

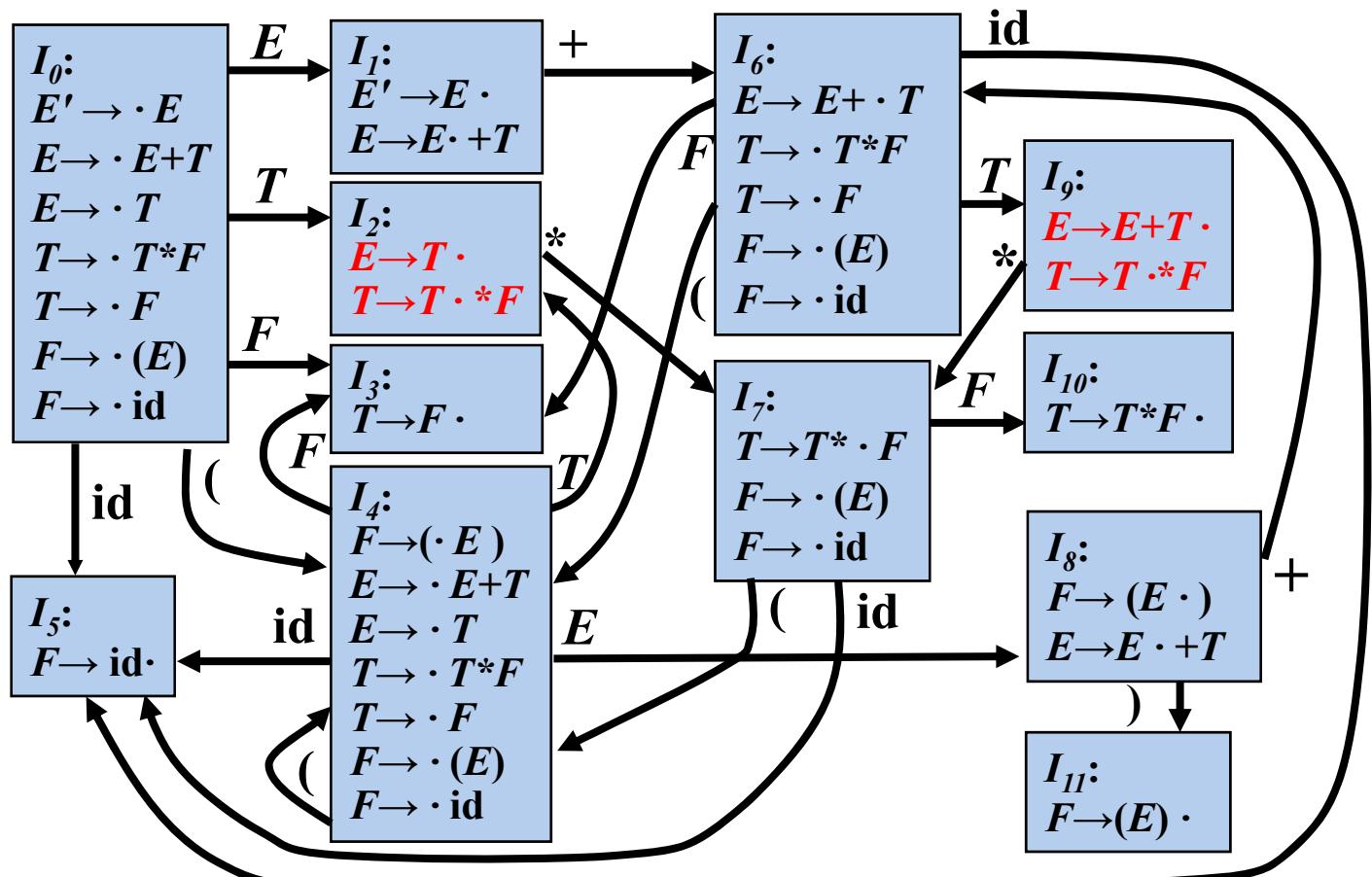
## 4.4.2 SLR 分析

例：LR(0) 分析过程中的冲突

文法：

- (0)  $E' \rightarrow E$
- (1)  $E \rightarrow E+T$
- (2)  $E \rightarrow T$
- (3)  $T \rightarrow T^*F$
- (4)  $T \rightarrow F$
- (5)  $F \rightarrow (E)$
- (6)  $F \rightarrow id$

$X$	$FOLLOW(X)$
$E$	$), +, \$$
$T$	$), +, \$, *$
$F$	$), +, \$, *$



## ► SLR 分析

► SLR分析法的基本思想

已知项目集  $I$ :

$$A_1 \rightarrow \alpha_1 \cdot a_1 \beta_1$$

$$A_2 \rightarrow \alpha_2 \cdot a_2 \beta_2$$

...

$$A_m \rightarrow \alpha_m \cdot a_m \beta_m$$

$$B_1 \rightarrow \gamma_1 \cdot$$

$$B_2 \rightarrow \gamma_2 \cdot$$

...

$$B_n \rightarrow \gamma_n \cdot$$

}  $m$  个移进项目

}  $n$  个归约项目

如果集合  $\{a_1, a_2, \dots, a_m\}$  和  $FOLLOW(B_1), FOLLOW(B_2), \dots, FOLLOW(B_n)$  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

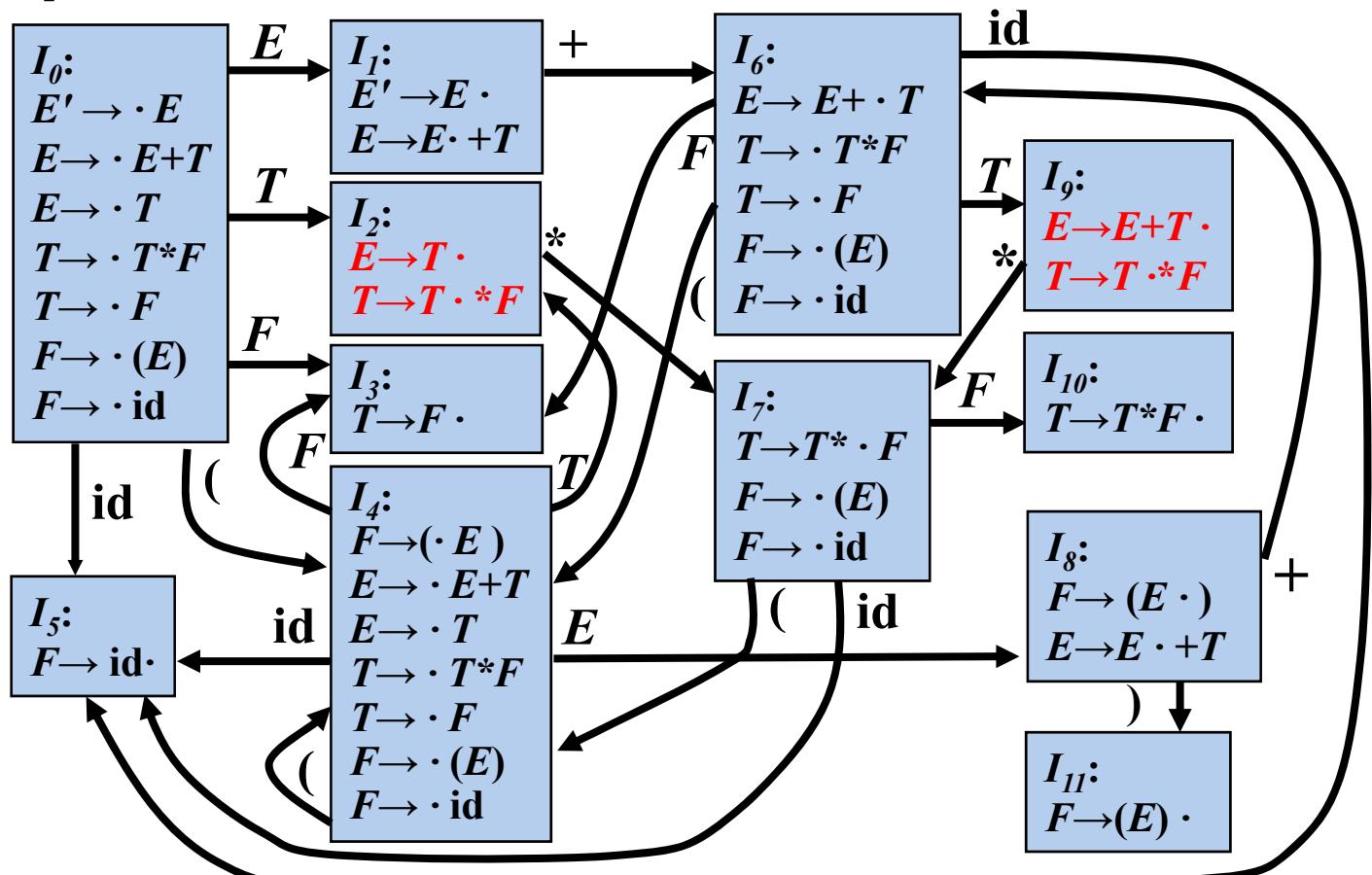
- 若  $a \in \{a_1, a_2, \dots, a_m\}$ , 则 移进  $a$
- 若  $a \in FOLLOW(B_i)$ , 则 用产生式  $B_i \rightarrow \gamma_i$  归约
- 此外，报错

## 例：LR(0) 分析过程中的冲突

文法：

- (0)  $E' \rightarrow E$
- (1)  $E \rightarrow E+T$
- (2)  $E \rightarrow T$
- (3)  $T \rightarrow T^*F$
- (4)  $T \rightarrow F$
- (5)  $F \rightarrow (E)$
- (6)  $F \rightarrow id$

