot AcBe):$

$S \rightarrow a \cdot AcBe$

$A \rightarrow \cdot b$

$A \rightarrow \cdot Ab$



## ■ 转移函数

若 $I$ 是文法的一个LR(0)项目集,  $X \in \{V_T \cup V_N\}$ , 则定义转移函数 $Go(I, X)$ 为:

**$Go(I, X) = \text{Closure}(J)$ ,**

**$J = \{A \rightarrow \alpha X \cdot \beta \mid A \rightarrow \alpha \cdot X \beta \in I\}$**

$I_0 = \text{Closure}(S' \rightarrow \bullet S):$

$S' \rightarrow \bullet S$

$S \rightarrow \bullet aAcBe$

a

$I_1 = \text{Closure}(S \rightarrow a \bullet AcBe):$

$S \rightarrow a \bullet AcBe$

$A \rightarrow \bullet b$

$A \rightarrow \bullet Ab$

$Go(I_0, a) = I_1$

## ■ 项目集规范族

对于构成识别一个文法活前缀的**DFA**项目集的全体称为这个文法的**LR(0)**项目集规范族。

```
PROCEDURE  items(G ');
BEGIN
    C:={Closure( S' → ·S) };
    REPEAT
        FOR C中每一项目集I和每一文法符号x DO
            IF GO(I,x) 非空且不属于C THEN
                把 GO(I,x) 放入C中;
    UNTIL  C不再增大;
END;
```

$G[E]$ :

