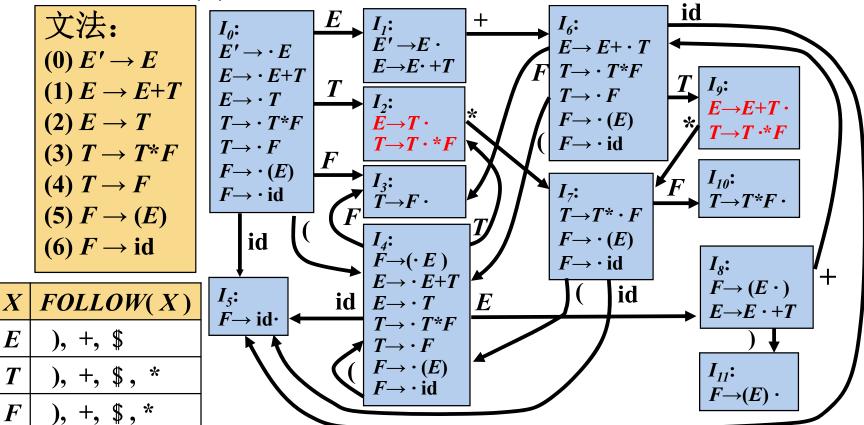
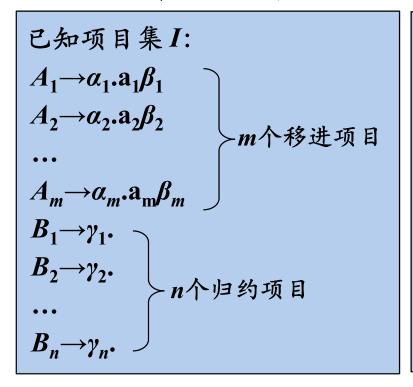


### 例: LR(0) 分析过程中的冲突



# SLR 分析

### >SLR分析法的基本思想

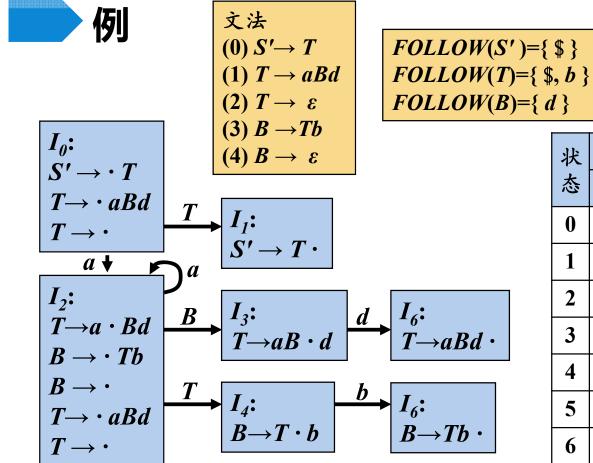


如果集合 $\{a_1, a_2, ..., a_m\}$ 和  $FOLLOW(B_1)$ ,  $FOLLOW(B_2)$ , ...,  $FOLLOW(B_n)$ 两两不相交,则项目 集I中的冲突可以按以下原则解决: 设a是下一个输入符号  $\Rightarrow$  若a  $\in$   $\{a_1, a_2, ..., a_m\}$ , 则移进a

- rackreta 若 $a \in FOLLOW(B_i)$ ,则用产生式  $B_i \rightarrow \gamma_i$  归约
- >此外,报错

# 表达式文法的SLR分析表

状态	ACTION						GOTO		
	id	+	*	(	)	\$	E	T	F
0	s <b>5</b>			<b>s4</b>			1	2	3
1		<b>s6</b>				acc			
2		r2	<b>s</b> 7		r2	r2			
3		r4	r4		r4	r4			
4	<b>s</b> 5			s4			8	2	3
5		r6	r6		r6	r6			
6	<b>s</b> 5			<b>s4</b>				9	3
7	s <b>5</b>			s4					10
8		<b>s6</b>			s11				
9		r1	<b>s</b> 7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			



### SLR(1)分析表

状态		AC'	GOTO			
	a	b	d	\$	T	В
0	<b>s2</b>	r2		r2	1	
1				acc		
2	<b>s2</b>	r2	r4	r2	4	3
3			<b>s</b> 5			
4		<b>s6</b>				
5		r1		r1		
6			r3			