$X$	$FOLLOW(X)$
$E$	$), +, \$$
$T$	$), +, \$, *$
$F$	$), +, \$, *$

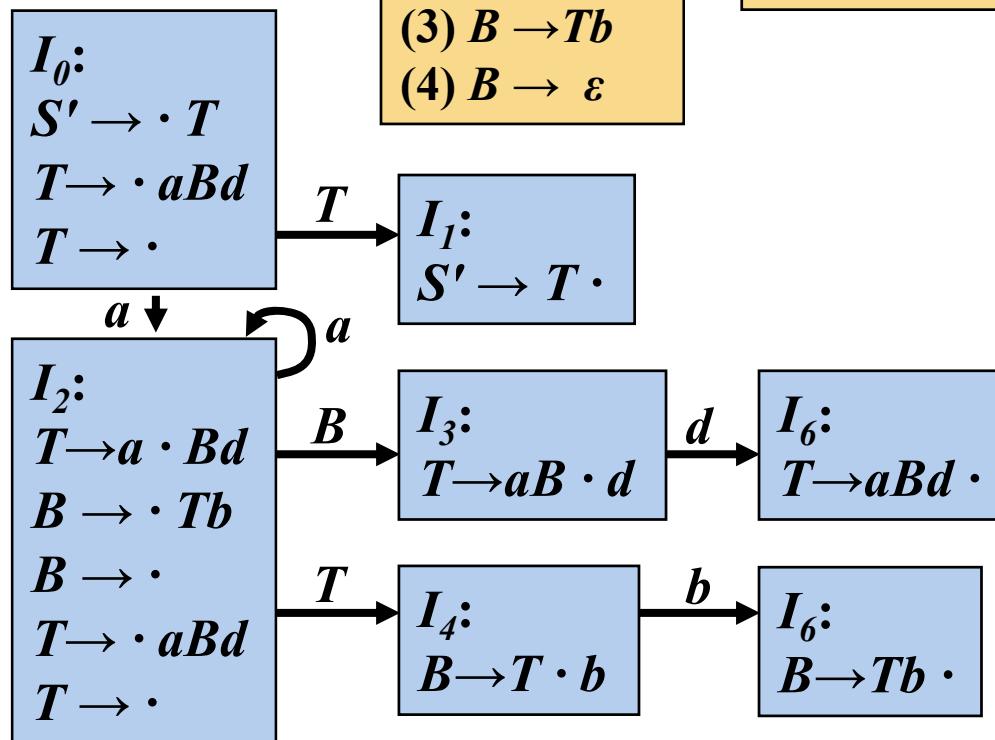




## 表达式文法的SLR分析表

状态	ACTION						GOTO		
	id	+	*	(	)	\$	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

- (0)  $S' \rightarrow T$
- (1)  $T \rightarrow aBd$
- (2)  $T \rightarrow \epsilon$
- (3)  $B \rightarrow Tb$
- (4)  $B \rightarrow \epsilon$

$FOLLOW(S') = \{ \$ \}$   
 $FOLLOW(T) = \{ \$, b \}$   
 $FOLLOW(B) = \{ d \}$

SLR(1)分析表

状态	ACTION				GOTO	
	$a$	$b$	$d$	$\$$	$T$	$B$
0	s2	r2		r2	1	
1				acc		
2	s2	r2	r4	r2	4	3
3			s5			
4		s6				
5		r1		r1		
6			r3			



## SLR 分析表构造算法

- 构造 $G'$ 的规范 $LR(0)$ 项集族 $C = \{ I_0, I_1, \dots, I_n \}$ 。
- 根据 $I_i$ 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：
  - $\text{if } A \rightarrow \alpha \cdot a\beta \in I_i \text{ and } GOTO(I_i, a) = I_j \text{ then } ACTION[i, a] = sj$
  - $\text{if } A \rightarrow a \cdot B\beta \in I_i \text{ and } GOTO(I_i, B) = I_j \text{ then } GOTO[i, B] = j$
  - $\text{if } A \rightarrow \alpha \cdot \in I_i \text{ 且 } A \neq S' \text{ then for } \forall a \in FOLLOW(A) \text{ do}$   
 $ACTION[i, a] = rj$  (j是产生式 $A \rightarrow \alpha$ 的编号)
  - $\text{if } S' \rightarrow S \cdot \in I_i \text{ then } ACTION[i, \$] = acc;$
- 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**

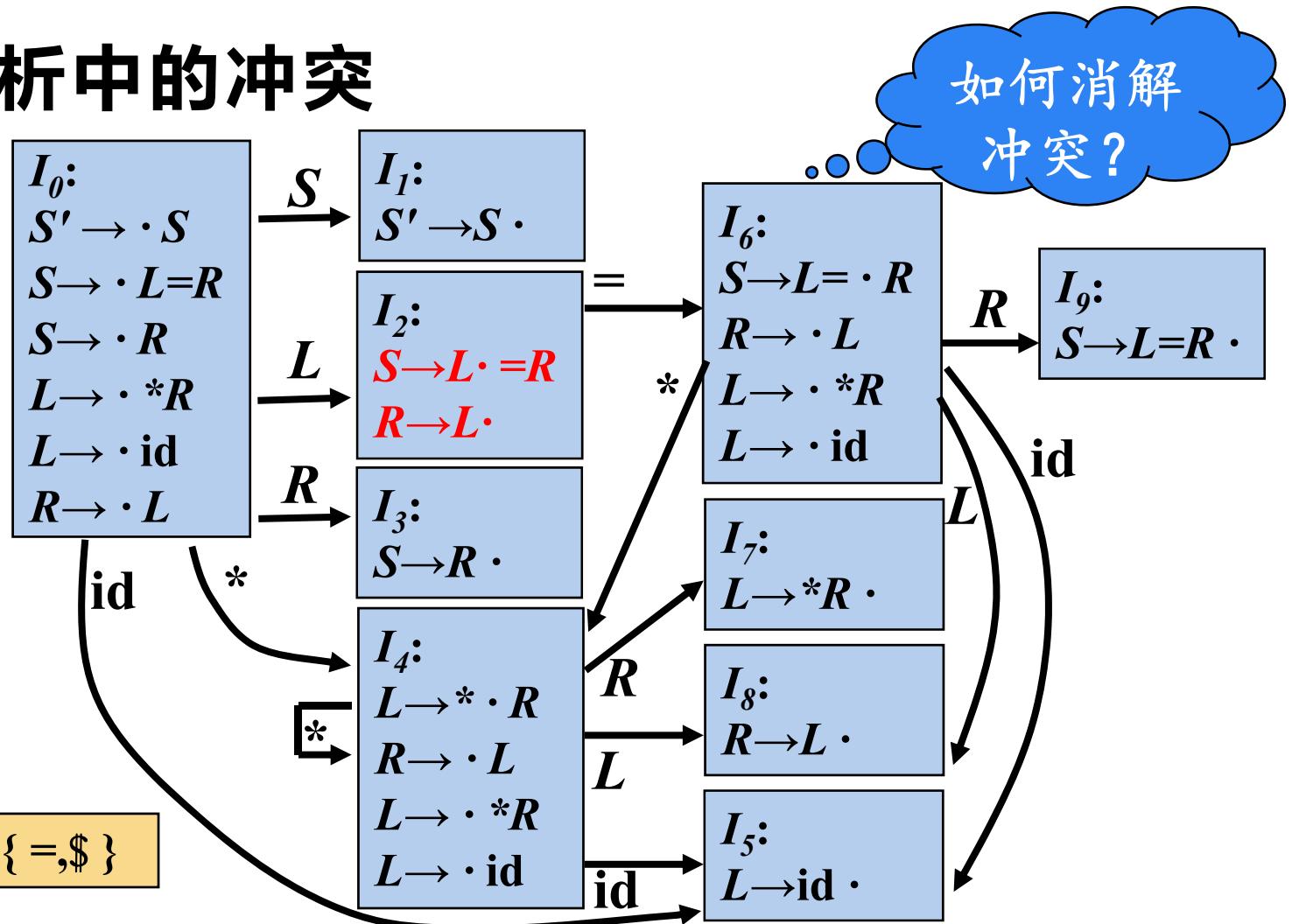


## SLR 分析中的冲突

➤ 例

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

$$FOLLOW(R) = \{ =, \$ \}$$





### 4.4.3 LR(1)分析

- *SLR*分析存在的问题
- *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  $A \rightarrow a$  相关联的  $\text{FOLLOW}(A)$ ，但  $b \in \text{FOLLOW}(A)$  只是归约  $a$  的一个必要条件，而非充分条件

## ► LR(1)分析法的提出

► 对于产生式  $A \rightarrow a$  的归约，在不同的使用位置，  
 $A$  会要求不同的后缀符号

0)  $S' \rightarrow S$

1)  $S \rightarrow L=R$

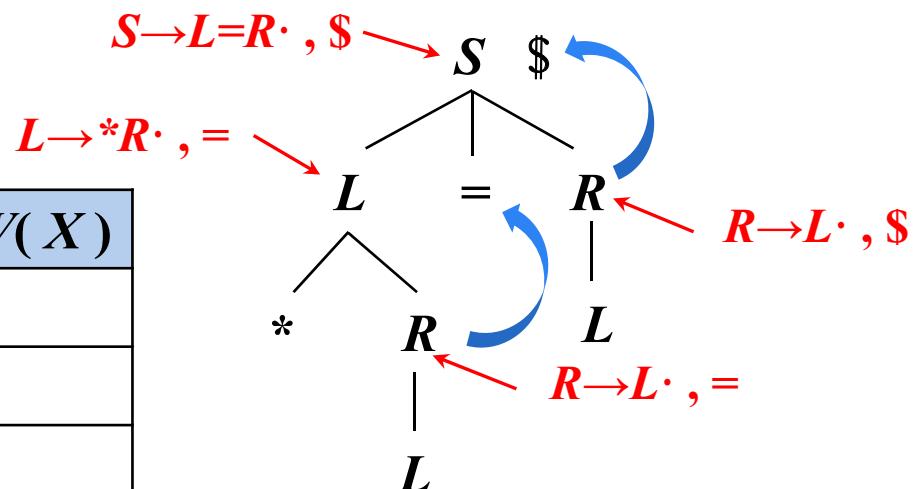
2)  $S \rightarrow R$

3)  $L \rightarrow *R$

4)  $L \rightarrow \text{id}$

5)  $R \rightarrow L$

$X$	$FOLLOW(X)$
$S$	$\$$
$L$	$=, \$$
$R$	$=, \$$



► 在特定位置， $A$  的后缀符集合是  $FOLLOW(A)$  的子集

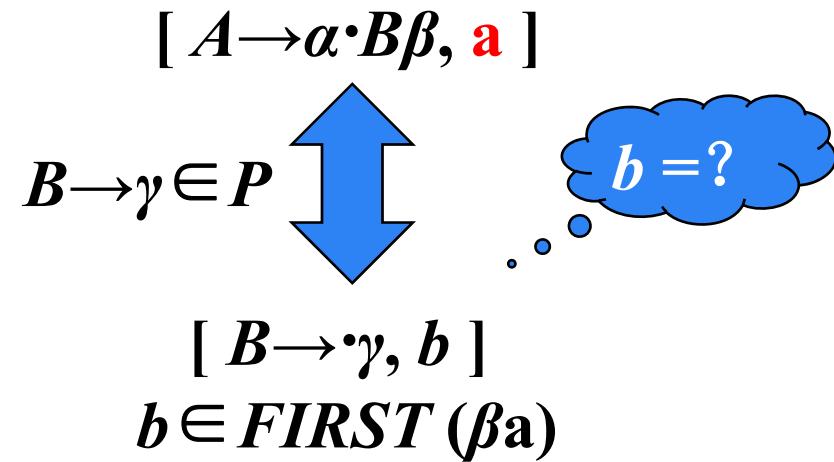
## 规范LR(1)项目

- 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
- 在形如  $[A \rightarrow \alpha \cdot \beta, a]$  且  $\beta \neq \epsilon$  的项中，展望符a没有任何作用
- 但是一个形如  $[A \rightarrow \alpha \cdot, a]$  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

$$\begin{array}{c} [ A \rightarrow \alpha \cdot B \beta, \textcolor{red}{a} ] \\ B \rightarrow \gamma \in P \\ \updownarrow \\ ? \end{array}$$