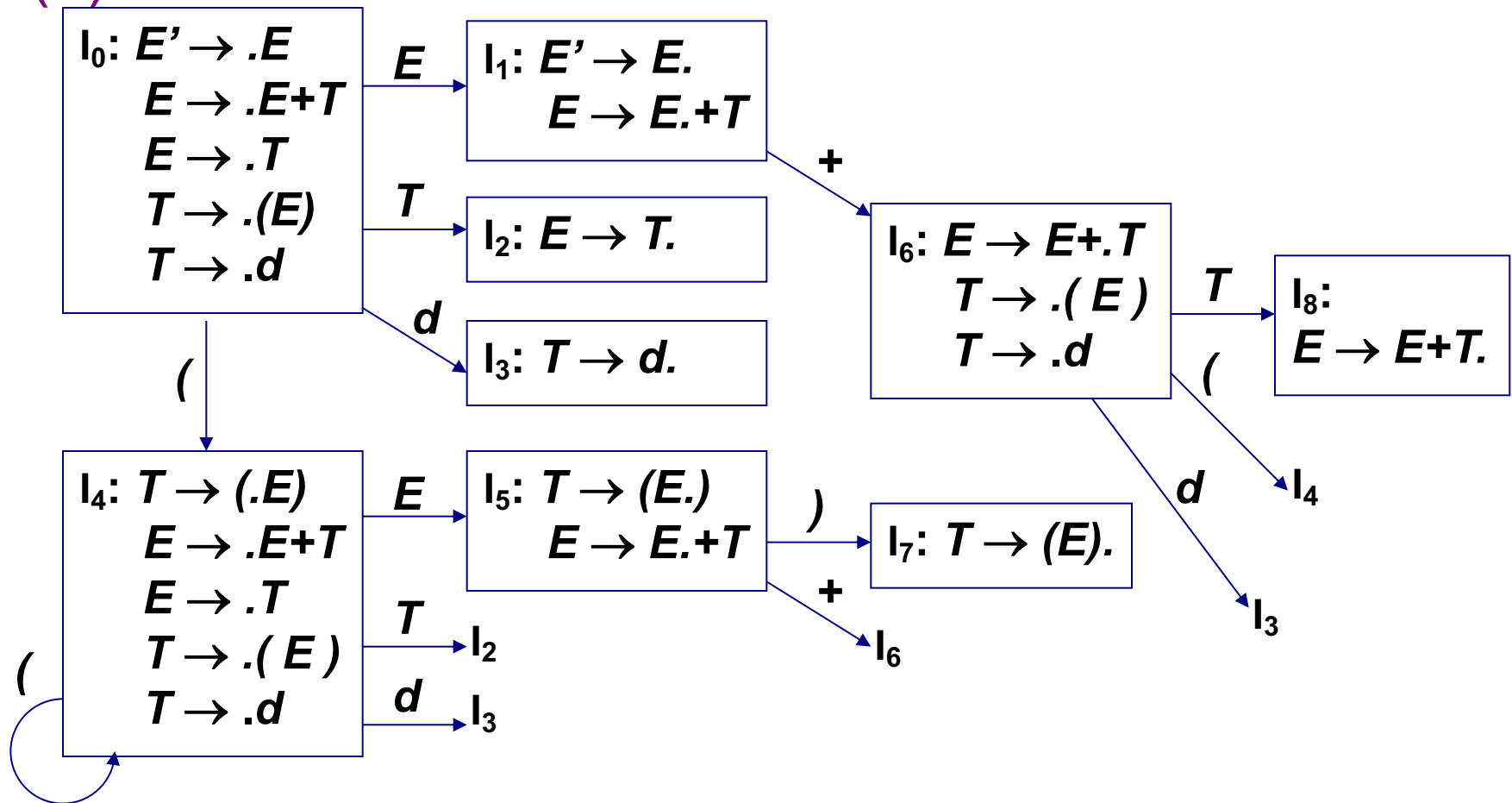
(1)  $E \rightarrow E+T$

(2)  $E \rightarrow T$

(3)  $T \rightarrow (E)$

(4)  $T \rightarrow d$

$E' \rightarrow E$



## ■ 例

**$G'[S']:$**

**$(0) S' \rightarrow S$**

**$(1) S \rightarrow aAcBe$**

**$(2) A \rightarrow b$**

**$(3) A \rightarrow Ab$**

**$(4) B \rightarrow d$**

# LR(0)分析表

- 若项目  $A \rightarrow \alpha \cdot a \beta$  属于  $I_k$  且  $GO(I_k, a) = I_j$ ,  $a$  为终结符, 则置  $ACTION[k, a]$  为 “把状态  $j$  和符号  $a$  移进栈”, 简记为 “ $s_j$ ”;
- 若项目  $A \rightarrow \alpha \cdot$  属于  $I_k$ , 那么, 对任何终结符  $a$ , 置  $ACTION[k, a]$  为 “用产生式  $A \rightarrow \alpha$  (文法  $G'$  的第  $j$  个产生式) 进行规约”, 简记为 “ $r_j$ ”;
- 若项目  $S' \rightarrow S \cdot$  属于  $I_k$ , 则置  $ACTION[k, \#]$  为 “接受”, 简记为 “ $acc$ ”;
- 若  $GO(I_k, A) = I_j$ ,  $A$  为非终结符, 则置  $GOTO(k, A) = j$ ;
- 分析表中凡不能用上述规则填入信息的空白格均置上 “出错标志”。

## 对输入串abbcde#的LR分析过程

[illegible]

