



5.1 “移近—归约”分析法

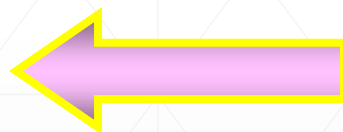
5.3 LR分析鸟瞰

5.4 LR (0)分析

5.5 SLR (1)分析

5.6 LR (1)分析

5.7 LALR (1)分析



5.8 LR分析对二义文法的应用

5.9 LR分析的错误处理与恢复

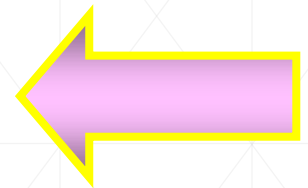
5.10 语法分析器的自动生成与YACC(自学)





## 5.7 LALR (1)分析

**5.7.1 LALR( 1 )分析实现思想**

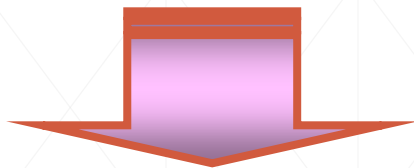


**5.7.2 LALR( 1 )分析表的构造**



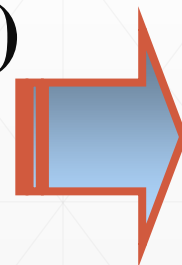
分析功能:  $\text{LR}(1) \supset \text{SLR}(1) \supset \text{LR}(0)$

存储开销:  $\text{LR}(1) \uparrow \text{SLR}(1) \text{LR}(0) \downarrow$



存储开销  $\Rightarrow \text{LR}(0), \text{SLR}(1)$

分析功能  $\Rightarrow \text{LR}(1)$



**LALR(1)**



**LALR(1)**

└─┬─→ **Look -Ahead LR(1)**

**LALR(1)分析表状态数 = LR(0)的状态数**

**LALR(1)分析功能  $\subseteq$  LR(1)**



例： 设有文法 $G(S)$ 为

$$S \rightarrow L=R \mid R$$

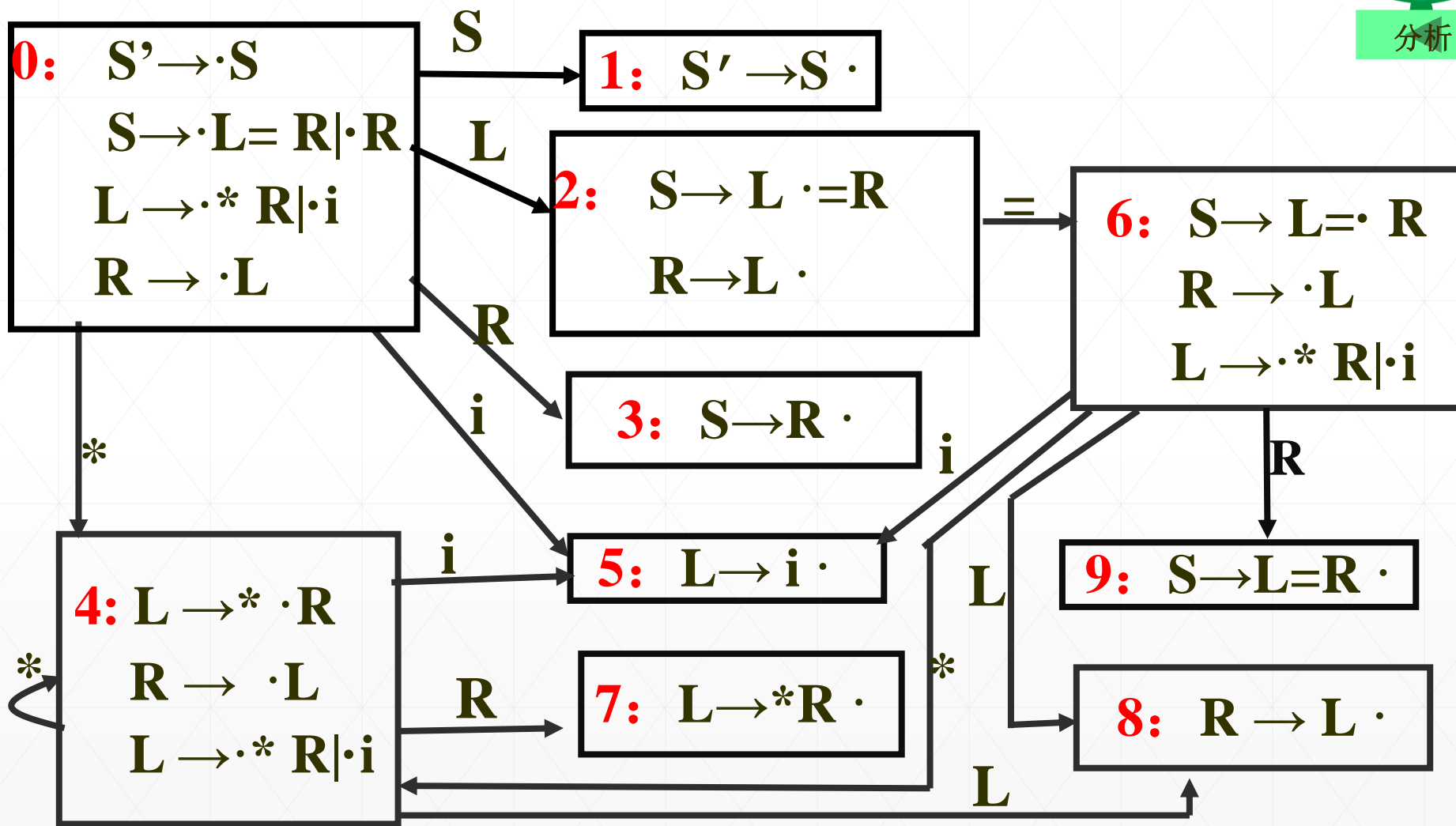
$$L \rightarrow *R \mid i$$

$$R \rightarrow L$$



分析

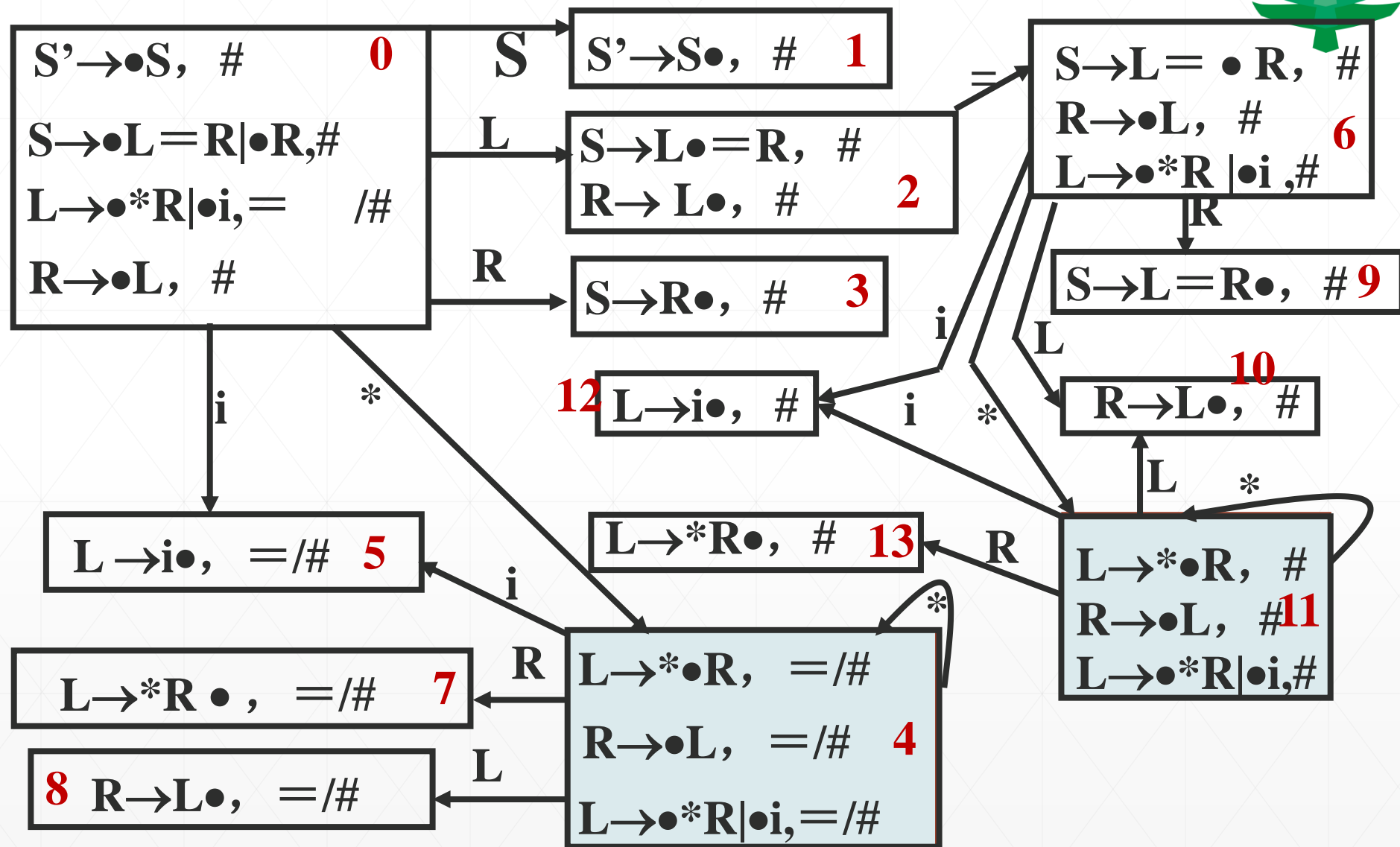
$G(S)$ 的识别LR(0)项目有效可归前缀的DFA:



$S' \rightarrow S$      $S \rightarrow L = R$  ① |  $R$  ②     $L \rightarrow * R$  ③ |  $i$  ④     $R \rightarrow L$  ⑤



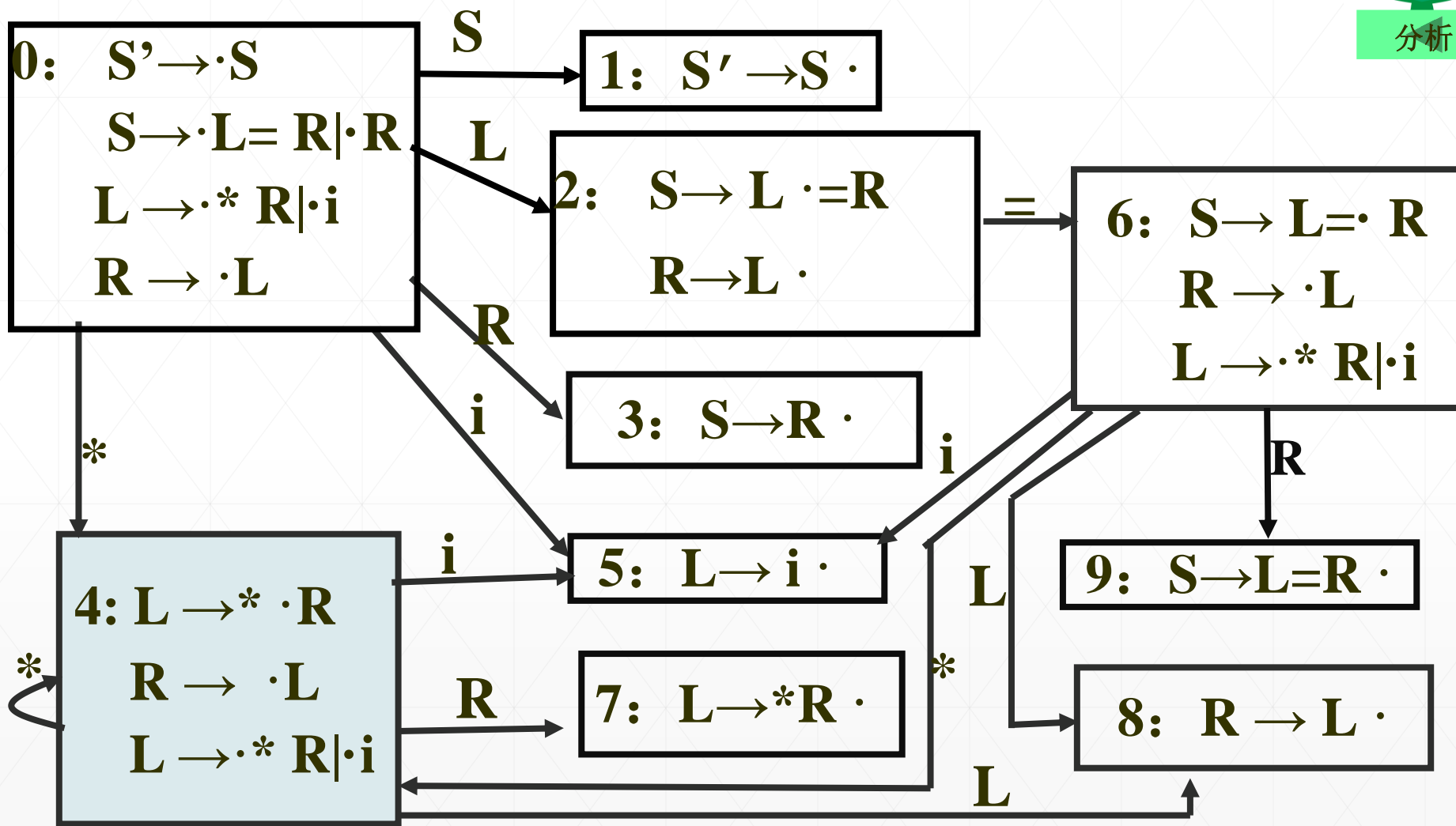
$G(S)$ 的识别LR(1)项目有效可归前缀的DFA:





分析

$G(S)$ 的识别LR(0)项目有效可归前缀的DFA:

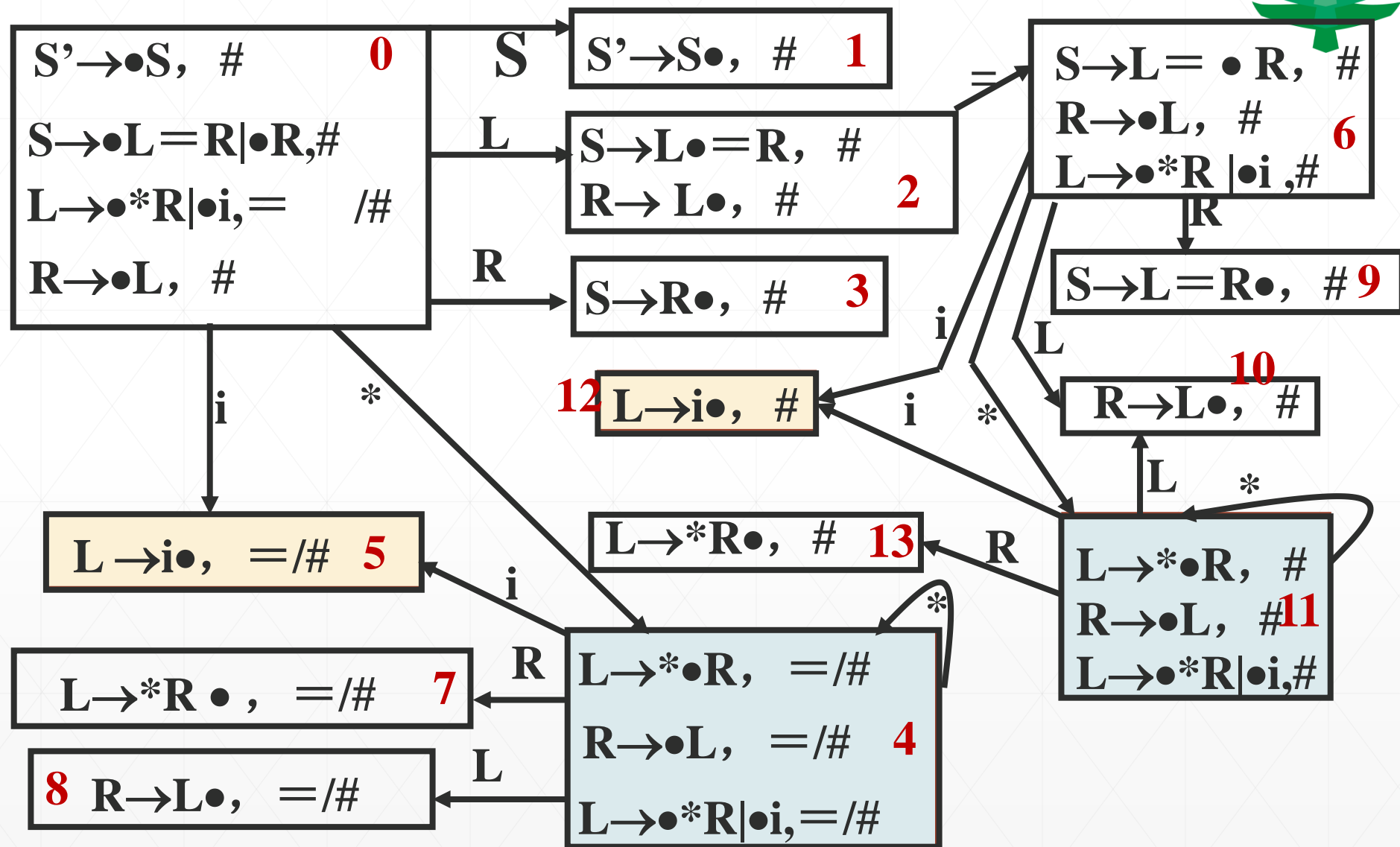


$S' \rightarrow S$      $S \rightarrow L = R$  ① |  $R$  ②     $L \rightarrow * R$  ③ |  $i$  ④     $R \rightarrow L$  ⑤