## 等价LR(1)项目

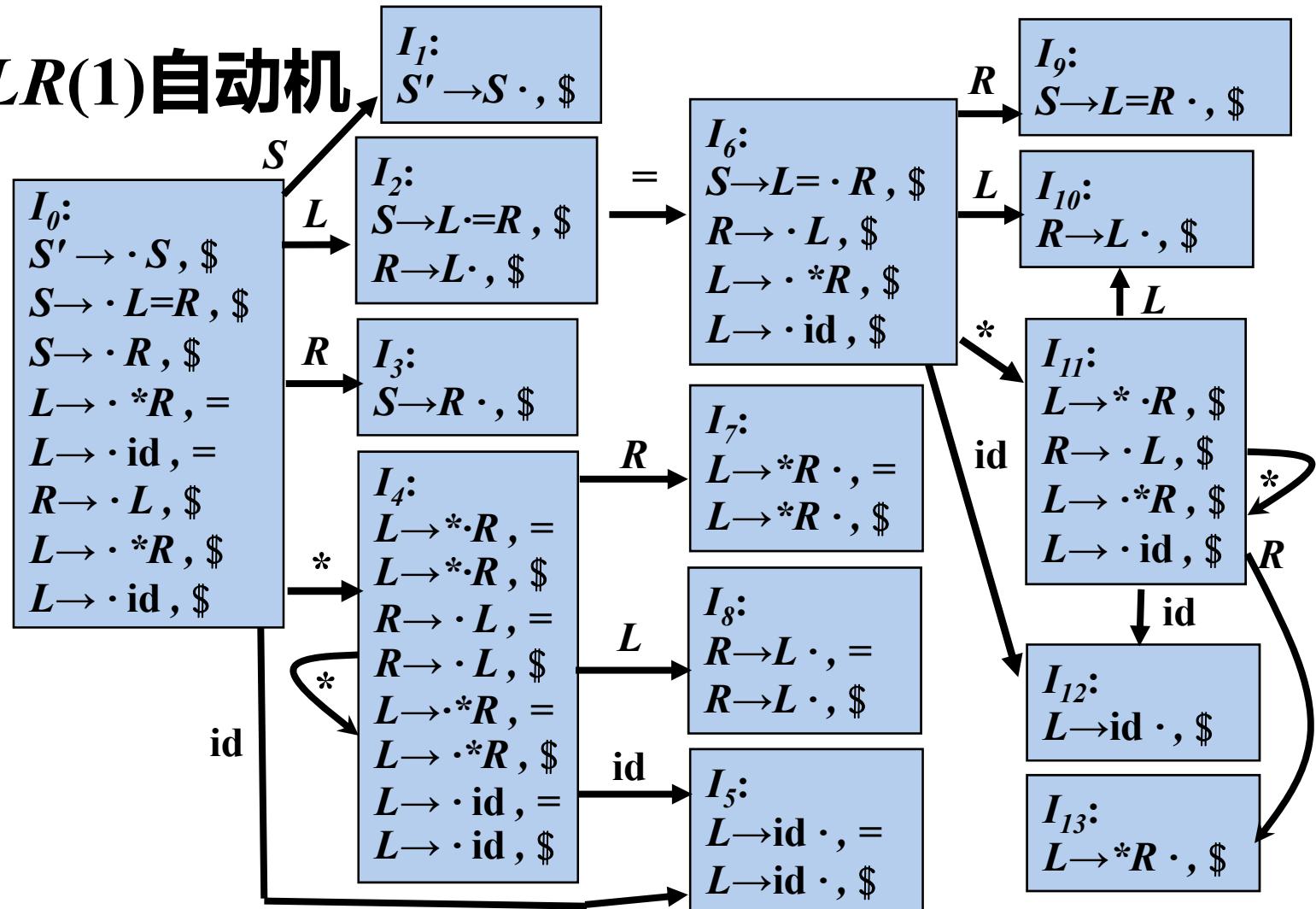


当  $\beta \Rightarrow^+ \varepsilon$  时，此时  $b=a$  叫 **继承的后继符**，  
否则叫 **自生的后继符**



## 例 : LR(1)自动机

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





## 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)  $R \rightarrow L$

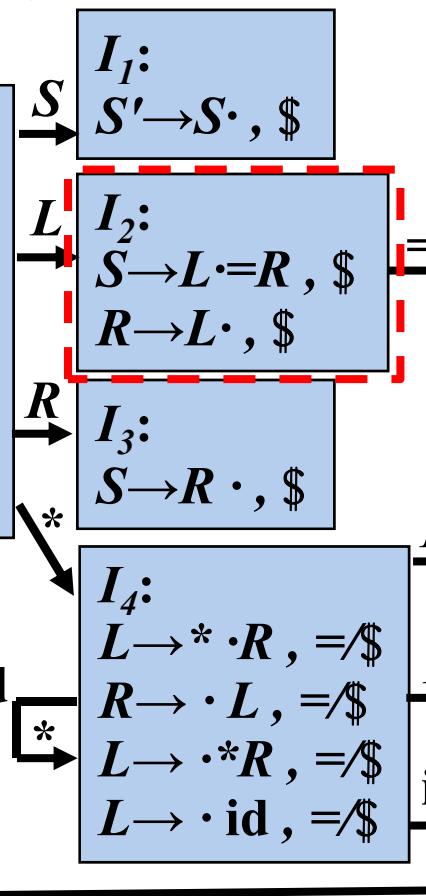
状态	ACTION				GOTO		
	*	id	=	\$	$S$	$L$	$R$
0	s4	s5			1	2	3
1				acc			
2			s6	r5			
3				r2			
4	s4	s5			8	7	
5			r4	r4			
6	s11	s12			10	9	
7			r3	r3			
8			r5	r5			
9				r1			
10				r5			
11	s11	s12			10	13	
12				r4			
13				r3			



## 例 : LR(1)自动机

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

$I_0:$   
 $S' \rightarrow \cdot S, \$$   
 $S \rightarrow \cdot L=R, \$$   
 $S \rightarrow \cdot R, \$$   
 $L \rightarrow \cdot *R, =/\$$   
 $L \rightarrow \cdot id, =/\$$   
 $R \rightarrow \cdot L, \$$



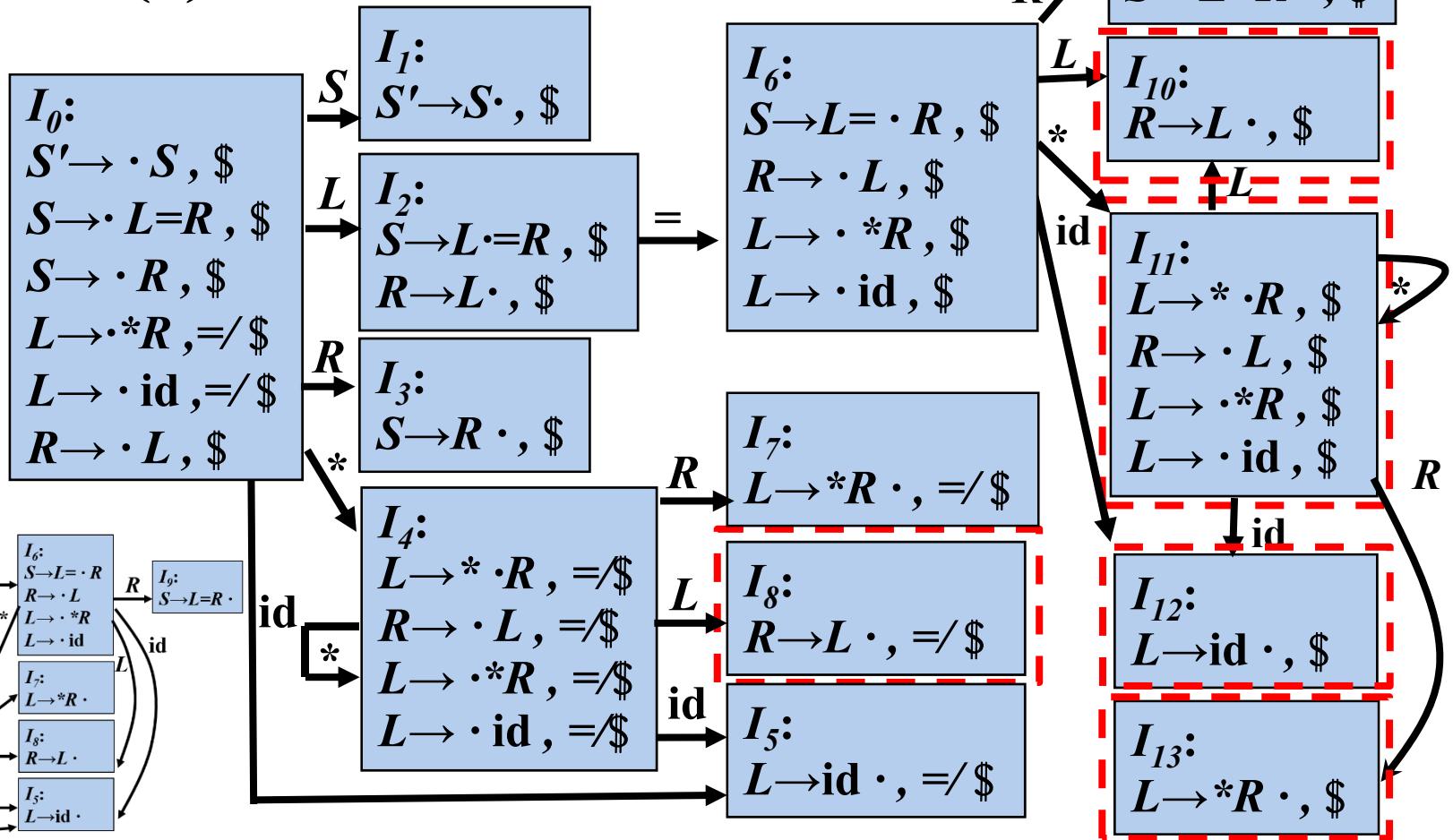
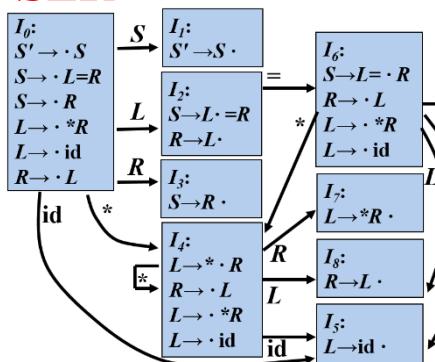
$$FOLLOW(R) = \{ =, \$ \}$$
$$FOLLOW(L) = \{ =, \$ \}$$