### SLR 分析表构造算法

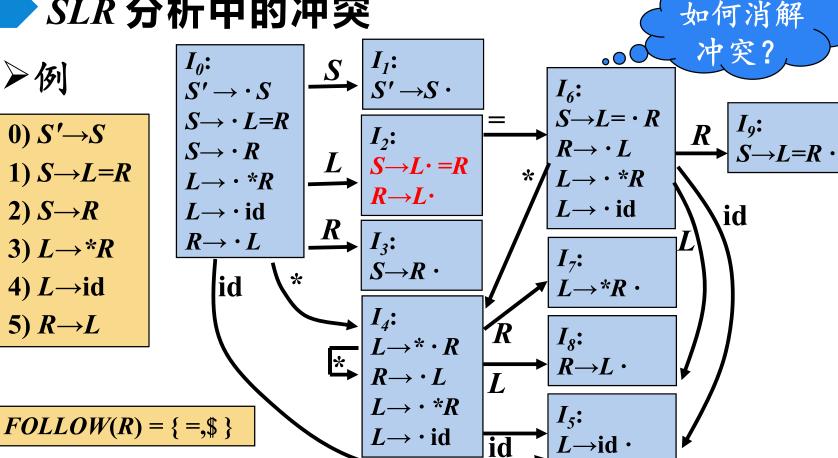
- $\triangleright$  构造G'的规范LR(0)项集族 $C = \{I_0, I_1, \dots, I_n\}$ 。
- ▶根据I<sub>i</sub>构造得到状态i。状态i的语法分析动作按照下面的方法决定:
  - $> if A \rightarrow \alpha \cdot a\beta \in I_i$  and  $GOTO(I_i, a) = I_i$  then ACTION[i, a] = sj
  - $\triangleright if A \rightarrow \alpha.B\beta \in I_i$  and  $GOTO(I_i, B) = I_i$  then GOTO[i, B] = j
  - $ightharpoonup if A 
    ightharpoonup \alpha \cdot \in I_i$ 且 $A \neq S'$  then for  $\forall a \in FOLLOW(A)$  do ACTION[i, a] = rj (j是产生式 $A 
    ightharpoonup \alpha$ 的编号)
  - $\triangleright if S' \rightarrow S \in I_i then ACTION[i, $]=acc;$
- ▶ 没有定义的所有条目都设置为"error"。

如果给定文法的SLR分析表中不存在有冲突的动作,那么该文法称为SLR文法

### SLR 分析中的冲突



- $0) S' \rightarrow S$
- 1)  $S \rightarrow L = R$
- 2)  $S \rightarrow R$
- 3)  $L \rightarrow *R$
- 4)  $L \rightarrow id$
- 5)  $R \rightarrow L$







### LR(1)分析法的提出

- ▶SLR分析存在的问题
  - $\triangleright$ SLR只是简单地考察下一个输入符号b是否属于与归约项目 $A\rightarrow \alpha$ 相关联的FOLLOW(A),但 $b\in FOLLOW(A)$  只是归约 $\alpha$ 的一个必要条件,而非充分条件

### LR(1)分析法的提出

ho对于产生式A
ightarrow lpha的归约,在不同的使用位置,A会要求不同的后继符号

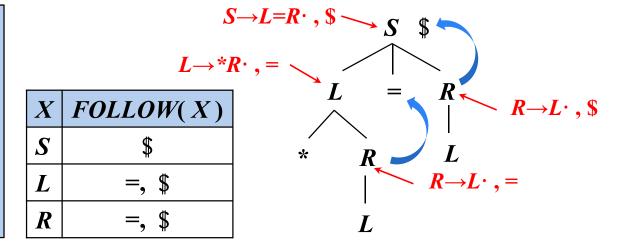
0) <i>S'</i> → <i>S</i>	
1) $S \rightarrow L = R$	,

2) 
$$S \rightarrow R$$

3) 
$$L \rightarrow *R$$

4) 
$$L \rightarrow id$$

5) 
$$R \rightarrow L$$



 $\triangleright$ 在特定位置,A的后继符集合是FOLLOW(A)的子集

### 规范LR(1)项目

- 》将一般形式为  $[A \rightarrow \alpha \cdot \beta, a]$ 的项称为 LR(1) 项,其中 $A \rightarrow \alpha \beta$  是一个产生式,a 是一个终结符(这里将\$视为一个特殊的终结符)它表示在当前状态下,A后面必须紧跟的终结符,称为该项的展望符(lookahead)
  - ► LR(1) 中的1指的是项的第二个分量的长度
  - ightharpoonup 在形如[A 
    ightharpoonup lpha 
    ho eta, a]且 $eta \neq \epsilon$ 的项中,展望符a没有任何作用
  - $\triangleright$ 但是一个形如[ $A\rightarrow\alpha$ , a]的项在只有在下一个输入符号等于a时才可以按照 $A\rightarrow\alpha$  进行归约
    - ▶这样的a的集合总是FOLLOW(A)的子集,而且它通常是一个真子集

## 等价*LR*(1)项目

$$[A \rightarrow \alpha \cdot B\beta, \mathbf{a}]$$

$$B \rightarrow \gamma \in P$$
?

### 等价LR(1)项目

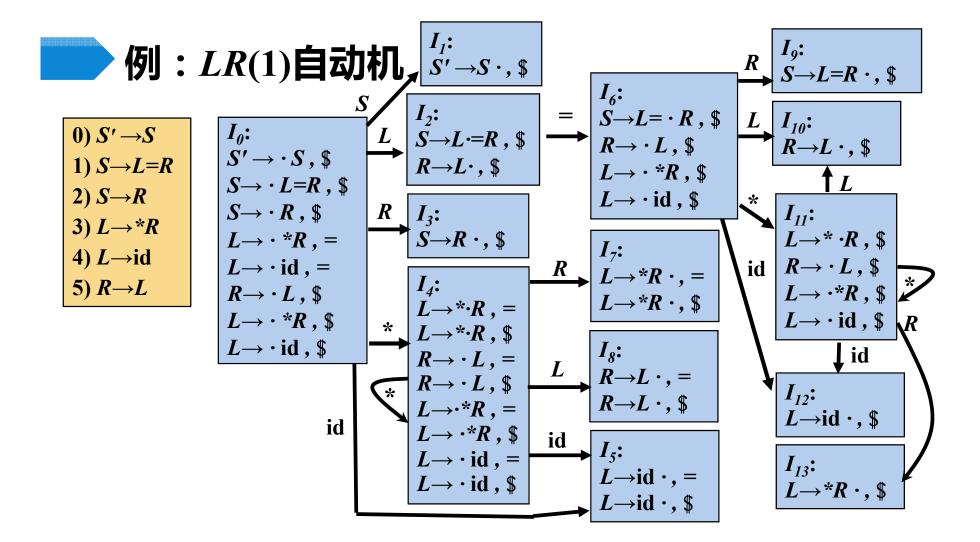
$$[A \rightarrow \alpha \cdot B\beta, \mathbf{a}]$$