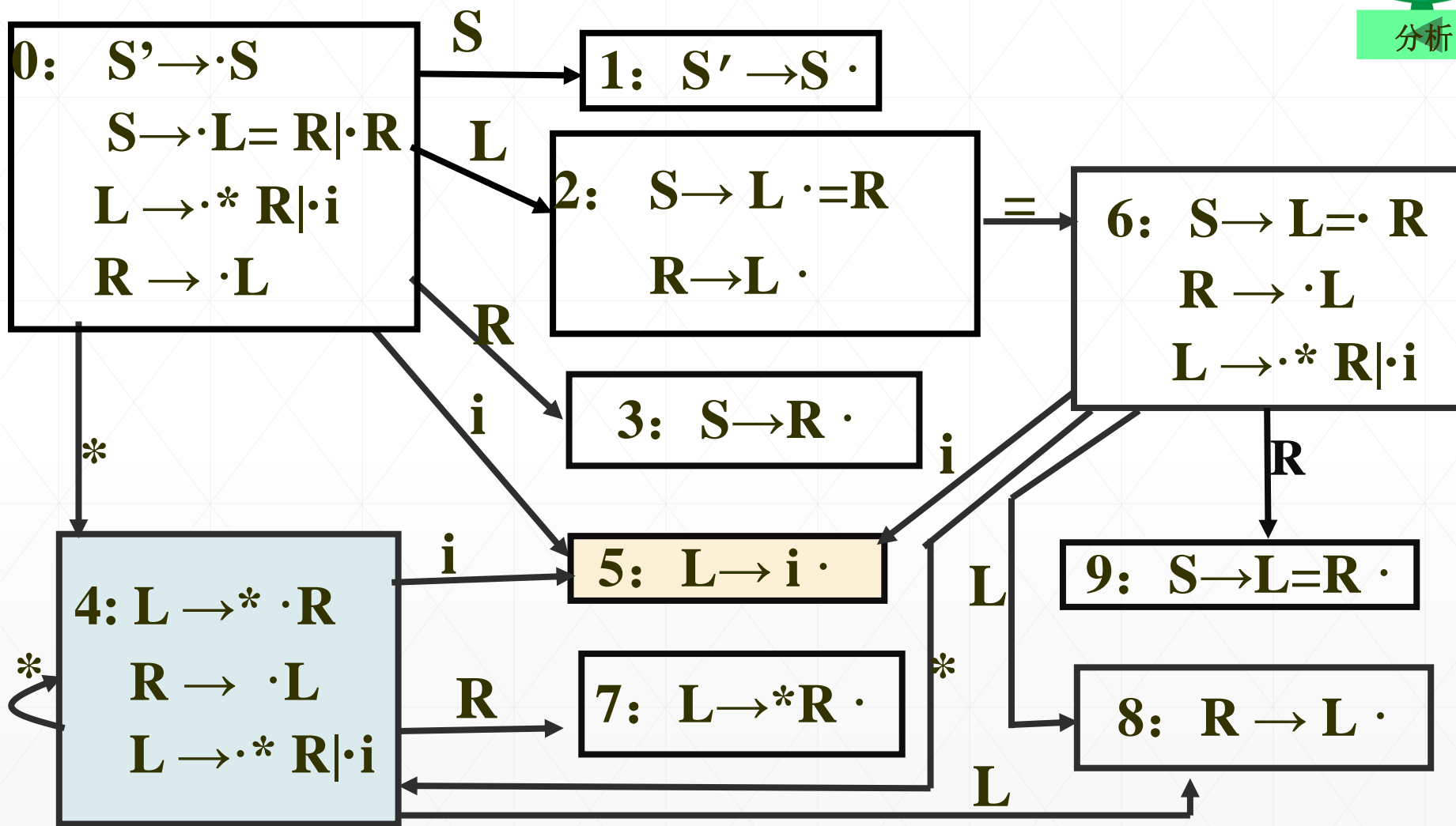
$G(S)$ 的识别LR(1)项目有效可归前缀的DFA:





分析

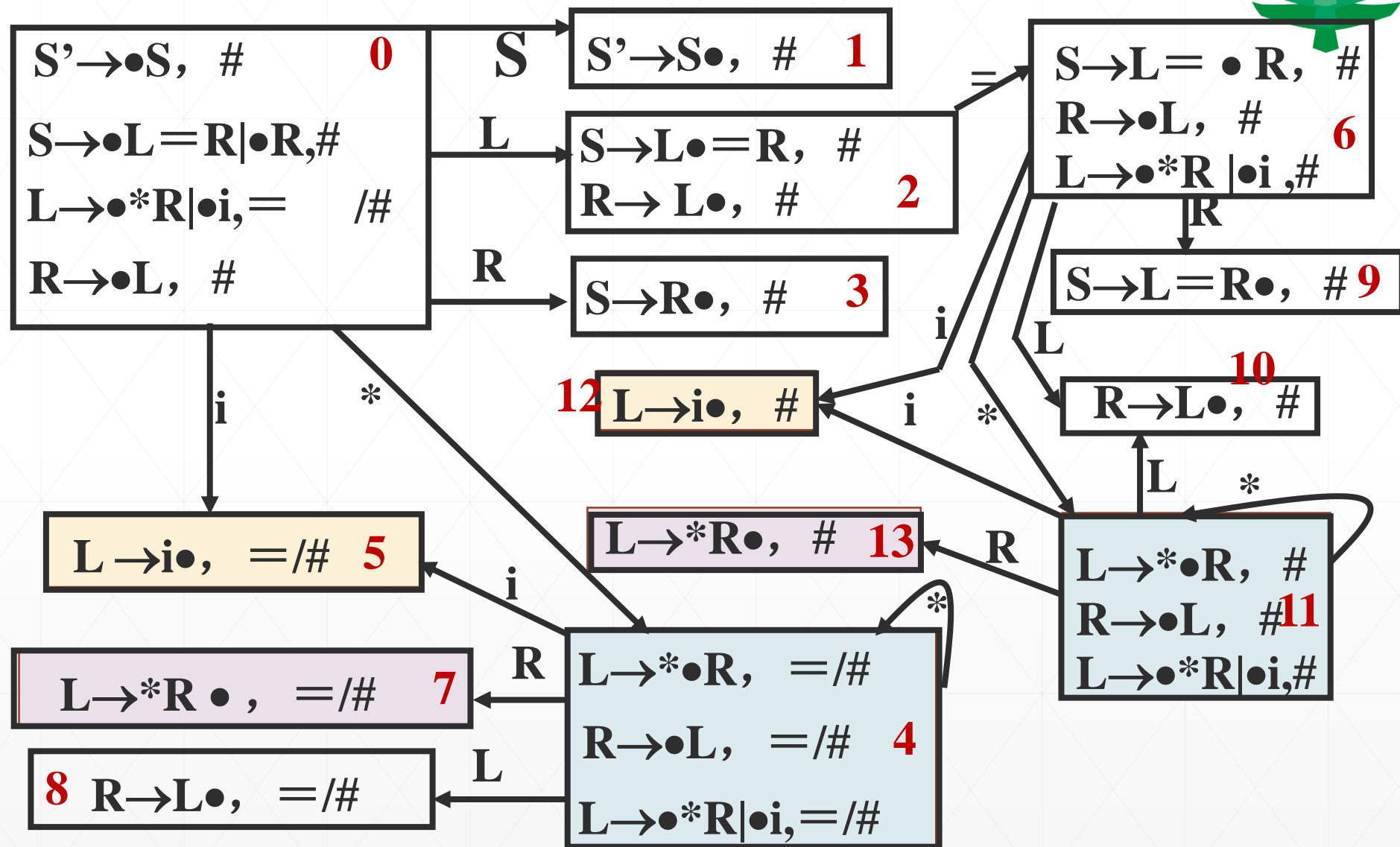
$G(S)$ 的识别LR(0)项目有效可归前缀的DFA:



$S' \rightarrow S$      $S \rightarrow L = R$  ① |  $R$  ②     $L \rightarrow * R$  ③ |  $i$  ④     $R \rightarrow L$  ⑤



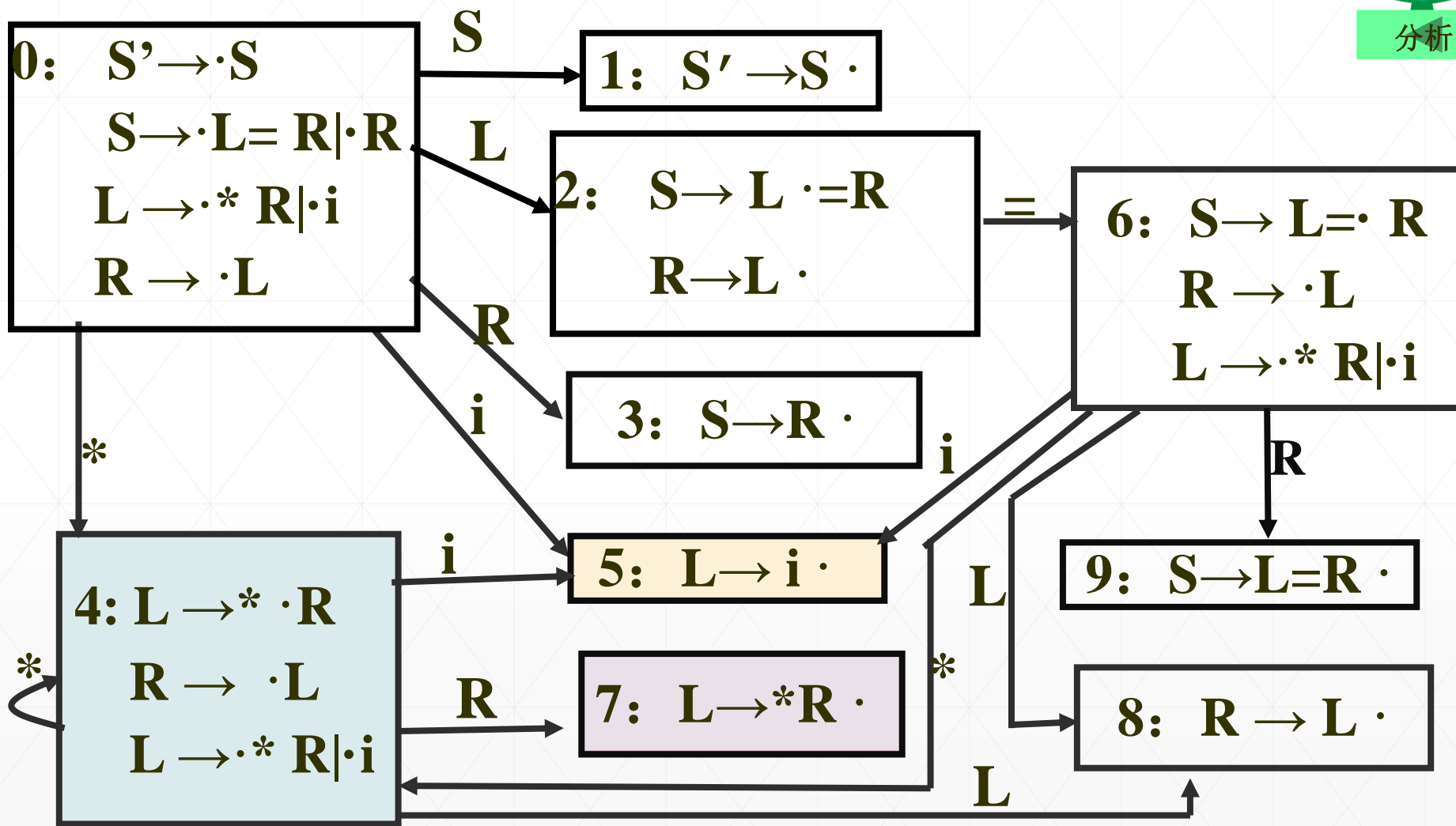
$G(S)$ 的识别LR(1)项目有效可归前缀的DFA:





分析

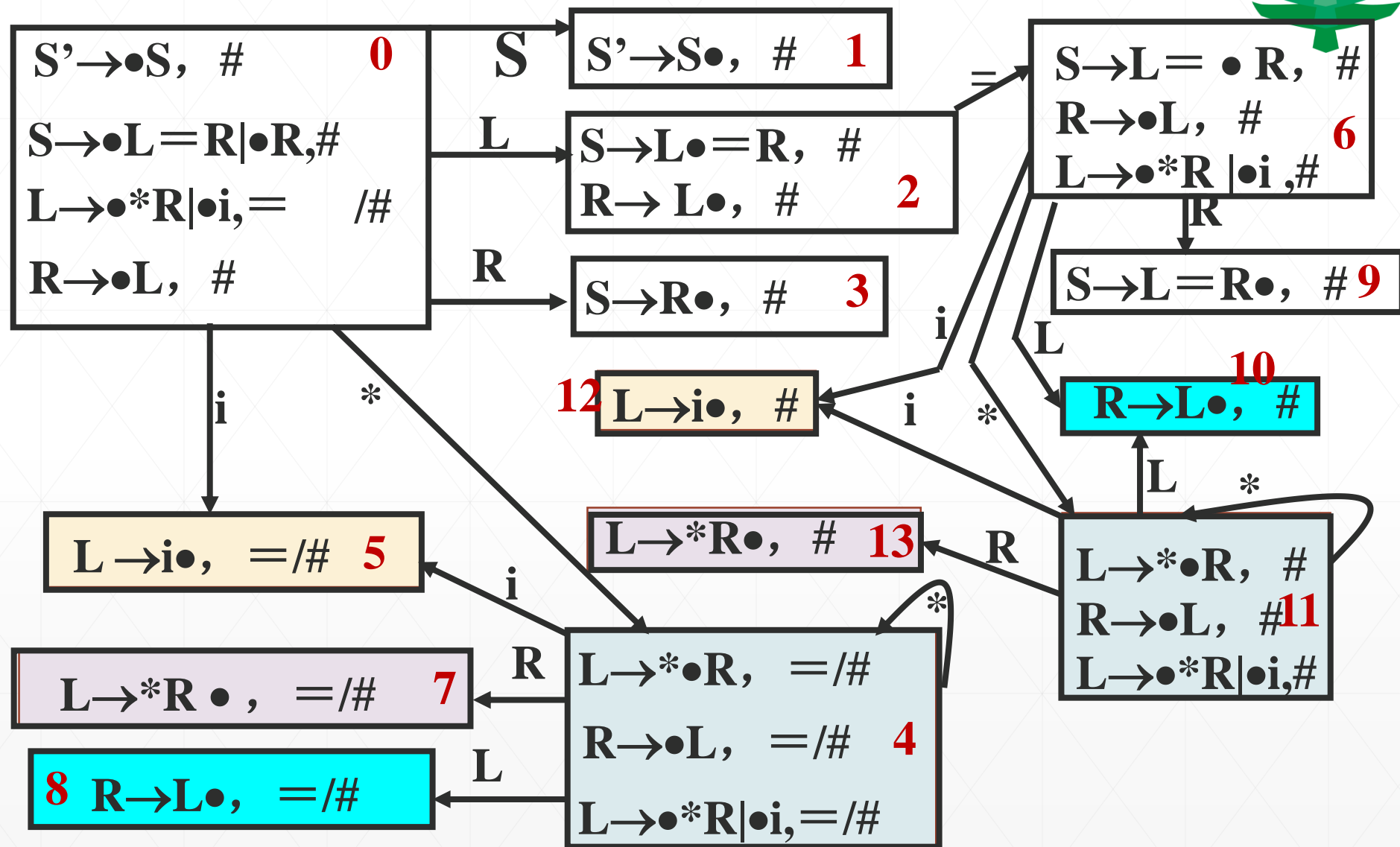
$G(S)$ 的识别LR(0)项目有效可归前缀的DFA:



$S' \rightarrow S$      $S \rightarrow L = R$  ① |  $R$  ②     $L \rightarrow * R$  ③ |  $i$  ④     $R \rightarrow L$  ⑤



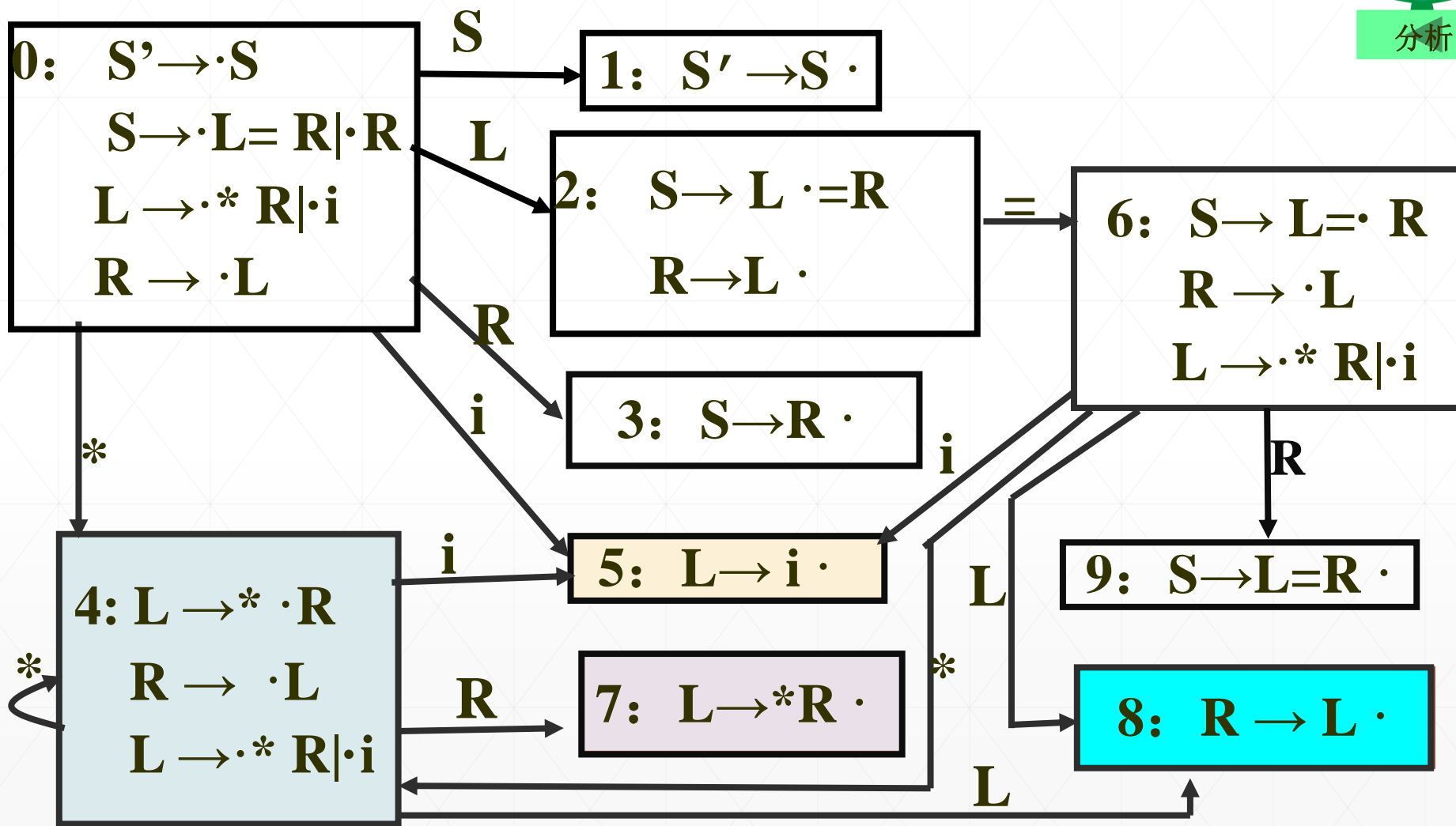
$G(S)$ 的识别LR(1)项目有效可归前缀的DFA:





分析

$G(S)$ 的识别LR(0)项目有效可归前缀的DFA:



$S' \rightarrow S$      $S \rightarrow L = R$  ① |  $R$  ②     $L \rightarrow * R$  ③ |  $i$  ④     $R \rightarrow L$  ⑤



# 对比文法 $G(S)$ 的识别LR(1)项目有效可归前缀DFA和识别LR(0)项目有效可归前缀的DFA

## LR(0)的DFA状态

4

分裂为

5

分裂为

7

分裂为

8

分裂为

## LR(1)的DFA状态

4、11

5、12

7、13

8、10

每个状态中  
项目仅搜索符  
不同，LR(0)  
项目相同



## ■ 定义（同心状态、同心项目集）

对文法 $G$ 的LR(1)项目集规范族(识别可归前缀的DFA), 若存在两个(或两个以上)项目集(状态) $I_0$ 、 $I_1$ , 其中的LR(0)项目相同, 仅搜索符不同, 则称 $I_0$ 、 $I_1$ 为 $G$ 的LR(1)的同心项目集(同心状态). 或称 $I_0$ 、 $I_1$ 具有相同的心.