## 例 : LR(1)自动机

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

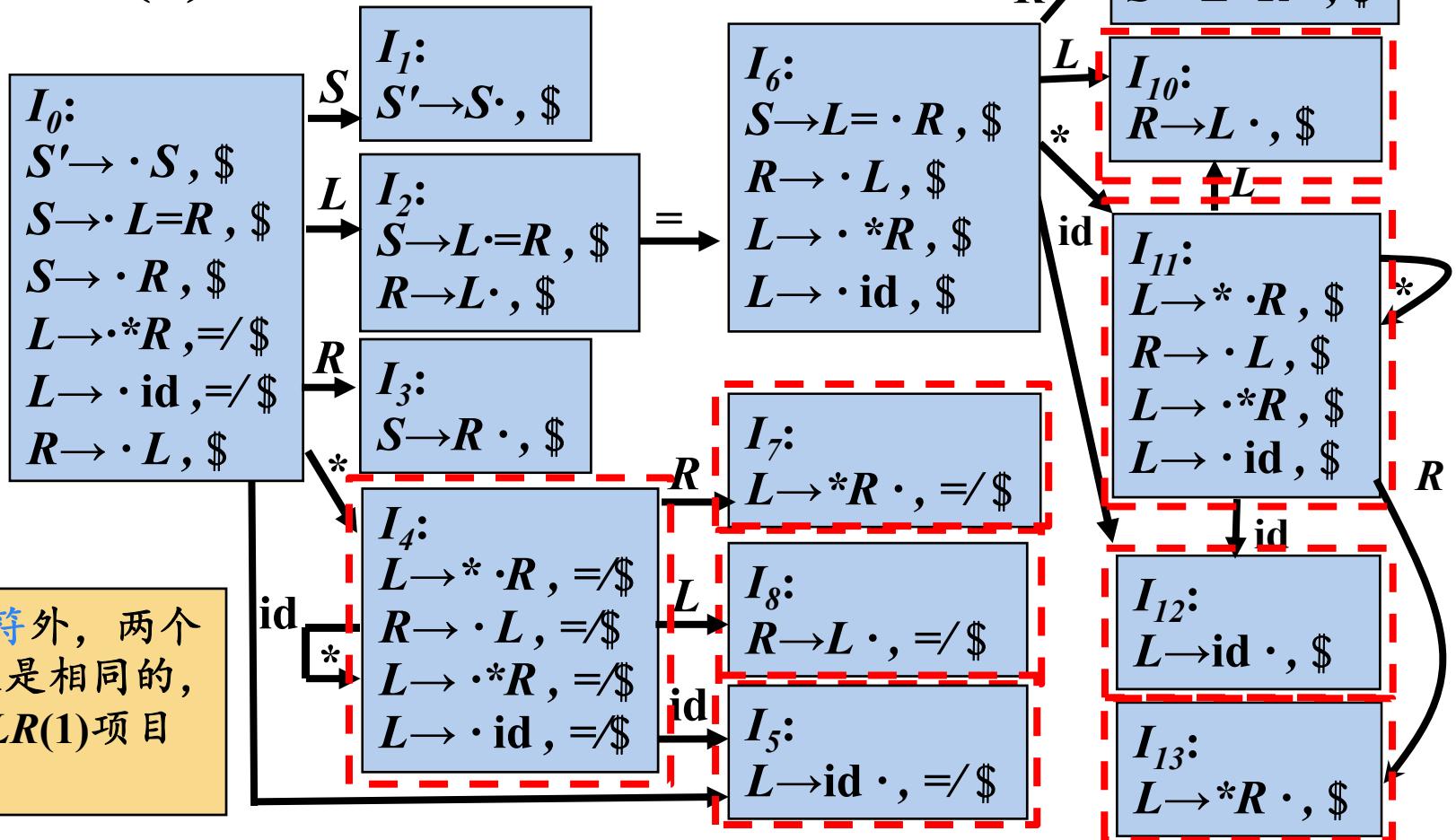
**SLR**





## 例 : LR(1)自动机

- 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)项目集闭包

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

```
SetOfItems CLOSURE (I) {
    repeat
        for (I 中的每个项 [A → α·Bβ, a])
            for (G' 的每个产生式 B → γ)
                for (FIRST (βa) 中的每个符号 b )
                    将 [B → ·γ, 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

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

```
SetOfItems GOTO (I, X) {
    将J 初始化为空集;
    for (I 中的每个项 [A → α · Xβ, a])
        将项 [A → α X · β, 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**

➤ 文法

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

➤  $LR(1)$  自动机

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

➤  $C = \{I_0\} \cup \{I \mid \exists J \in C, X \in V_N \cup V_T, I = GOTO(J, X)\}$

➤  $I_0 = CLOSURE(\{S' \rightarrow \cdot S, \$\})$

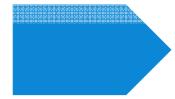
➤  $F = \{ CLOSURE(\{S' \rightarrow S^\bullet, \$\}) \}$



## LR分析表构造算法

- 构造 $G'$ 的规范 $LR(1)$ 项集族 $C = \{ I_0, I_1, \dots, I_n \}$
- 根据 $I_i$ 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：
  - $\text{if } [A \rightarrow a \cdot a\beta, b] \in I_i \text{ and } GOTO(I_i, a) = I_j \text{ then } ACTION[i, a] = sj$
  - $\text{if } [A \rightarrow a \cdot B\beta, b] \in I_i \text{ and } GOTO(I_i, B) = I_j \text{ then } GOTO[i, B] = j$
  - $\text{if } [A \rightarrow a \cdot, a] \in I_i \text{ 且 } A \neq S' \text{ then } ACTION[i, a] = rj$
  - $\text{if } [S' \rightarrow S \cdot, \$] \in I_i \text{ then } ACTION[i, \$] = acc;$
- 没有定义的所有条目都设置为“error”

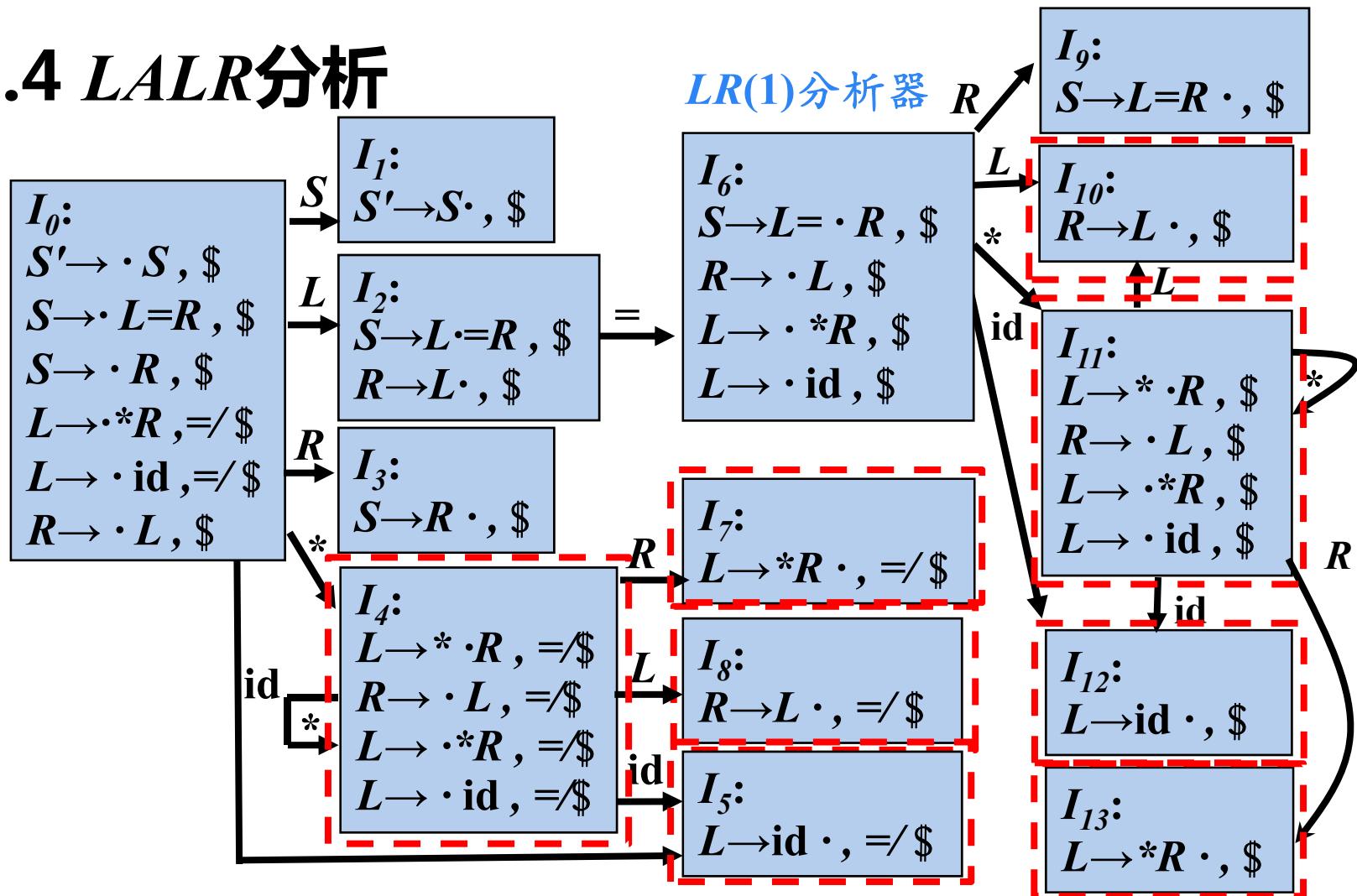
如果 $LR(1)$ 分析表中没有语法分析动作冲突，  
那么给定的文法就称为 $LR(1)$ 文法