$$B \rightarrow \gamma \in P$$

$$[B \rightarrow \gamma, b]$$

$$b \in FIRST(\beta \mathbf{a})$$

当 $\beta \Rightarrow^+ \varepsilon$ 时,此时b=a叫继承的后继符,否则叫自生的后继符



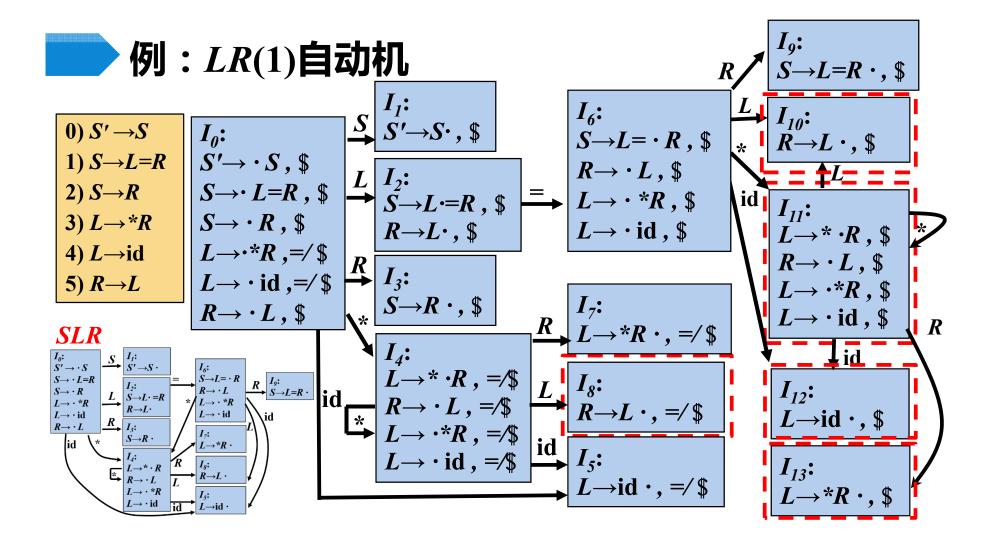
# 赋值语句文法的 LR(1)分析表

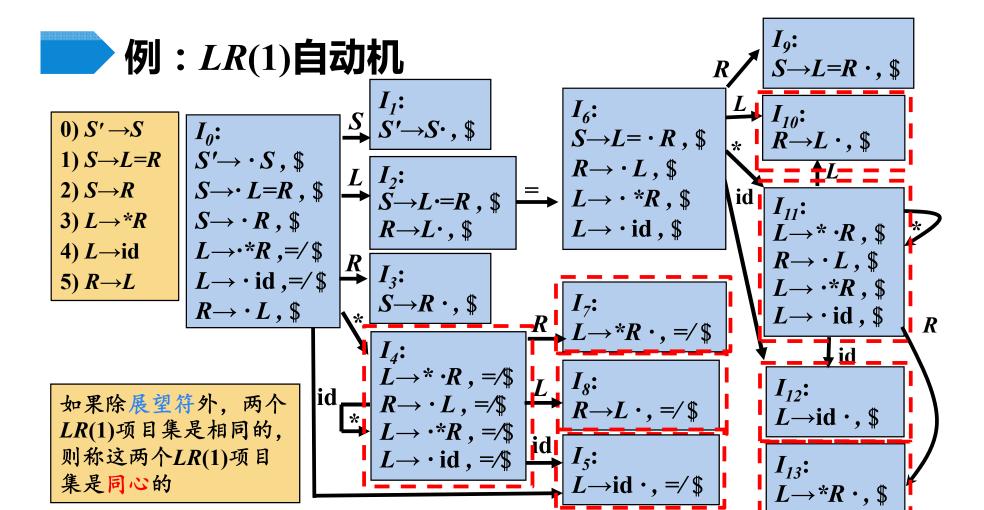
### 文法

- $0) S' \rightarrow S$
- 1)  $S \rightarrow L = R$
- 2)  $S \rightarrow R$
- 3)  $L \rightarrow *R$
- 4)  $L\rightarrow id$
- 5)  $R \rightarrow L$

11· *		ACT	TION	GOTO			
状态	*	id	=	\$	S	L	R
0	<b>s4</b>	<b>s</b> 5			1	2	3
1				acc			
2			<b>s6</b>	r5			
3				r2			
4	<b>s4</b>	<b>s</b> 5				8	7
5			r4	r4			
6	s11	s12				10	9
7			r3	r3			
8			r5	r5			
9				r1			
10				r5			
11	s11	s12				10	13
12				r4			
13				r3			_