## ■ LR(0)文法

对于一个上下文无关文法，如果能够构造一张LR(0)分析表，使得它的每一个条目均是唯一的，则称该上下文无关文法为LR(0)文法。

## ■ 练习

**$G[S']$ :**

**(0)  $S' \rightarrow E$**

**(1)  $E \rightarrow aA$**

**(2)  $E \rightarrow bB$**

**(3)  $A \rightarrow cA$**

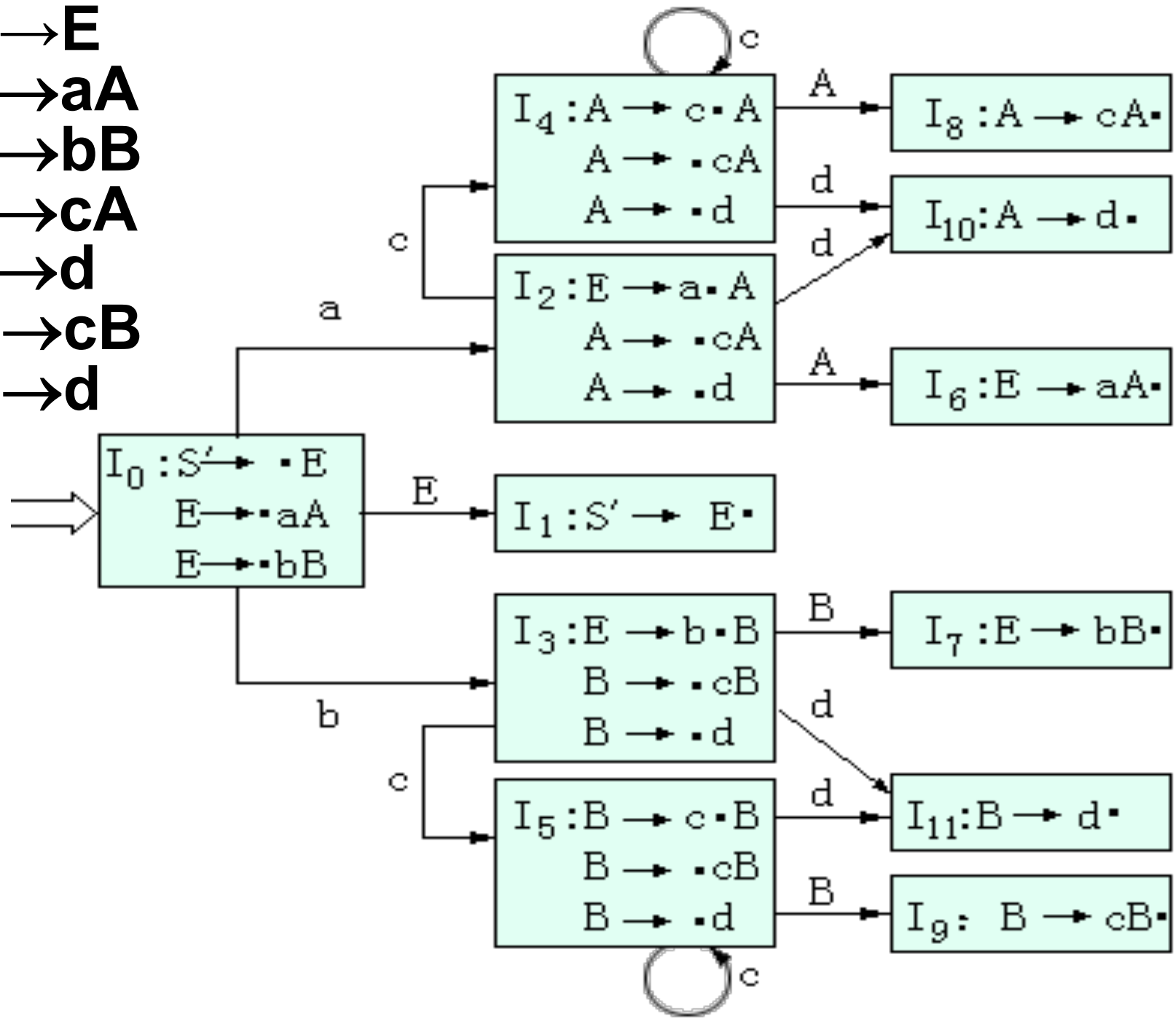
**(4)  $A \rightarrow d$**

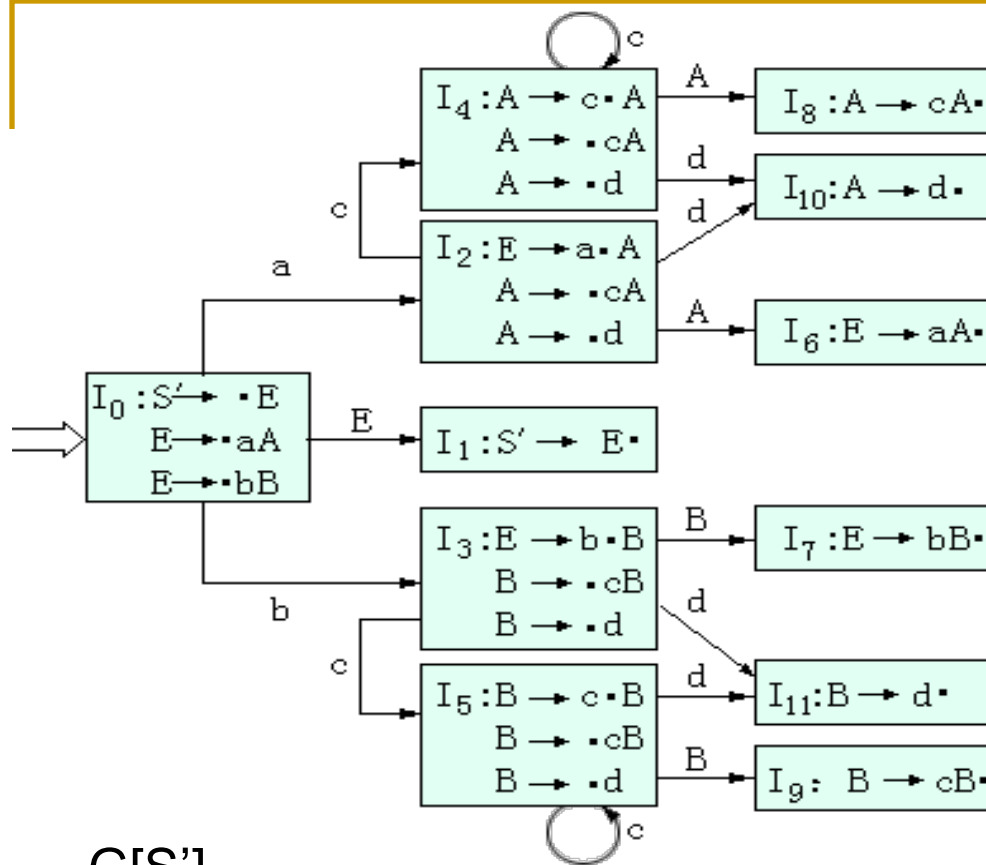
**(5)  $B \rightarrow cB$**

**(6)  $B \rightarrow d$**



- (0)  $S' \rightarrow E$
- (1)  $E \rightarrow aA$
- (2)  $E \rightarrow bB$
- (3)  $A \rightarrow cA$
- (4)  $A \rightarrow d$
- (5)  $B \rightarrow cB$
- (6)  $B \rightarrow d$





$G[S']$ :

(0)  $S' \rightarrow E$

(1)  $E \rightarrow aA$

(2)  $E \rightarrow bB$

(3)  $A \rightarrow cA$

(4)  $A \rightarrow d$

(5)  $B \rightarrow cB$

(6)  $B \rightarrow d$

	ACTION					GOTO		
	a	b	c	d	#	E	A	B
0	S2	S3				1		
1					acc			
2			S4	S10			6	
3			S5	S11				7
4			S4	S10			8	
5			S5	S11				9
6	r1	r1	r1	r1	r1			
7	r2	r2	r2	r2	r2			
8	r3	r3	r3	r3	r3			
9	r5	r5	r5	r5	r5			
10	r4	r4	r4	r4	r4			
11	r6	r6	r6	r6	r6			

# 引例

$G[S']$ :