## 4.4.4 LALR分析

➤ 例

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





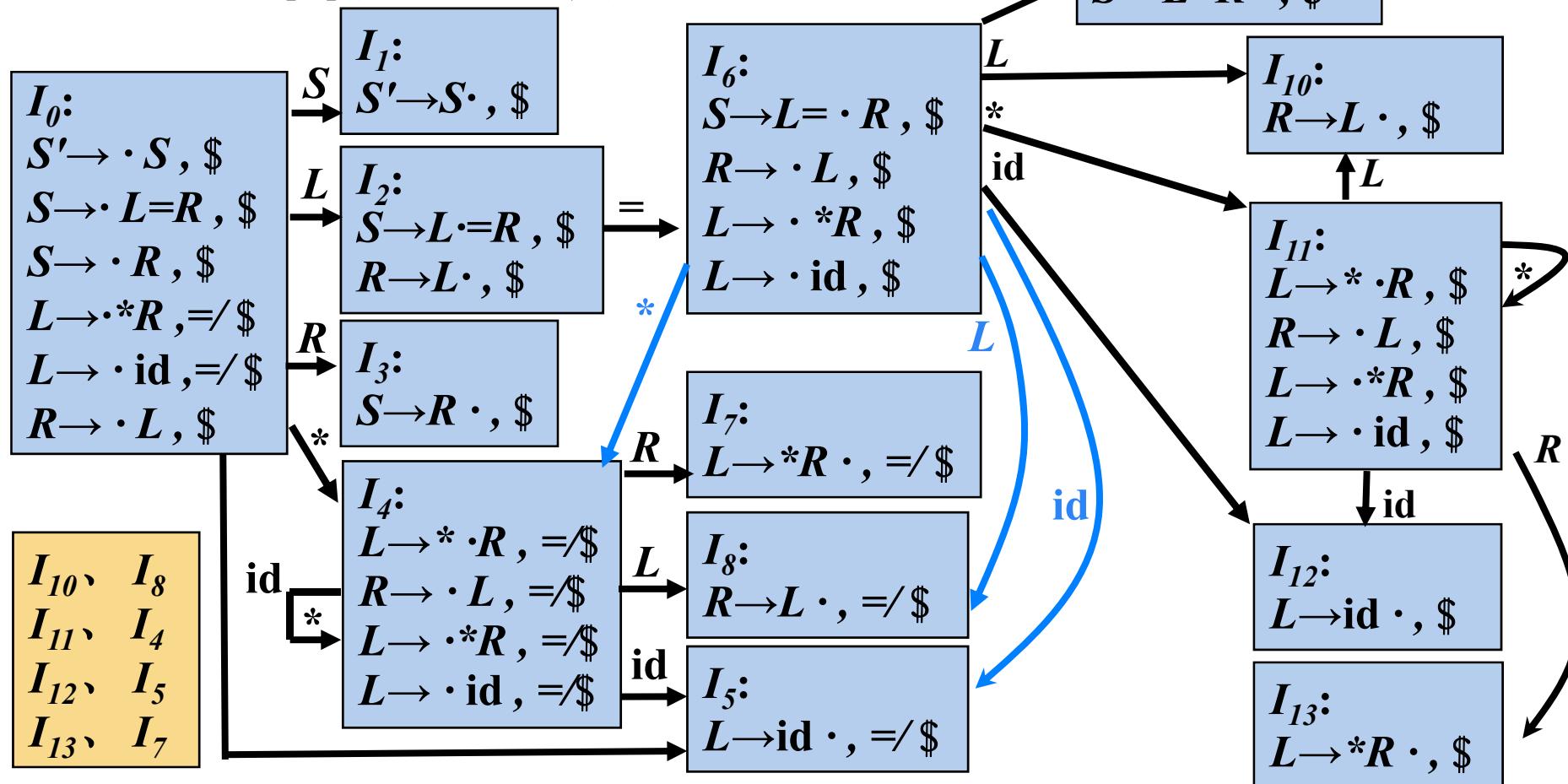
## *LALR ( lookahead-LR )分析的基本思想*

- 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



## 例：合并同心项集

LALR自动机

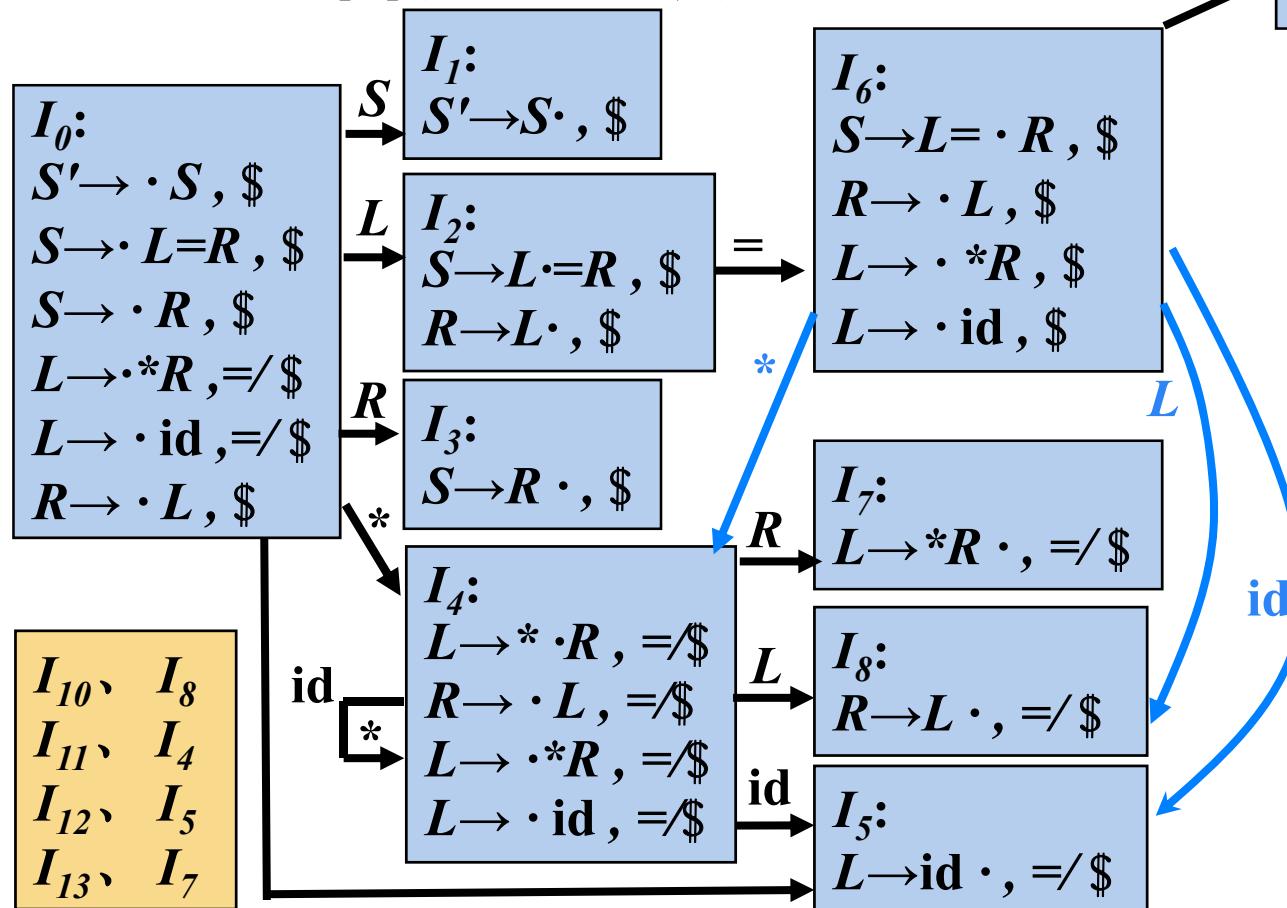




## 例：合并同心项集

LALR自动机

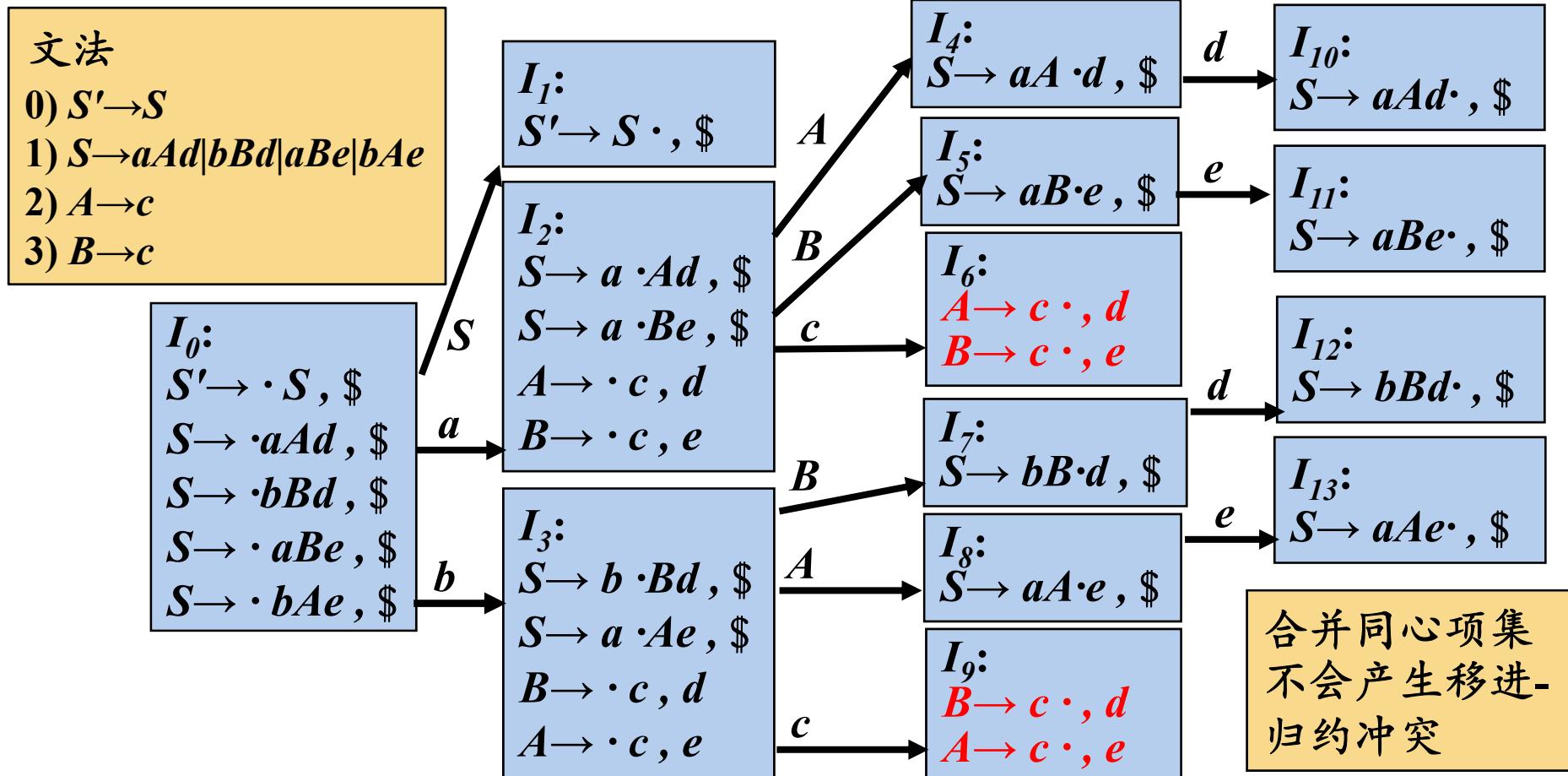