### 例:*LR*(1)自动机 $S \rightarrow L = R \cdot , \$$ $I_6$ : $S \mid S' \rightarrow S',$ \$ $0) S' \rightarrow S$ $I_{\theta}$ : $S \rightarrow L = \cdot R$ , \$ $R \rightarrow L \cdot, \$$ $S' \rightarrow \cdot S$ , \$ 1) $S \rightarrow L = R$ $\begin{vmatrix} L \\ S \rightarrow L = R, \$ \end{vmatrix} = \begin{vmatrix} L \rightarrow \cdot *R, \$ \\ L \rightarrow \cdot *R, \$ \end{vmatrix}$ $R \rightarrow L$ , \$ $S \rightarrow L = R$ , \$ 2) $S \rightarrow R$ \id 3) $L \rightarrow R$ $S \rightarrow R$ , \$ $R \rightarrow L$ , \$ $L \rightarrow \cdot \mathrm{id}$ , \$ $L \rightarrow * R,$ $L \rightarrow *R = /$ 4) $L \rightarrow id$ $R I_3$ : $R \rightarrow L,$ \$ $L \rightarrow \cdot id = /$ 5) $R \rightarrow L$ $L \rightarrow {}^{\cdot *}R$ , \$ $S \rightarrow R \cdot ,$ $R \rightarrow L,$ \$ $L \rightarrow \cdot \mathrm{id}$ , \$ $R \longrightarrow L \rightarrow R \cdot , =/\$$ R $\operatorname{id} \xrightarrow{\overset{\bullet}{L}} \stackrel{I_4:}{\overset{L\to *\cdot R}{R\to L\cdot ,=/\$}} \stackrel{L}{\underset{L\to \cdot *R,=/\$}{}} \stackrel{I_8:}{\underset{R\to L\cdot ,=/\$}{}} \stackrel{I_8:}{\underset{L\to \cdot *R,=/\$}{}} \stackrel{I_6:}{\underset{I_5:}{}} \stackrel{I_6:}{\underset{I_5:}{}} \stackrel{I_7:}{\underset{I_5:}{}} \stackrel{I_7:}{\underset{I_7:}{}} \stackrel{I_7:}{\underset{I_7:}} \stackrel{I_7:}{\underset{I_7:}} \stackrel{I_7:$ **⊥**id *I*<sub>12</sub>: $L \rightarrow id \cdot, \$$ $FOLLOW(R) = \{ =, \$ \}$ $FOLLOW(L) = \{=, \$\}$ $L \rightarrow id \cdot, =/$ \$ $L \rightarrow *R \cdot , \$$





### LR(1)项目集闭包

 $CLOSURE(I) = I \cup \{ [B \rightarrow \gamma, b] \mid [A \rightarrow \alpha \cdot B\beta, a] \in CLOSURE(I), B \rightarrow \gamma \in P, b \in FIRST(\beta a) \}$ 

```
SetOfltems CLOSURE (I) {
repeat
for (I中的每个项[A \rightarrow \alpha \cdot B\beta, a])
for (G'的每个产生式B \rightarrow \gamma)
for (FIRST (\betaa)中的每个符号b)
将[B \rightarrow \cdot \gamma, b]加入到集合I中;
until 不能向I中加入更多的项;
until I;
}
```

# GOTO 函数

 $GOTO(I, X) = CLOSURE(\{[A \rightarrow \alpha X \cdot \beta, \mathbf{a}] | [A \rightarrow \alpha \cdot X \beta, \mathbf{a}] \in I\})$ 

```
SetOfltems GOTO (I, X) { 将J 初始化为空集; for (I \text{ 中的每个项}[A \rightarrow \alpha \cdot X\beta, \mathbf{a}]) 将项[A \rightarrow \alpha X \cdot \beta, \mathbf{a}]加入到集合J中; return CLOSURE (J); }
```

### 为文法G'构造LR(1)项集族

```
void items (G') {
    将C初始化为{CLOSURE ({[S' \rightarrow \cdot S, \$]})} ;
    repeat
    for (C中的每个项集 I)
    for (每个文法符号X)
    if (GOTO(I, X)非空且不在C中)
        将GOTO(I, X)加入C中;
    until 不再有新的项集加入到C中;
}
```

### LR(1)自动机的形式化定义

户文法

$$G = (V_N, V_T, P, S)$$

► LR(1)自动机

$$M = (C, V_N \cup V_T, GOTO, I_0, F)$$

- $\succ C = \{I_0\} \cup \{I \mid \exists J \in C, X \in V_N \cup V_T, I = GOTO(J,X)\}$
- $>I_0=CLOSURE(\{S'\rightarrow S, \$\})$
- $\succ F = \{ CLOSURE(\{S' \rightarrow S^{\bullet}, \$\}) \}$