合并LR(1)的同心项目集(状态) $\Rightarrow$  LALR (1)的C或DFA



减少DFA的状态数

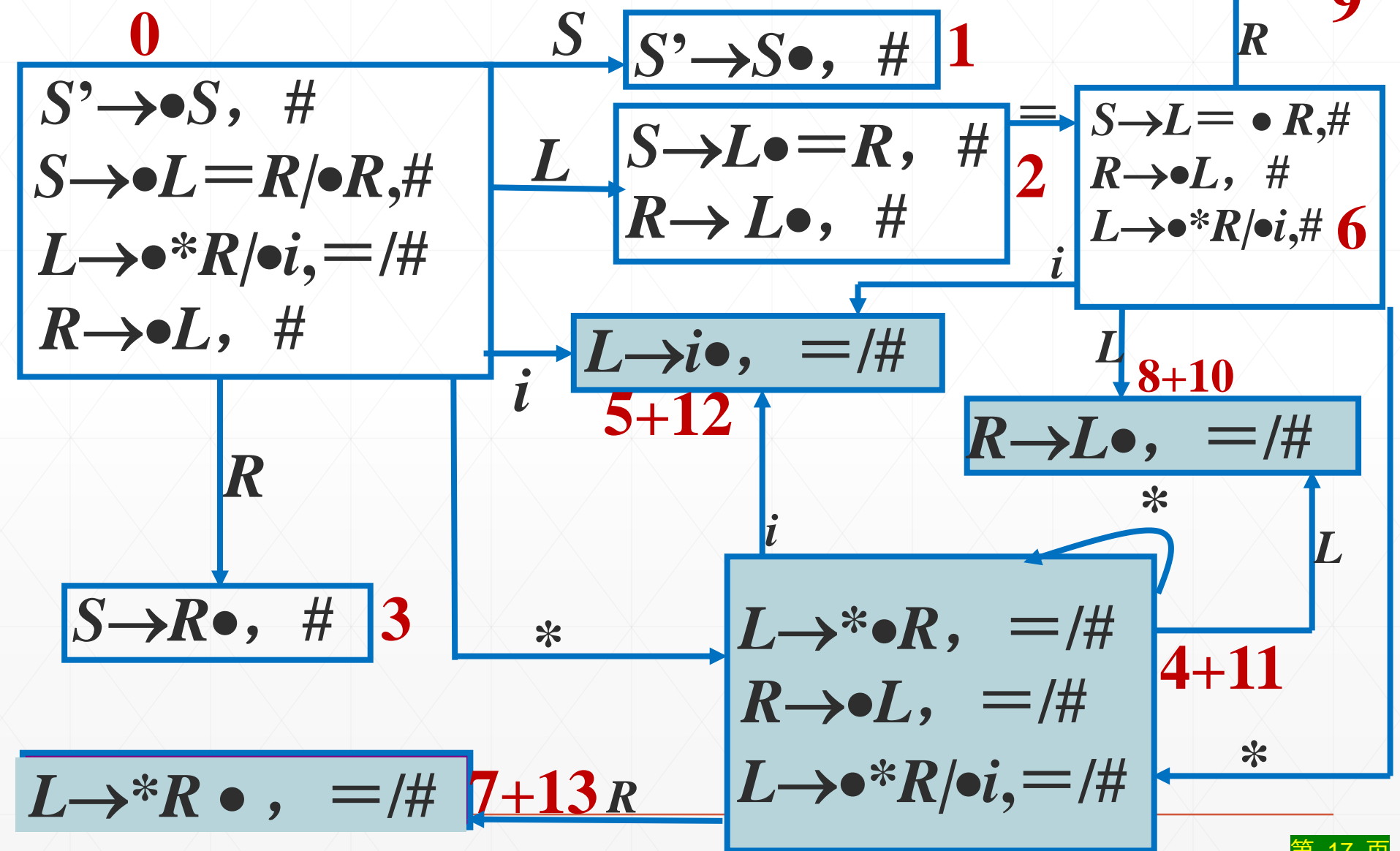


DFA的状态转移或GO函数做相应的修改。



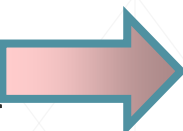


文法 $G(S')$ 的识别LALR(1)项目有效可归前缀的DFA





注意：

1. LR(1)的 $C(DFA)$ 无冲突  LALR(1)的 $C(DFA)$

LALR(1)的 $C(DFA)$  { 无冲突：构造LALR(1)分析表  
有冲突：构造LR(1)的分析表

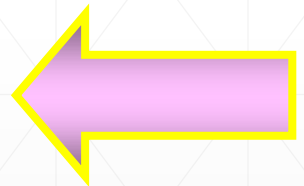
2. LALR(1)的 $C$ 无冲突，分析能力同LR(1)。  
确定串中错误前有时会比LR(1)多产生若干步归约。



## 5.7 LALR (1)分析

5.7.1 LALR( 1 )分析实现思想

5.7.2 LALR( 1 )分析表的构造





## ■ 算法5.11 ( LALR(1)分析表构造)

输入：拓广的文法 $G'$ 及文法 $G'$ 的LALR(1)项目集规范族 $C$ 和GO函数

输出：文法 $G'$  的LALR(1)分析表

方法：设 $C = \{I_0, I_1, \dots, I_n\}$ , 分析表的状态 =  $\{0, 1, \dots, n\}$ 。

①若 $GO(I_k, a) = I_j$ ,  $a \in V_T$ , 则置 $action(K, a) = S_j$ ;  
若 $GO(I_k, A) = I_j$ ,  $A \in V_N$ , 则置 $GOTO[K, A] = j$ 。

②若 $[A \rightarrow \alpha \cdot, a] \in I_k$ , 则置 $action(K, a) = r_j$ , 其中  
 $A \rightarrow \alpha$ 为文法 $G$ 的第 $j$ 个产生式并且 $A$ 不是文法的开始符号;

③ 若接受项目 $[S' \rightarrow S \cdot, \#] \in I_k$ ,  $S'$ 是文法的开始符号,  
则置 $action(K, \#) = acc$ ;

④分析表中不能用①至③规则填入信息的元素, 则置“出错标志”



## ■ 算法（LALR(1)分析表构造）

输入：拓广的文法 $G'$ 及文法 $G'$ 的识别LALR(1)项目有效  
可归前缀的DFA

输出：文法 $G'$ 的LALR(1)分析表

方法：设状态集 $Q = \{0, 1, \dots, n\}$  = 分析表的状态

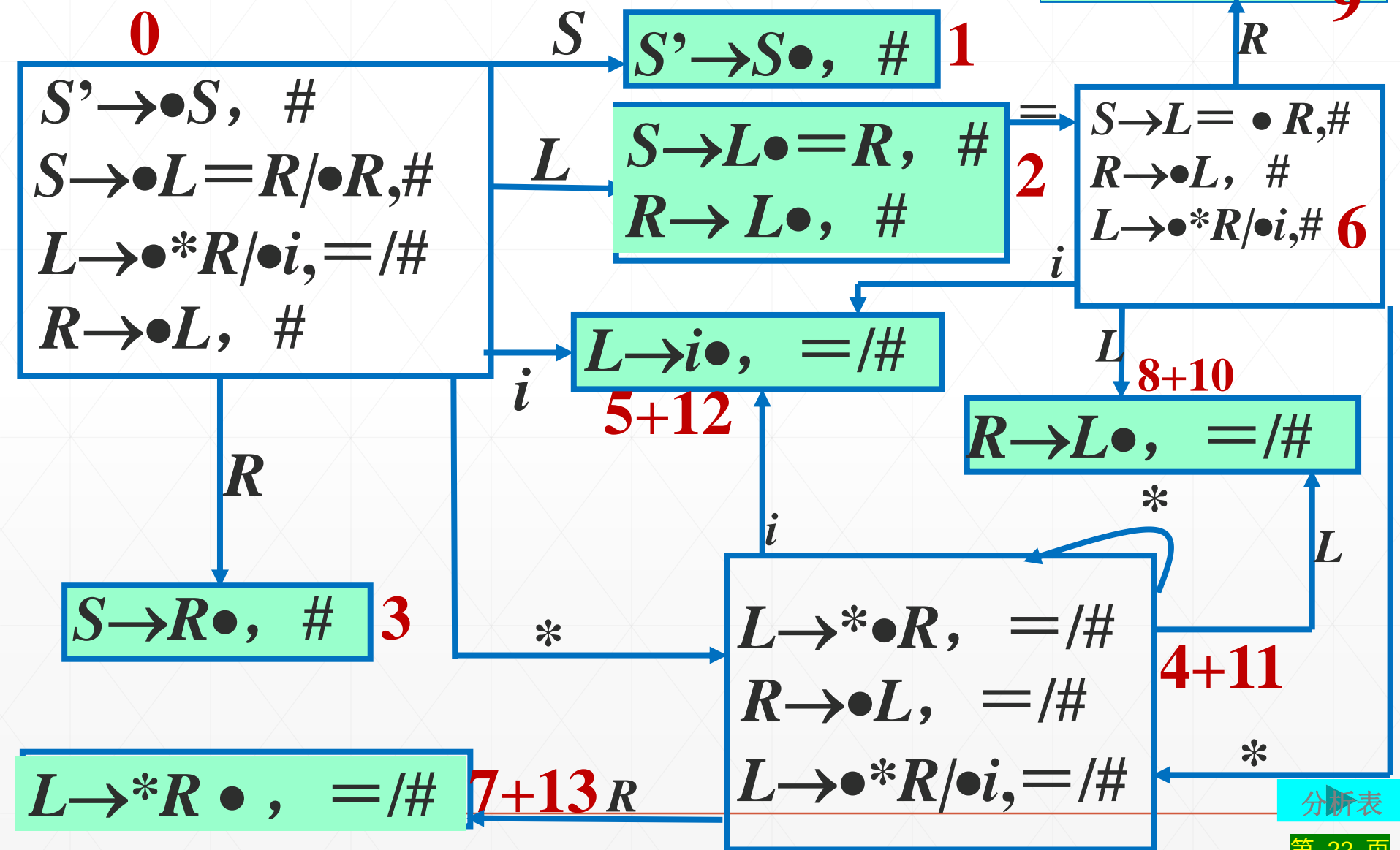
① 若 $f(k, a) = j$ ,  $a \in V_T$ , 则置  $\text{action}(k, a) = S_j$ ;  
若 $f(k, A) = j$ ,  $A \in V_N$ , 则置  $\text{GOTO}[K, A] = j$ 。