$I_9: S \rightarrow L=R \cdot, \$$



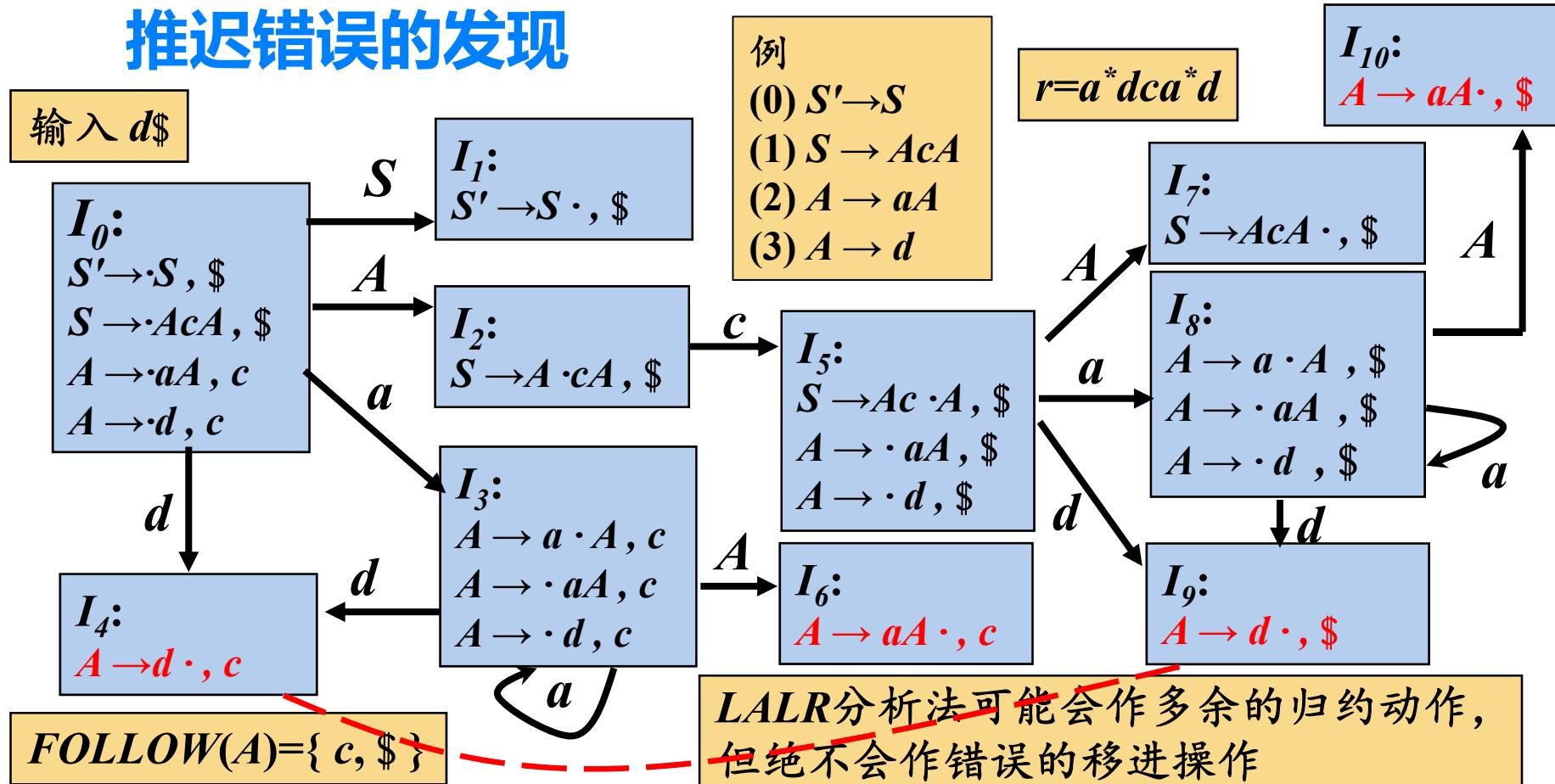
LALR分析表

状态	ACTION					GOTO		
	*	id	=	\$		S	L	R
0	s4	s5				1	2	3
1					acc			
2			s6	r5				
3				r2				
4	s4	s5					8	7
5			r4	r4				
6	<b>s4</b>	<b>s5</b>					<b>8</b>	<b>9</b>
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