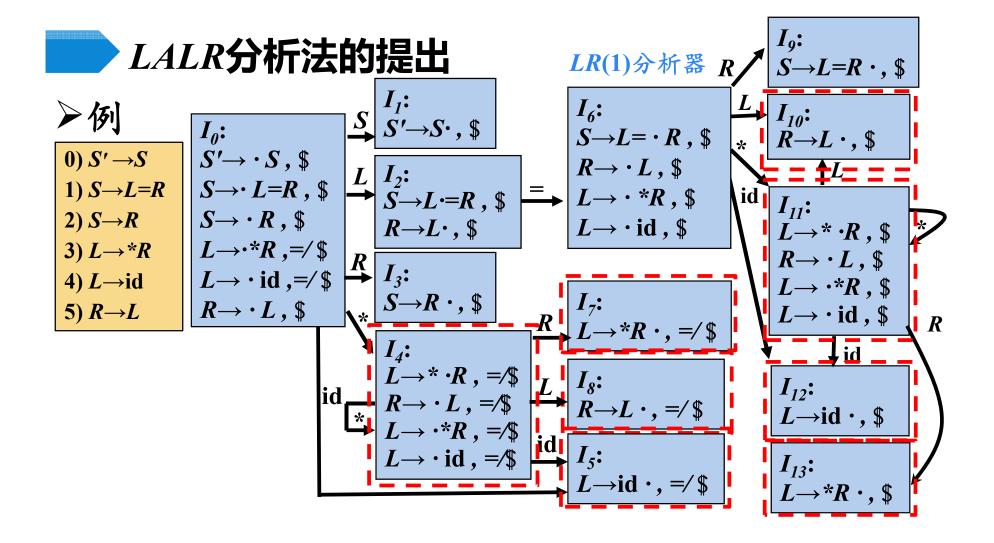
### LR分析表构造算法

- 构造G'的规范LR(1)项集族 $C = \{ I_0, I_1, \ldots, I_n \}$
- ▶根据I<sub>i</sub>构造得到状态i。状态i 的语法分析动作按照下面的方法决定:
  - $\succ if[A \rightarrow \alpha \cdot a\beta, b] \subseteq I_i \text{ and } GOTO(I_i, a) = I_i \text{ then } ACTION[i, a] = sj$
  - $\triangleright if[A \rightarrow \alpha \cdot B\beta, b] \subseteq I_i \text{ and } GOTO(I_i, B) = I_i \text{ then } GOTO[i, B] = j$
  - ightharpoonup if  $[A 
    ightharpoonup lpha \cdot, \mathbf{a}] \in I_i \perp A \neq S'$  then ACTION[i, a] = rj (j是产生式A 
    ightharpoonup lpha的编号)
  - $\triangleright if[S' \rightarrow S; \$] \subseteq I_i then ACTION[i, \$] = acc;$
- ▶没有定义的所有条目都设置为"error"

如果LR(1)分析表中没有语法分析动作冲突,那么给定的文法就称为LR(1)文法

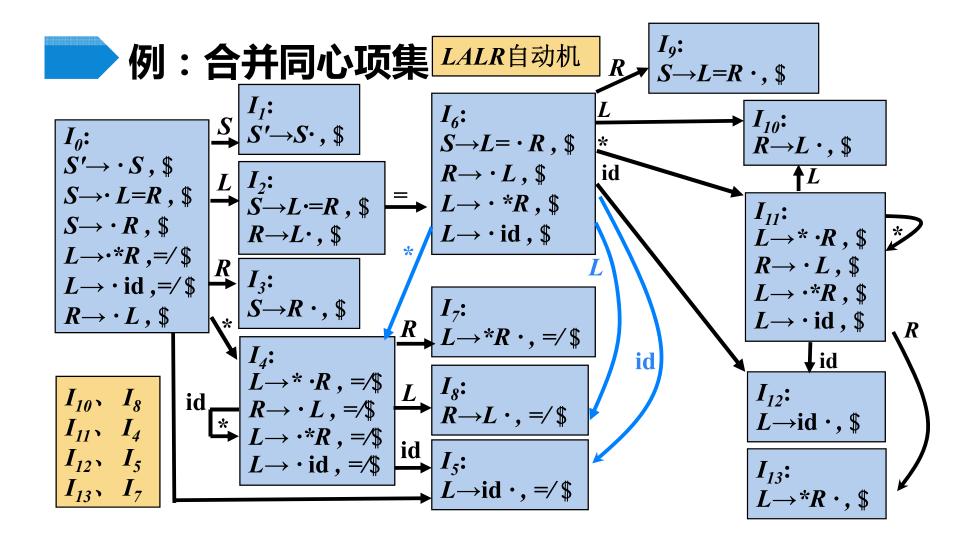


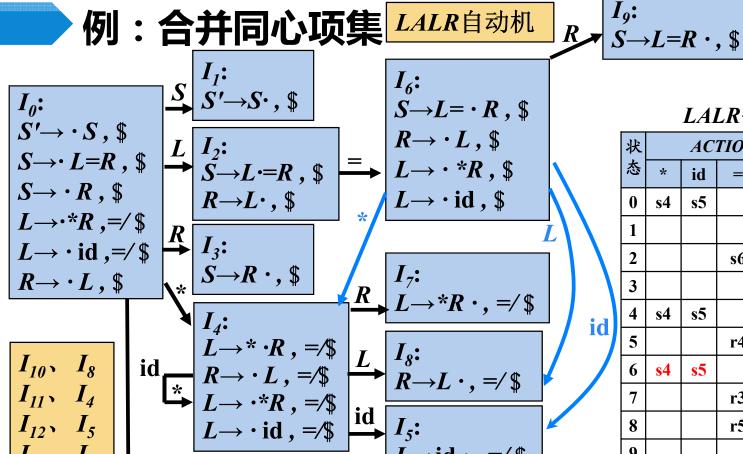




### LALR (lookahead-LR)分析的基本思想

- ▶寻找具有相同核心的LR (1) 项集,并将这些项集合并为一个项集。所谓项集的核心就是其第一分量的集合
- >然后根据合并后得到的项集族构造语法分析表
- ▶如果分析表中没有语法分析动作冲突,给定的文法就称为LALR (1) 文法,就可以根据该分析表进行语法分析