② 若  $[A \rightarrow \alpha \cdot, a] \in k$ , 则置  $\text{action}(K, a) = r_j$ , 其中  
 $A \rightarrow \alpha$  为文法 $G$ 的第 $j$ 个产生式并且 $A$ 不是文法的开始符号;

③ 若接受项目  $[S' \rightarrow S \cdot, \#] \in k$ ,  $S'$ 是文法的开始符号,  
则置  $\text{action}(K, \#) = \text{acc}$ ;

④ 分析表中不能用①至③规则填入信息的元素, 则置“出错标志”

# 文法 $G(S')$ 的识别LALR(1)项目有效可归前缀的DFA





# 文法G(S)LALR(1)分析表

state	Action				Goto		
	=	*	<i>i</i>	#	<i>L</i>	<i>R</i>	<i>S</i>
0		$S_{4+11}$	$S_{5+12}$		2	3	1
1				acc			
2	$S_6$			$r_5$			
3				$r_2$			
4+11		$S_{4+11}$	$S_{5+12}$		8+10	7+13	
5+12	$r_4$			$r_4$			
6		$S_{4+11}$	$S_{5+12}$		8+10	9	
7+13	$r_3$			$r_3$			
8+10	$r_5$			$r_5$			
9				$r_1$			

 $S' \rightarrow S$ 
 $S \rightarrow L=R \text{ ①} | R \text{ ②}$ 
 $L \rightarrow *R \text{ ③} | i \text{ ④}$ 
 $R \rightarrow L \text{ ⑤}$



## ■ 定义

按照LALR(1)的项目集规范族构造的文法 $G$ 的LALR (1)分析表，如果每个入口不含多重定义，则称文法 $G$ 为LALR(1) 文法。  
使用 LALR(1) 分析表的语法分析器称作LALR(1)分析器。





## 构造LALR(1)的分析表综述:

1. 构造文法 $G$ 的LR(1)的项目集规范族 $C(DFA)$

$$C = \{ I_0, I_1, I_2, \dots, I_n \}$$

2. 合并 $C$ 中的所有同心的LR(1)项目集, 用 $M$ 表示合并后的新项目集

$$M = \{ M_0, M_1, M_2, \dots, M_m \} \quad \text{其中: } m < n$$

$M$ 称为LALR(1)的项目集规范族。

3.  $M$ 若没有冲突, 据 $M$ 构造文法 $G$ 的LALR(1)的分析表。



LALR(1)的项目集规范族若存在冲突，只能是归约—归约冲突。 (尽管原来的LR(1)的C不冲突)

证明:

设原LR(1)的项目集规范族C中有如下项目集

$$\begin{array}{ll} I_k: [A \rightarrow \alpha \cdot, W_1] & I_j: [A \rightarrow \alpha \cdot, W_2] \\ [B \rightarrow \beta \cdot a \gamma, b] & [B \rightarrow \beta \cdot a \gamma, c] \end{array}$$

由于 $I_k$ 与 $I_j$ 均无冲突，故有

$$W_1 \cap \{a\} = \emptyset$$

$$W_2 \cap \{a\} = \emptyset$$

从而

$$(W_1 \cup W_2) \cap \{a\} = \emptyset$$



现将 $I_k$ 与 $I_j$ 合并, 有

$I_{k/j}$ :

$[A \rightarrow \alpha \bullet, W_1 \cup W_2]$

$[B \rightarrow \beta \bullet a \gamma, \{b\} \cup \{c\}]$

若此时 $I_{k/j}$ 有“移进 — 归约”冲突, 则必有

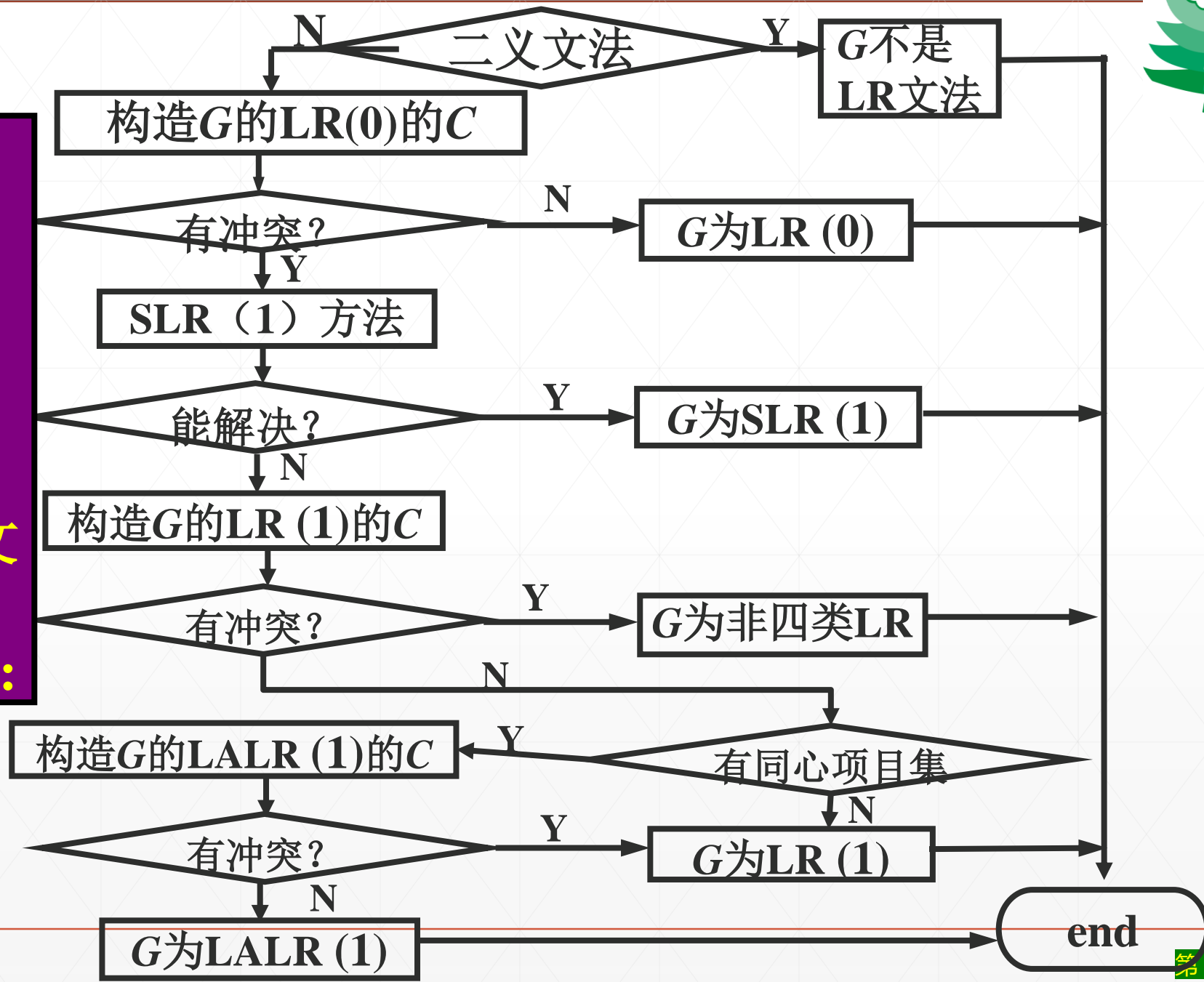
$$(W_1 \cup W_2) \cap \{a\} \neq \emptyset$$

与 $I_k$ 和 $I_j$ 无冲突的假设矛盾。

$\therefore$  合并同心集后不会引入新的“移进—归约”冲突。证毕。



判断  
一个  
文法  
属于  
那一  
类  
LR文  
法的  
步骤:





例:设有下列文法

$$S \rightarrow Aa \mid bAc \mid Bc \mid bBa$$

$$A \rightarrow d$$

$$B \rightarrow d$$

判断该文法是哪类LR文法?