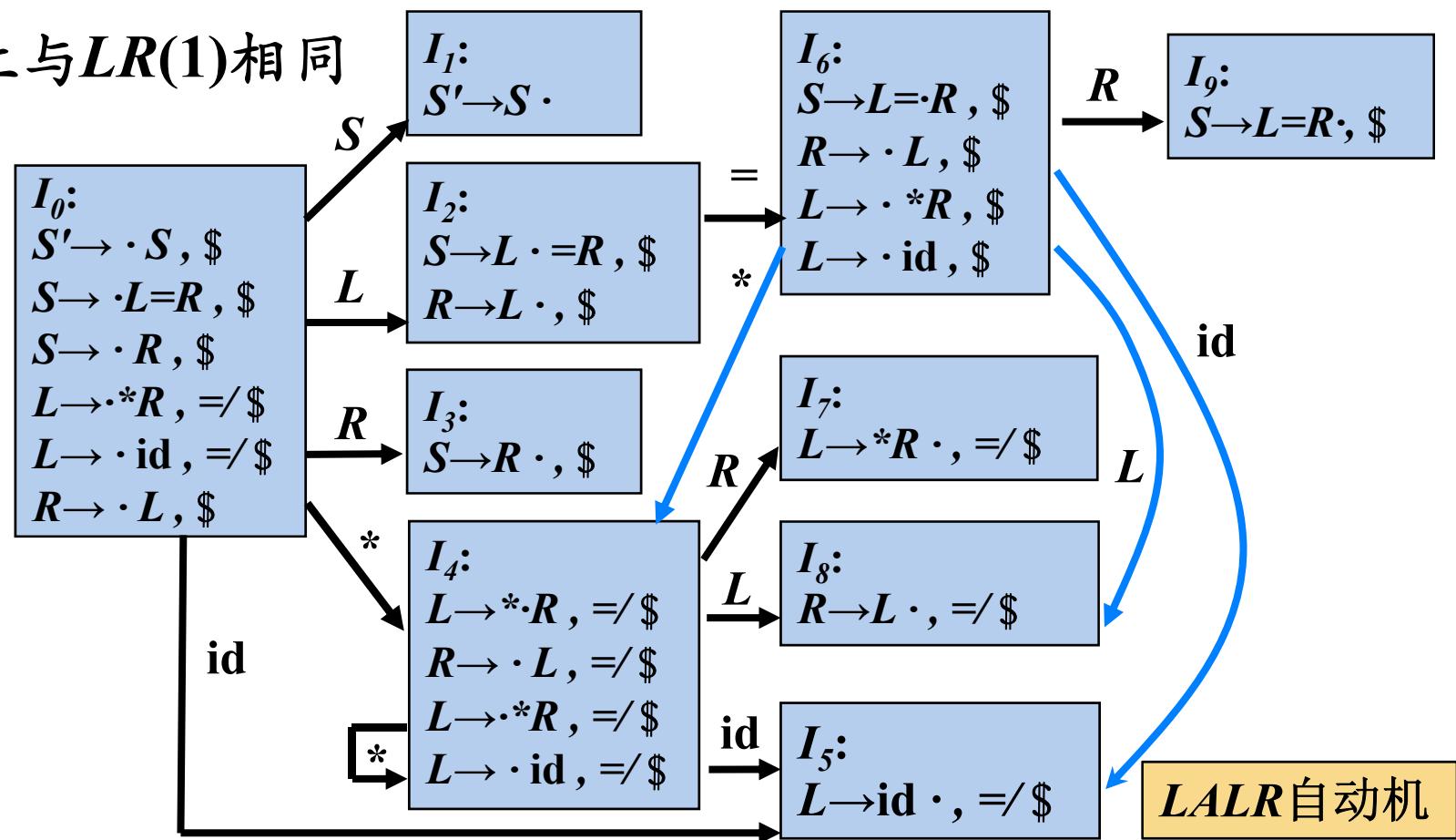


## 合并同心项集后，虽然不产生冲突，但可能会推迟错误的发现



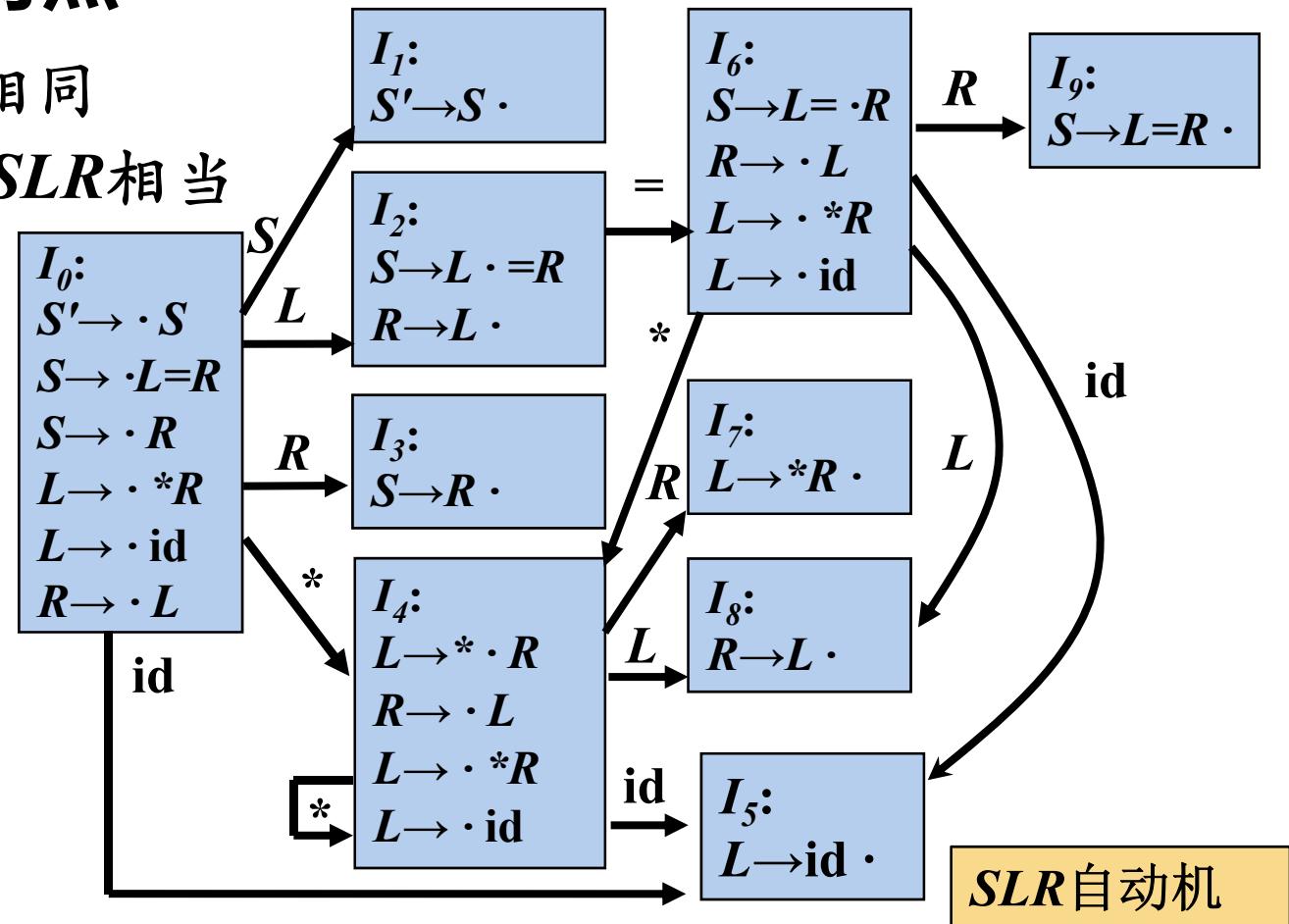
## ► $LALR(1)$ 的特点

► 形式上与 $LR(1)$ 相同



## ► LALR(1)的特点

- 形式上与LR(1)相同
- 大小上与LR(0)/SLR相当





## ***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



## 4.4.5 二义性文法的LR分析

- 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	
①	$E \rightarrow E + E$
②	$E \rightarrow E^* E$
③	$E \rightarrow ( E )$
④	$E \rightarrow \text{id}$

### 非二义性文法

- ①  $E \rightarrow E + T$
- ②  $E \rightarrow T$
- ③  $T \rightarrow T * F$
- ④  $T \rightarrow F$
- ⑤  $F \rightarrow ( E )$
- ⑥  $F \rightarrow \text{id}$

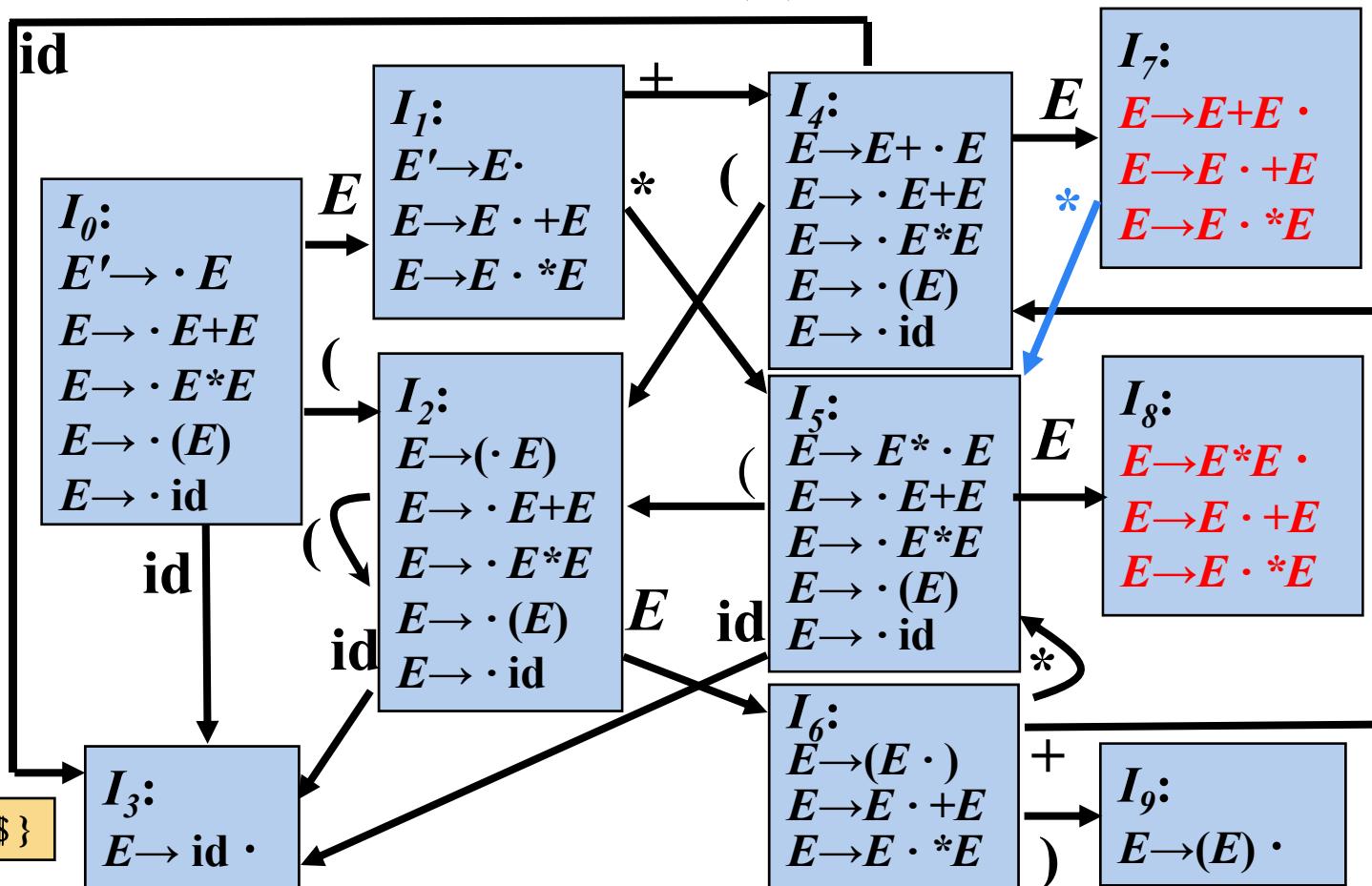
## 二义性算术表达式文法的LR(0)分析器

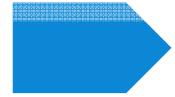
➤ 例

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

用优先级和结合性解决冲突

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



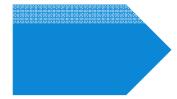


## 二义性算术表达式文法的SLR分析表

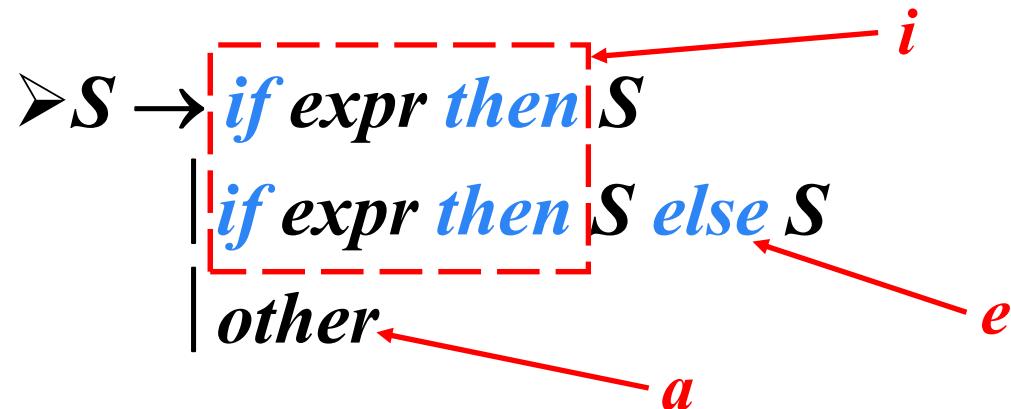
文法  
 $E \rightarrow E + E$   
 $E \rightarrow E * E$   
 $E \rightarrow (E)$   
 $E \rightarrow \text{id}$