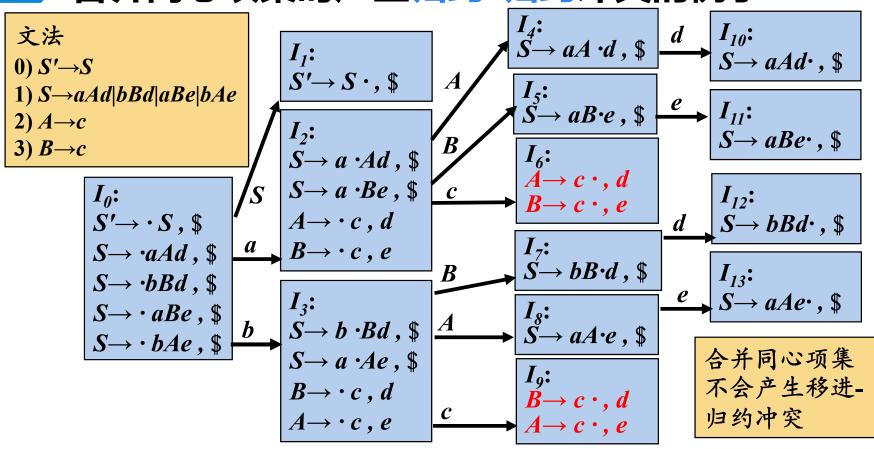


 $L \rightarrow id \cdot , =/$ \$

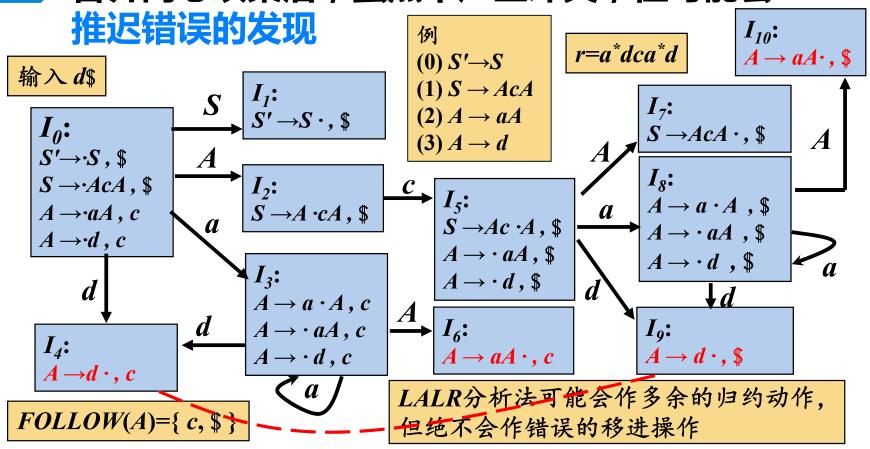
### LALR分析表

	状		AC	GOTO				
	态	*	id	=	\$	S	L	R
	0	<b>s4</b>	<b>s</b> 5			1	2	3
	1				acc			
	2			<b>s6</b>	r5			
	3				r2			
	4	<b>s4</b>	<b>s</b> 5				8	7
	5			r4	r4			
	6	<b>s4</b>	<b>s</b> 5				8	9
	7			r3	r3			
	8			r5	r5			
	9				r1			