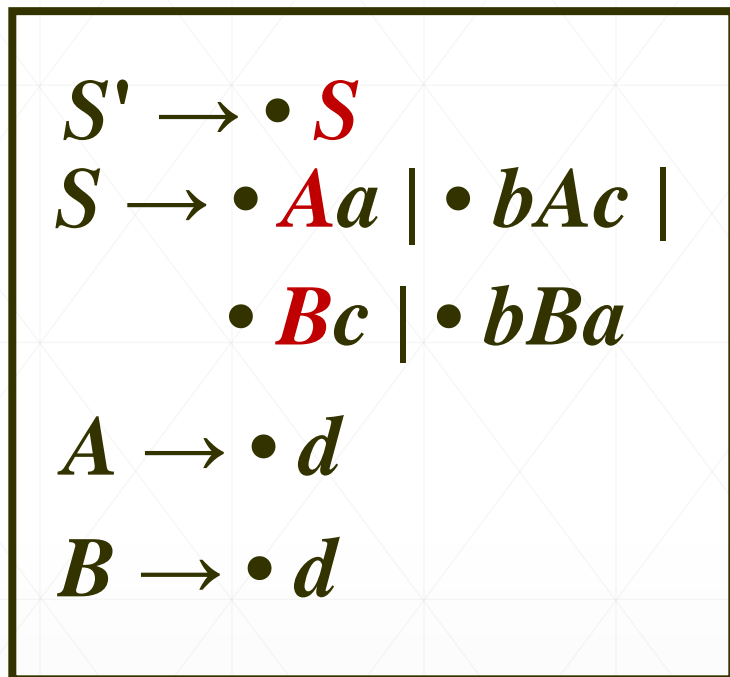
没有发现有二义性, 所以从LR(0)开始判断。

拓广文法:

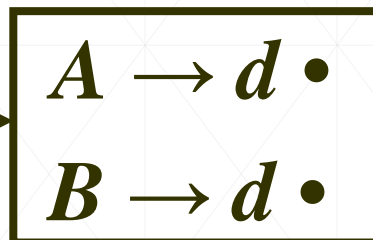
增加一产生式:  $S' \rightarrow S$



0

 $d$ 

1

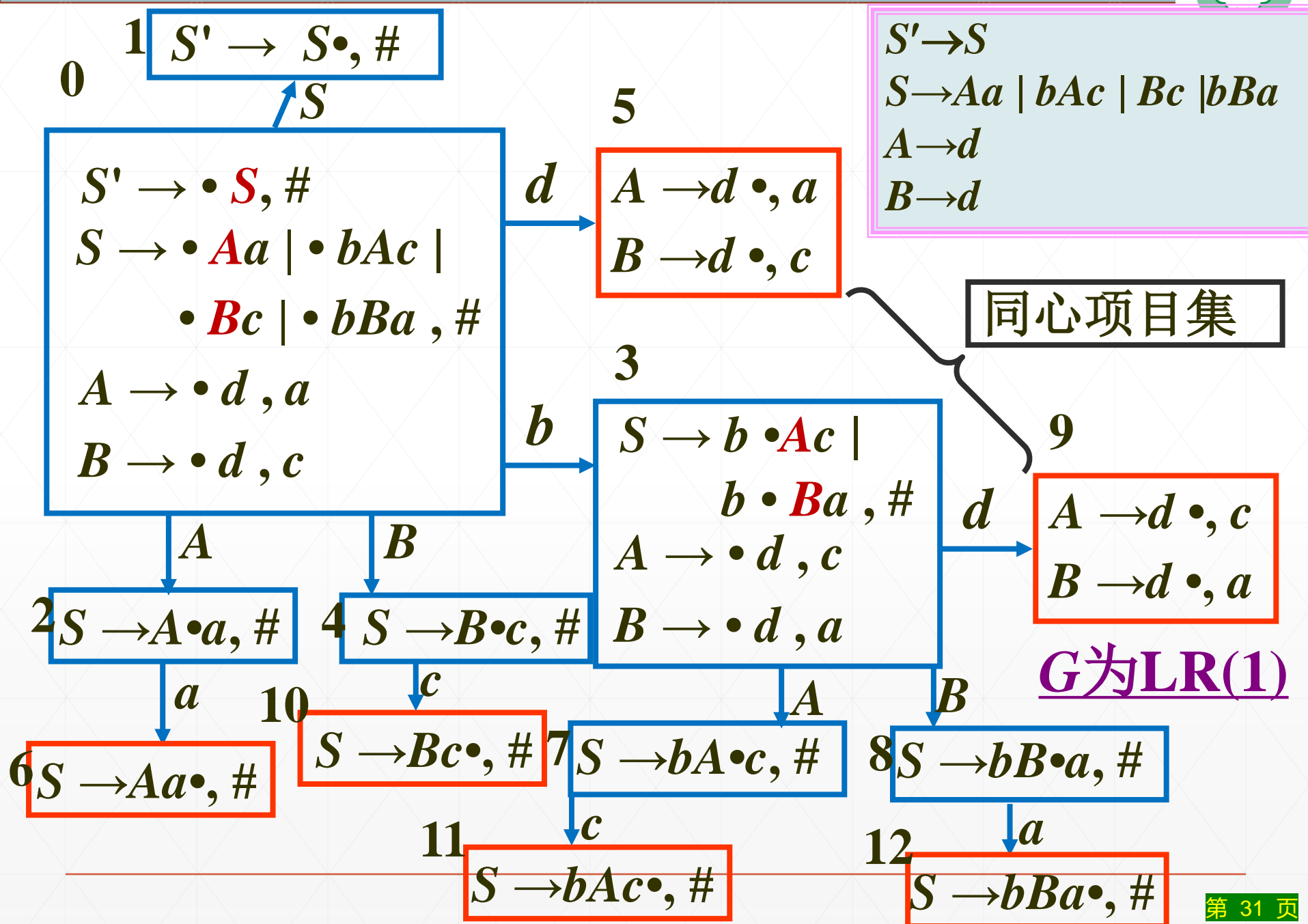


归约—归约冲突

G非LR(0)文法

用SLR(1)方法

 $\text{FOLLOW}(A) = \{a, c\}$  $\text{FOLLOW}(B) = \{a, c\}$  $\cap \neq \emptyset$ G非SLR(1)文法 $S' \rightarrow S$  $S \rightarrow Aa \mid bAc \mid Bc \mid bBa$  $A \rightarrow d$  $B \rightarrow d$





## 状态5和状态9合并

5+9

$A \rightarrow d \bullet, a/c$   
 $B \rightarrow d \bullet, c/a$

归约—归约冲突。

所以G不是LALR(1)



G为LR(1)