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

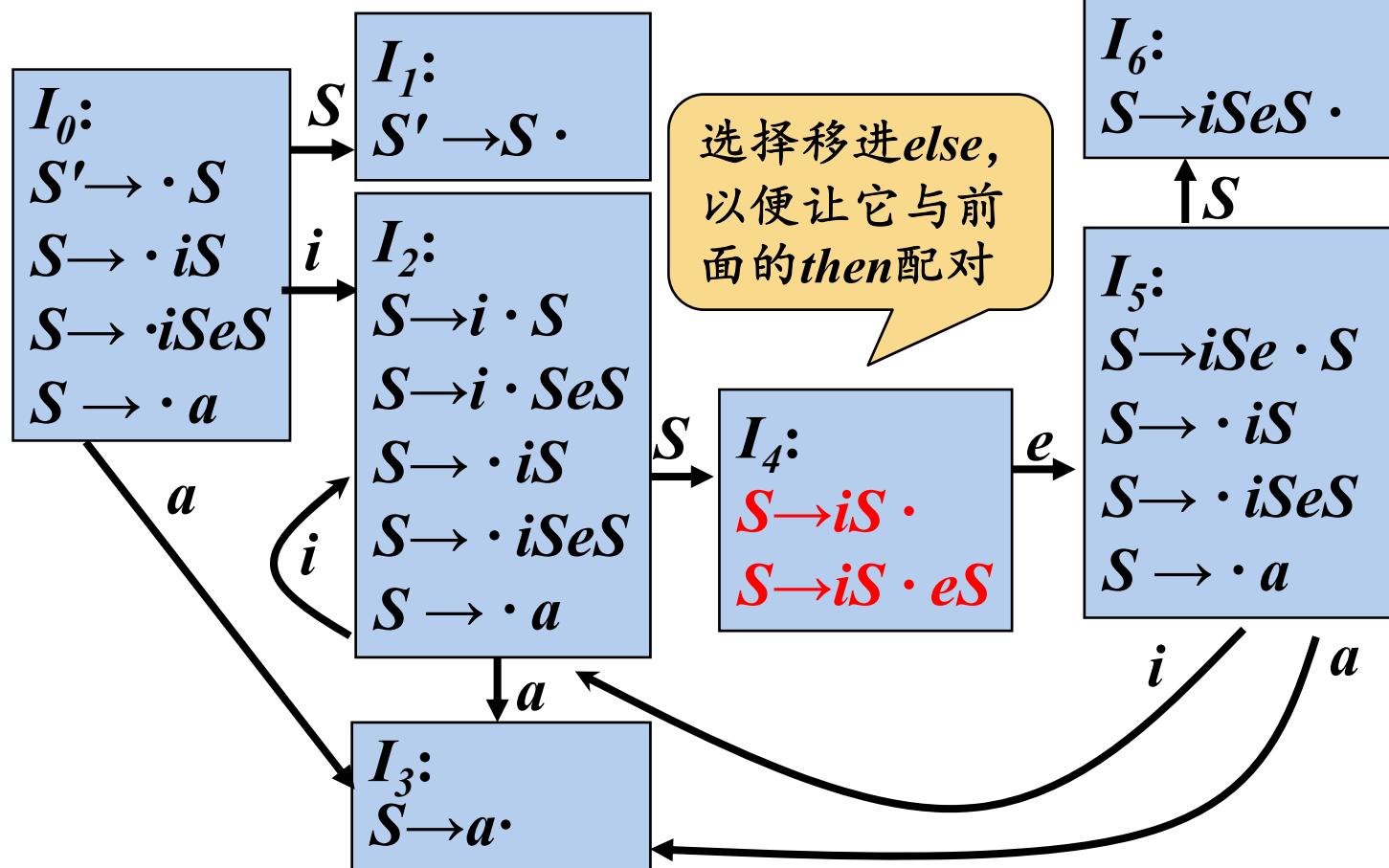
状态	ACTION						$GOTO$
	id	+	*	(	)	\$	
0	s3				s2		1
1		s4	s5			acc	
2	s3				s2		6
3		r4	r4		r4	r4	
4	s3				s2		7
5	s3				s2		8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	



## 例：二义性if语句文法的LR分析


$$\succ S \rightarrow i\ S \mid i\ S\ e\ S \mid a$$

→  $S \rightarrow iS | iSeS | a$





## 二义性if语句文法的SLR分析表

状态	ACTION				GOTO
	i	e	a	\$	
0	s2		s3		1
1				acc	
2	s2		s3		4
3		r3		r3	
4		s5		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



## 4.4.6 LR分析中的错误处理

- 语法错误的检测
  - 当LR分析器在查询语法分析动作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
  - 短语层次错误恢复



## 恐慌模式错误恢复

$$\begin{array}{c} s_0 s_1 \dots s_i s_{i+1} \dots s_m \\ \$ X_1 \dots X_i A \dots X_m \end{array}$$

- 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。
- 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

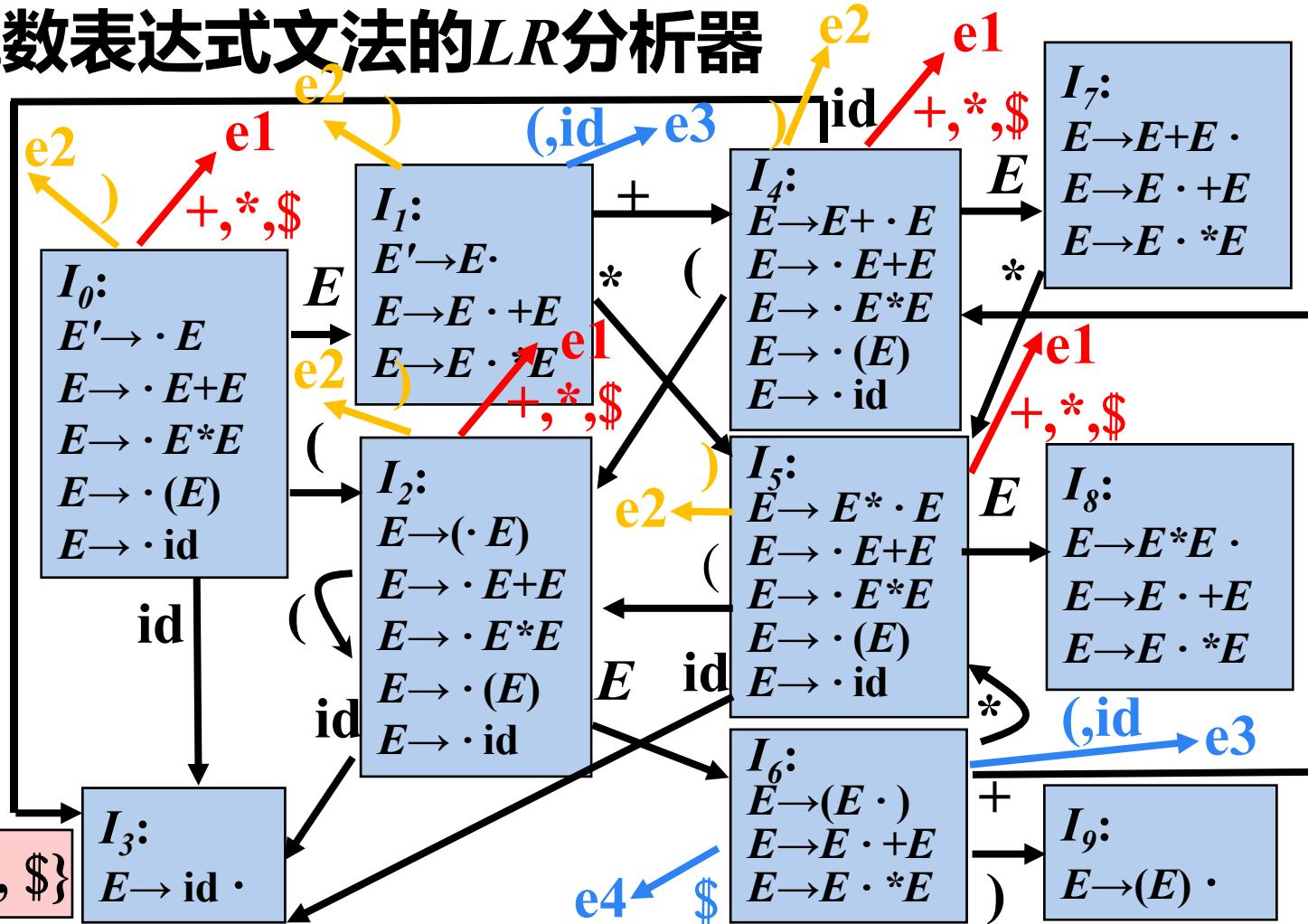
- 检查LR分析表中的每一个报错条目，并根据语言的使用方法来决定程序员所犯的何种错误最有可能引起这个语法错误
- 然后构造出适当的恢复过程



## 例：算数表达式文法的LR分析器

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

$V_T = \{+, *, (,), id, \$\}$



# 带有错误处理子程序的算术表达式文法LR分析表

状态	ACTION						GOTO
	id	+	*	(	)	\$	
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	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”



## 本章小结

- 自顶向下的分析
- 预测分析法
  - LL(1) 文法
  - 递归的预测分析法
  - 非递归的预测分析法
  - 预测分析中的错误处理
- 自底向上的分析
- LR分析法
  - LR(0) 分析
  - SLR 分析
  - LR(1) 分析
  - LALR ( lookahead-LR ) 分析法
  - 二义性文法的LR分析
  - LR分析中的错误处理
- 语法分析器自动生成工具



结束