(0)  $S' \rightarrow E$

(1)  $E \rightarrow E+T$

(2)  $E \rightarrow T$

(3)  $T \rightarrow T * F$

(4)  $T \rightarrow F$

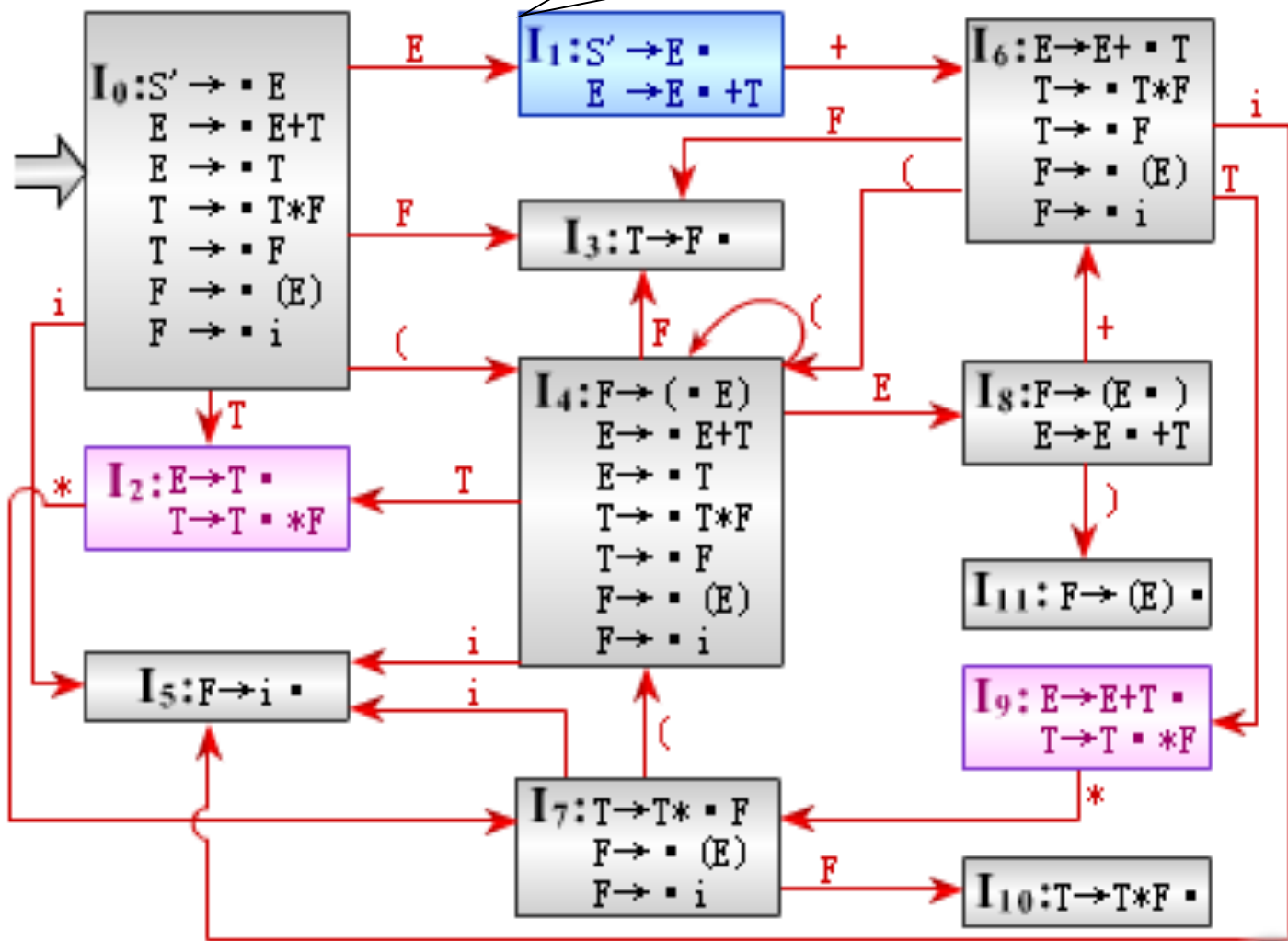
(5)  $F \rightarrow (E)$

(6)  $F \rightarrow i$

向前查看一个符号，看其是否是 $S'$  的后继符号( $FOLLOW(S')$ )。

若否，则移进；

若是，则归约。



## 6.3 SLR(1)分析

- 一个LR(0)规范族中含有如下的项目集(状态)

$$I = \{X \rightarrow \alpha \cdot b\beta, A \rightarrow \gamma \cdot, B \rightarrow \delta \cdot\}$$

若有  $\text{FOLLOW}(A) \cap \text{FOLLOW}(B) = \emptyset$

$$\text{FOLLOW}(A) \cap \{b\} = \emptyset$$

$$\text{FOLLOW}(B) \cap \{b\} = \emptyset$$

则当状态I面临输入符号a时,

- 1) 若  $a=b$ , 则移进;
  - 2) 若  $a \in \text{FOLLOW}(A)$ , 则用产生式  $A \rightarrow \gamma$  进行归约;
  - 3) 若  $a \in \text{FOLLOW}(B)$ , 则用产生式  $B \rightarrow \delta$  进行归约;
  - 4) 此外, 报错。
- 若一个文法的LR(0)分析表中所含有的动作冲突都能用上述方法解决, 则称这个文法是SLR(1)文法。