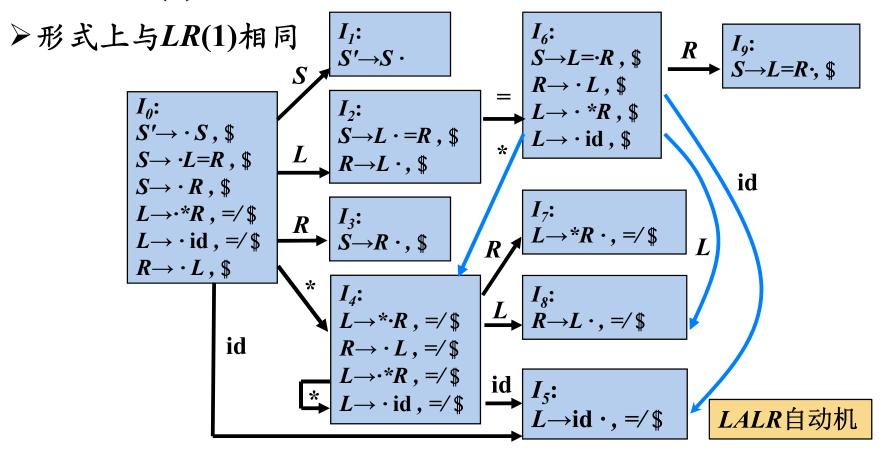
### 合并同心项集时产生归约-归约冲突的例子



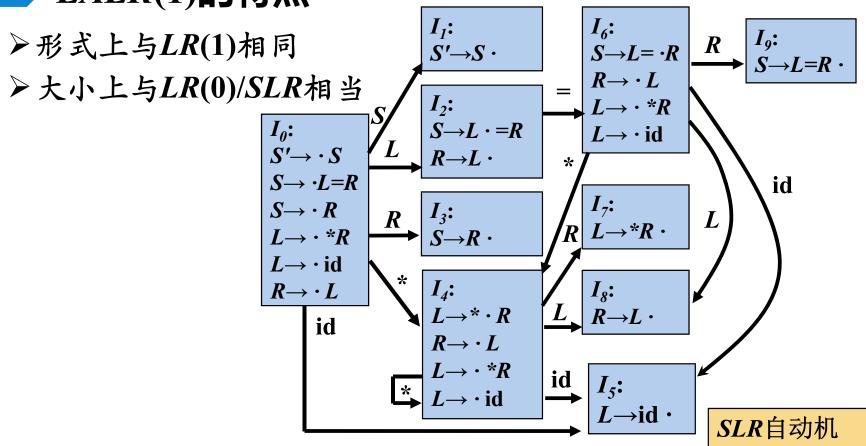
### 合并同心项集后,虽然不产生冲突,但可能会



### LALR(1)的特点



### LALR(1)的特点



### **LALR**(1)的特点

- ▶形式上与LR(1)相同
- ▶大小上与LR(0)/SLR相当
- ▶分析能力介于SLR和LR(1)二者之间

SLR < LALR(1) < LR(1)

▶合并后的展望符集合仍为FOLLOW集的子集





### 二义性文法的特点

- ▶每个二义性文法都不是LR的
- ▶某些类型的二义性文法在语言的描述和实现中很有用
  - ▶更简短、更自然

〉例

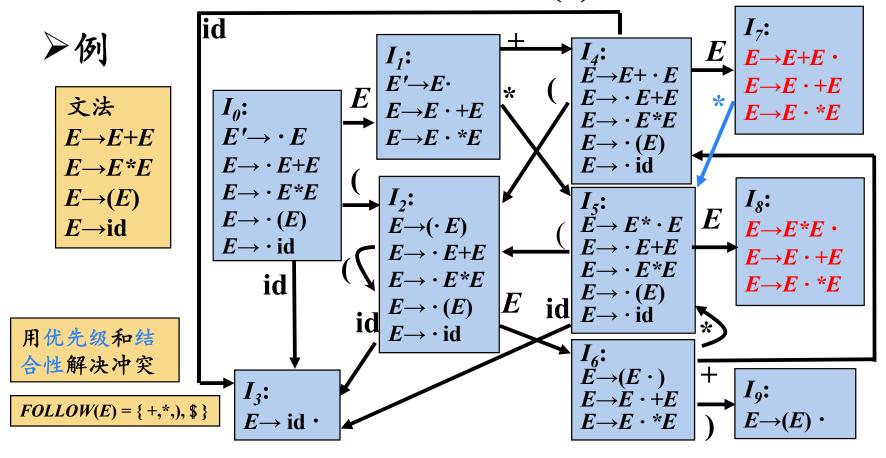
二义性文法

- $\textcircled{1} E \rightarrow E + E$
- $2 E \rightarrow E^*E$
- $\textcircled{4} E \rightarrow \operatorname{id}$

非二义性文法

- $\bigcirc 1 E \rightarrow E + T$
- $2 E \rightarrow T$
- $\textcircled{3} T \rightarrow T * F$
- $\textcircled{4} T \to F$
- $\bigcirc F \rightarrow (E)$
- $\bigcirc F \rightarrow id$

# 二义性算术表达式文法的LR(0)分析器



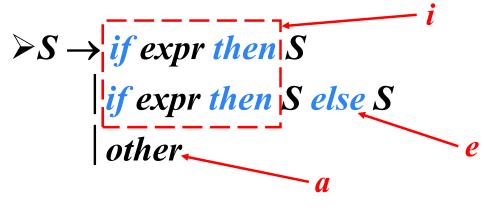
### 二义性算术表达式文法的SLR分析表

文法  $E \rightarrow E + E$   $E \rightarrow E * E$   $E \rightarrow (E)$   $E \rightarrow id$ 

状	ACTION						GOTO
状态	id	+	*	(	)	\$	E
0	<b>s</b> 3			<b>s2</b>			1
1		<b>s4</b>	<b>s</b> 5			acc	
2	<b>s</b> 3			<b>s2</b>			6
3		r4	r4		r4	r4	
4	<b>s</b> 3			<b>s2</b>			7
5	<b>s</b> 3			<b>s2</b>			8
6		<b>s4</b>	<b>s</b> 5		<b>s9</b>		
7		r1	<b>s</b> 5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	