## ■ SLR(1)分析表的构造步骤:

- 若项目  $A \rightarrow \alpha \bullet a \beta$  属于  $I_k$ ，且转换函数  $GO(I_k, a) = I_j$ ，当  $a$  为终结符时，则置  $ACTION[k, a]$  为  $S_j$
- 若项目  $A \rightarrow \alpha \bullet$  属于  $I_k$ ，则对  $a$  为任何终结符或 ‘#’，且满足  $a \in FOLLOW(A)$  时，置  $ACTION[k, a] = r_j$ ， $j$  为产生式在文法  $G'$  中的编号
- 若  $GO(I_k, A) = I_j$ ，则置  $GOTO[k, A] = j$ ，其中  $A$  为非终结符， $j$  为某一状态号
- 若项目  $S' \rightarrow S \bullet$  属于  $I_k$ ，则置  $ACTION[k, \#] = acc$
- 其它填上“报错标志”

(0)  $S' \rightarrow E$

(1)  $E \rightarrow E+T$

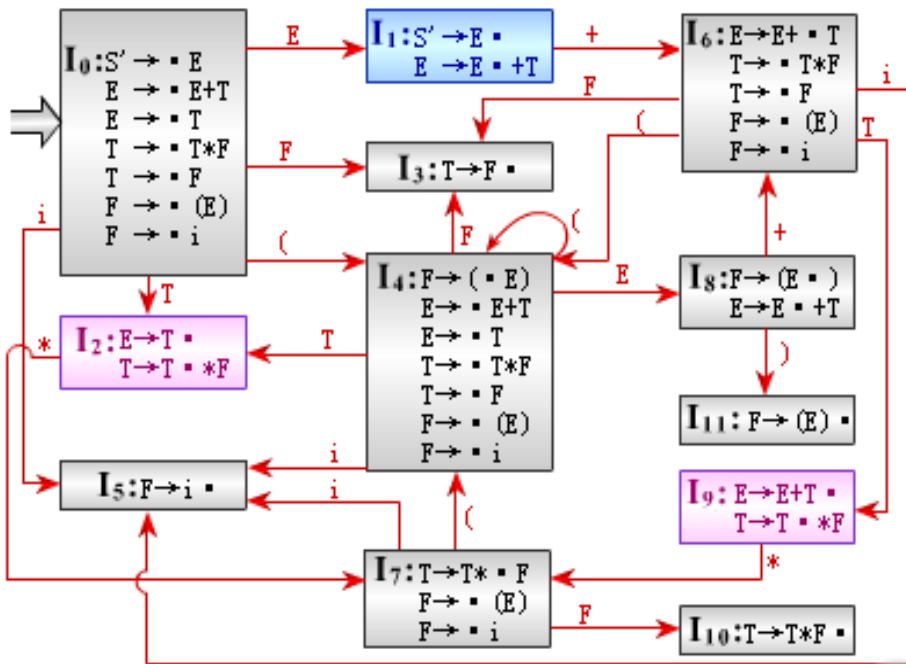
(2)  $E \rightarrow T$

(3)  $T \rightarrow T * F$

(4)  $T \rightarrow F$

(5)  $F \rightarrow (E)$

(6)  $F \rightarrow i$



状态	ACTION						GOTO		
	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

例文法G为:

(1)  $S \rightarrow aAd$

(2)  $S \rightarrow bAc$

(3)  $S \rightarrow aec$

(4)  $S \rightarrow bed$

(5)  $A \rightarrow e$

不是所有的文法都是SLR(1)的。

为确定一个句型的句柄, 引进

LR(1)项目:

$[U \rightarrow x.y, a]$

$a$ 为向前搜索符, 它只对归约项目  $[U \rightarrow x., a]$  有意义, 表示当它所属状态处于栈顶, 且下一个输入符号为  $a$  时, 才可以将栈顶符号串  $x$  归约为  $U$ 。在其它项目中,  $a$  只是起到传递作用。

- 将这些状态合并后得到的新状态若没有冲突出现，则按新状态构造分析表。这就是**LALR(1)分析法**。

## 同芯状态合并

$E \rightarrow (.L, E) ,$	$, / ) / \#$
$F \rightarrow (.F) ,$	$, / ) / \#$
$L \rightarrow .L, E ,$	$,$
$L \rightarrow .E ,$	$,$
$F \rightarrow .(F) ,$	$, / )$
$F \rightarrow .d ,$	$, / )$
$E \rightarrow .(L, E) ,$	$,$
$E \rightarrow .F ,$	$,$

$I_3: E \rightarrow (.L, E) , \#$   
 $F \rightarrow (.F) , \#$   
 $L \rightarrow .L, E , ,$   
 $L \rightarrow .E , ,$   
 $F \rightarrow .(F) , , / )$   
 $F \rightarrow .d , , / )$   
 $E \rightarrow .(L, E) , ,$   
 $E \rightarrow .F , ,$

$I_8: E \rightarrow (.L, E) , ,$   
 $F \rightarrow (.F) , , / )$   
 $L \rightarrow .L, E , ,$   
 $L \rightarrow .E , ,$   
 $F \rightarrow .(F) , , / )$   
 $F \rightarrow .d , , / )$   
 $E \rightarrow .(L, E) , ,$   
 $E \rightarrow .F , ,$

$I_{13}: E \rightarrow (.L, E) , , / )$   
 $F \rightarrow (.F) , , / )$   
 $L \rightarrow .L, E , ,$   
 $L \rightarrow .E , ,$   
 $F \rightarrow .(F) , , / )$   
 $F \rightarrow .d , , / )$   
 $E \rightarrow .(L, E) , ,$   
 $E \rightarrow .F , ,$



## 6.6 二义性文法在LR分析中的应用

### ■ 例

**G'[E']:**

**(0)  $E' \rightarrow E$**

**(1)  $E \rightarrow E + E$**

**(2)  $E \rightarrow E * E$**

**(3)  $E \rightarrow (E)$**

**(4)  $E \rightarrow \text{id}$**

$I_0:$   $E' \rightarrow \cdot E$   
 $E \rightarrow \cdot E + E$   
 $E \rightarrow \cdot E * E$   
 $E \rightarrow \cdot (E)$   
 $E \rightarrow \cdot \text{id}$

$I_1:$   $E' \rightarrow E \cdot$   
 $E \rightarrow E \cdot + E$   
 $E \rightarrow E \cdot * E$

$I_2:$   $E \rightarrow (\cdot E)$   
 $E \rightarrow \cdot E + E$   
 $E \rightarrow \cdot E * E$   
 $E \rightarrow \cdot (E)$   
 $E \rightarrow \cdot \text{id}$

$I_3:$   $E \rightarrow \text{id} \cdot$

$I_4:$   $E \rightarrow E + \cdot E$   
 $E \rightarrow \cdot E + E$   
 $E \rightarrow \cdot E * E$   
 $E \rightarrow \cdot (E)$   
 $E \rightarrow \cdot \text{id}$

$I_5:$   $E \rightarrow E * \cdot E$   
 $E \rightarrow \cdot E + E$   
 $E \rightarrow \cdot E * E$   
 $E \rightarrow \cdot (E)$   
 $E \rightarrow \cdot \text{id}$