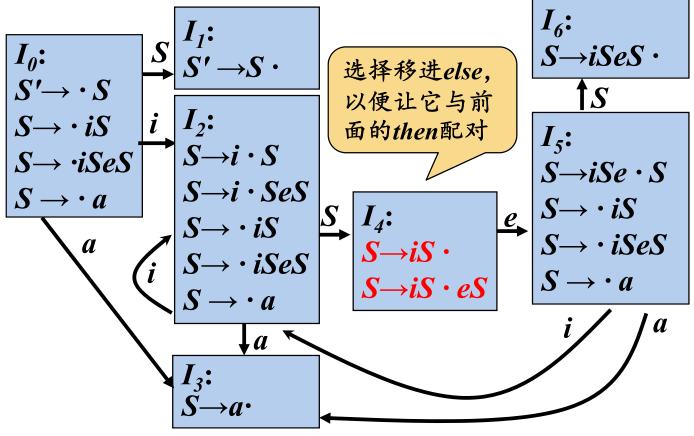
 $FOLLOW(E)=\{+,*,),\$\}$ 

# 例:二义性if 语句文法的LR分析



$$>S \rightarrow iS | iS eS | a$$





# 二义性if语句文法的SLR分析表

状态		GOTO			
	i	e	a	\$	S
0	<b>s2</b>		<b>s</b> 3		1
1				acc	
2	<b>s2</b>		<b>s</b> 3		4
3		r3		r3	
4		s <b>5</b>		r1	
5	s2		s3		6
6		r2		r2	

#### 二义性文法的使用

▶应该保守地使用二义性文法,并且必须在严格 控制之下使用,因为稍有不慎就会导致语法分 析器所识别的语言出现偏差



### LR分析中的错误处理

- ▶语法错误的检测
  - ▶当LR分析器在查询分析表并发现一个报错条目时, 就检测到了一个语法错误
- >错误恢复策略
  - > 恐慌模式错误恢复
  - 户短语层次错误恢复

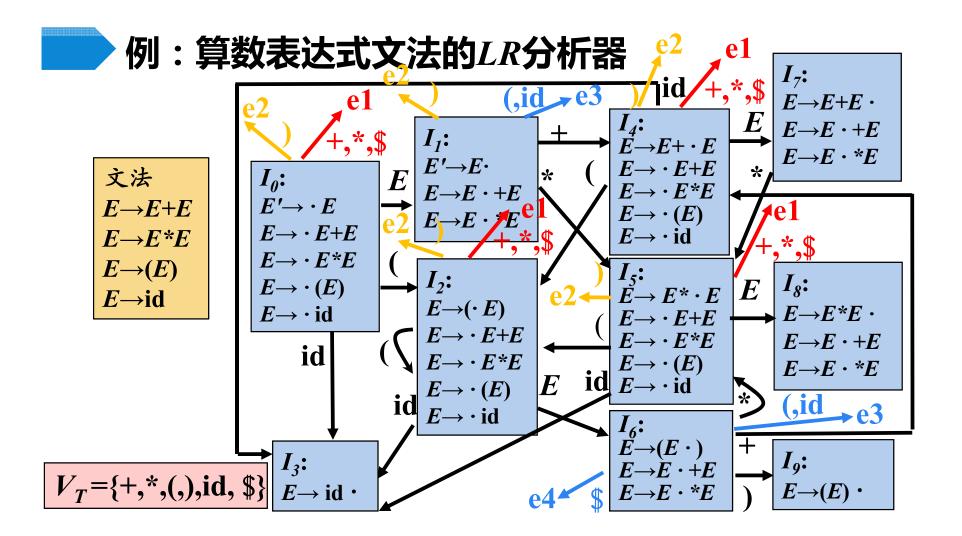
#### 恐慌模式错误恢复

$$S_0S_1...S_i S_{i+1}...S_m$$
  
 $X_1...X_i A ...X_m$ 

- 》从栈顶向下扫描,直到发现某个状态 $s_i$ ,它有一个对应于某个非终结符A的GOTO目标,可以认为从这个A推导出的串中包含错误
- ▶然后丢弃0个或多个输入符号,直到发现一个可能合法地跟在 A之后的符号a为止。
- $\triangleright$ 之后将 $s_{i+1}$ = $GOTO(s_i, A)$ 压入栈中,继续进行正常的语法分析

#### 短语层次错误恢复

- ▶检查LR分析表中的每一个报错条目,并根据语言的使用方法来决定程序员所犯的何种错误最有可能引起这个语法错误
- >然后构造出适当的恢复过程



#### 带有错误处理子程序的算术表达式文法LR分析表

状	ACTION						GOTO
态	id	+	*	(		\$	E
0	<b>s</b> 3	e1	e1	<b>s2</b>	<b>e2</b>	e1	1
1	<b>e3</b>	<b>s4</b>	<b>s</b> 5	e3	<b>e2</b>	acc	
2	<b>s</b> 3	e1	e1	s2	<b>e2</b>	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	<b>e2</b>	e1	7
5	<b>s</b> 3	e1	e1	s2	<b>e2</b>	e1	8
6	<b>e3</b>	s4	<b>s5</b>	e3	<b>s9</b>	e4	
7	r1	r1	<b>s5</b>	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	

#### >错误处理例程

- ► e1: 将状态3压入栈中,发出 诊断信息"缺少运算分量"
- ▶ e2: 从输入中删除")",发 出诊断信息"不匹配的右括号"
- ▶ e3:将状态4压入栈中,发出 诊断信息"缺少运算符"
- ▶ e4:将状态9压入栈中,发出 诊断信息"缺少右括号"