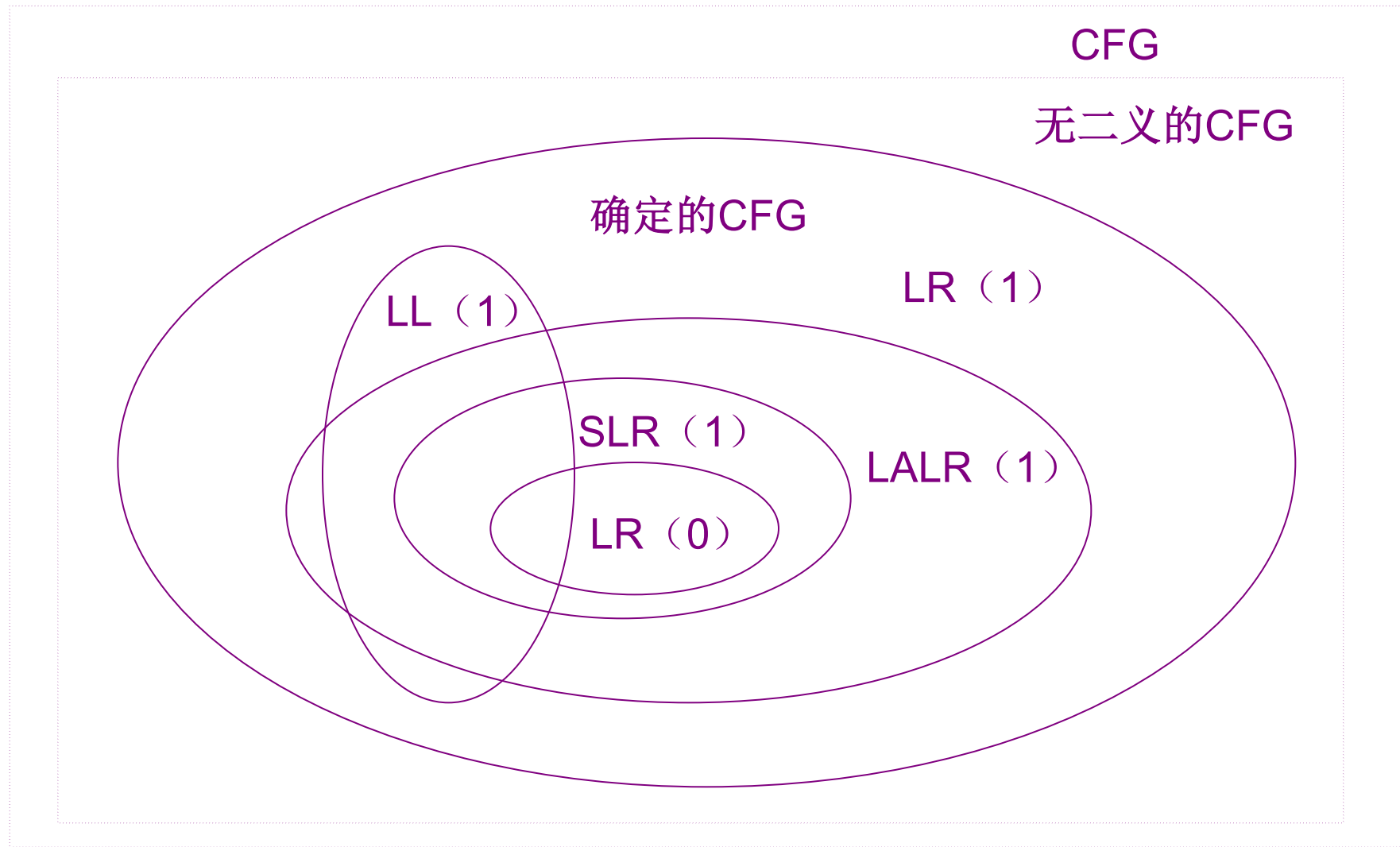
$I_6:$   $E \rightarrow (E \cdot)$   
 $E \rightarrow E \cdot + E$   
 $E \rightarrow E \cdot * E$

$I_7:$   $E \rightarrow E + E \cdot$   
 $E \rightarrow E \cdot + E$   
 $E \rightarrow E \cdot * E$

$I_8:$   $E \rightarrow E * E \cdot$   
 $E \rightarrow E \cdot + E$   
 $E \rightarrow E \cdot * E$

$I_9:$   $E \rightarrow (E) \cdot$

# 几类分析文法之间的关系



# 练习

- 文法**D[S]**为:
- (1)  **$S \rightarrow AB$**
- (2)  **$A \rightarrow aBa | \varepsilon$**
- (3)  **$B \rightarrow bAb | \varepsilon$**
- 1.该文法是**SLR(1)**的吗?
- 2.若是构造它的分析表;
- 3.给出输入串**baab#**的分析过程。

设有拓广文法  $G[S']$ :

(0)  $S' \rightarrow S$

(1)  $S \rightarrow SbAb$

(2)  $S \rightarrow A$

(3)  $A \rightarrow a$

(4)  $A \rightarrow \varepsilon$

- 1.该文法是**SLR(1)**的吗?
- 2.若是构造它的分析表;
- 3.给出输入串**bb#**的分析过程。