## 5.1 “移近—归约”分析法

## 5.3 LR分析鸟瞰

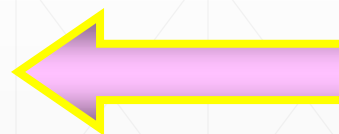
## 5.4 LR (0)分析

## 5.5 SLR (1)分析

## 5.6 LR (1)分析

## 5.7 LALR (1)分析

## 5.8 LR分析对二义文法的应用



## 5.9 LR分析的错误处理与恢复

## 5.10 语法分析器的自动生成与YACC(自学)



## ■ 定理

任何一个二义文法都不是一个**LR**文法。



二义文法会导致语法分析的二义性。



二义文法的有用之处在于可以缩小文法的规模。



例如,

$$G: E \rightarrow E+E \mid E * E \mid (E) \mid i$$

$$G': E \rightarrow T \mid E+T$$

$$T \rightarrow F \mid T * F$$

$$F \rightarrow (E) \mid i$$

例如,

$$G: S \rightarrow iSeS \mid iS \mid a$$

$$G': S \rightarrow \{iSeS\} \mid \{iS\} \mid a$$



例：设有文法 $G(E)$ ：

$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow (E)$$

$$E \rightarrow i$$

拓广文法：

$$E' \rightarrow E$$

$$E \rightarrow E + E \quad \textcircled{1}$$

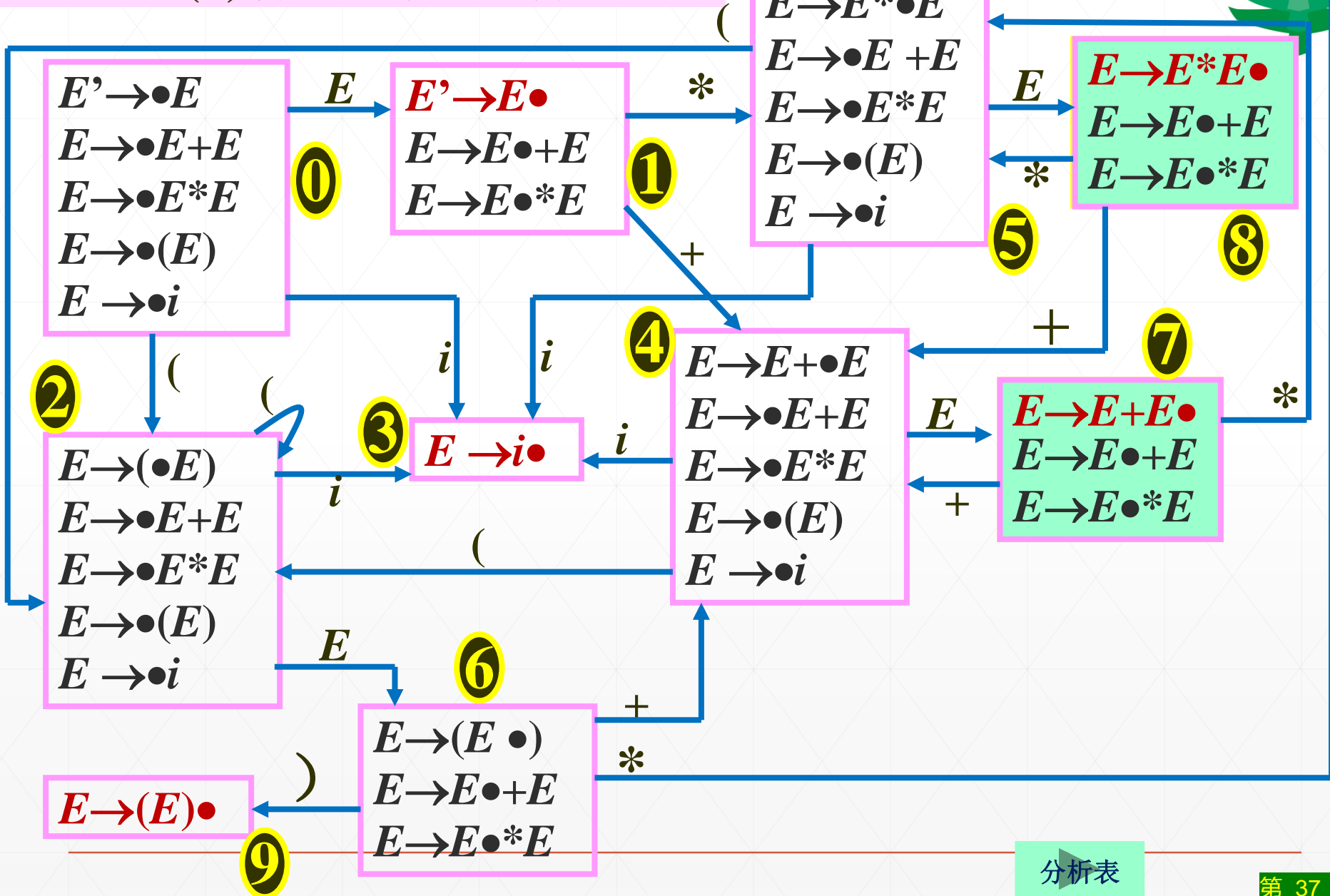
$$E \rightarrow E * E \quad \textcircled{2}$$

$$E \rightarrow (E) \quad \textcircled{3}$$

$$E \rightarrow i \quad \textcircled{4}$$



# 识别LR(0)有效可归前缀的DFA





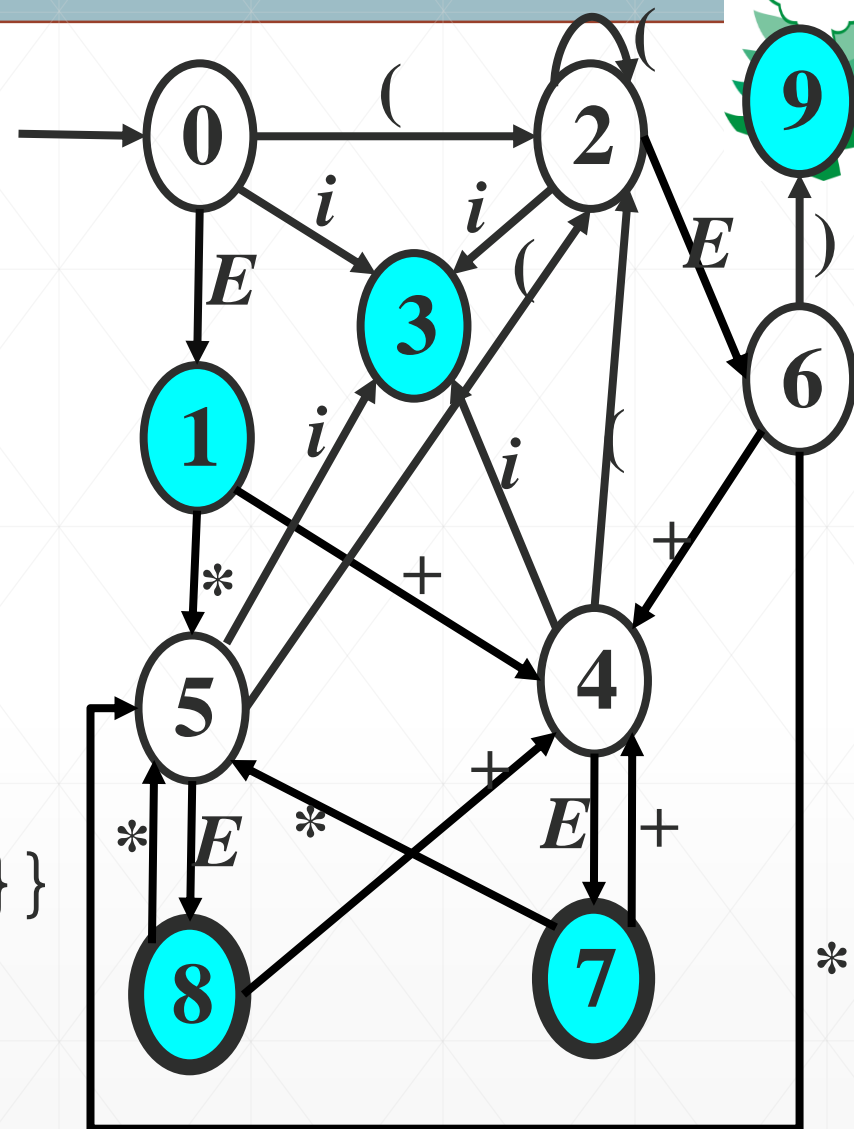
⑦

 $E \rightarrow E + E \bullet$  $E \rightarrow E \bullet + E$  $E \rightarrow E \bullet * E$ 

⑧

 $E \rightarrow E * E \bullet$  $E \rightarrow E \bullet + E$  $E \rightarrow E \bullet * E$ 

**FOLLOW(E) =**  
 $\{+, *, ), \#\} \cap \{\{+\} \cup \{*\}\}$   
 $\neq \Phi$

 $E' \rightarrow E$  $E \rightarrow E + E$ 

①

 $E \rightarrow E * E$ 

②

 $E \rightarrow (E)$ 

③

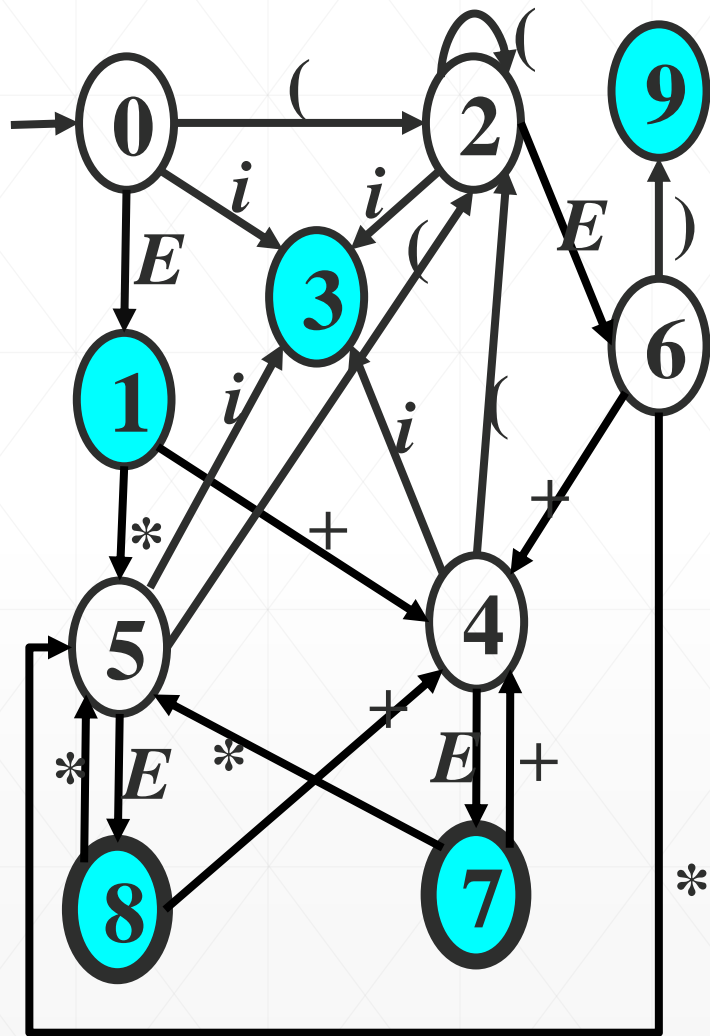
 $E \rightarrow i$ 

④



$E' \rightarrow E$     ①  $E \rightarrow E+E$     ②  $E \rightarrow E^*E$     ③  $E \rightarrow (E)$     ④  $E \rightarrow i$

## 文法G(E)的LR分析表



状态	STATION						GOTO
	+	*	(	)	i	#	
0			s <sub>2</sub>		s <sub>3</sub>		1
1	s <sub>4</sub>	s <sub>5</sub>				acc	
2			s <sub>2</sub>		s <sub>3</sub>		6
3	r <sub>4</sub>	r <sub>4</sub>	r <sub>4</sub>	r <sub>4</sub>	r <sub>4</sub>	r <sub>4</sub>	
4			s <sub>2</sub>		s <sub>3</sub>		7
5			s <sub>2</sub>		s <sub>3</sub>		8
6	s <sub>4</sub>	s <sub>5</sub>		s <sub>9</sub>			
7	r <sub>1</sub>	s <sub>5</sub>	r <sub>1</sub>	r <sub>1</sub>	r <sub>1</sub>	r <sub>1</sub>	
8	r <sub>2</sub>	r <sub>2</sub>	r <sub>2</sub>	r <sub>2</sub>	r <sub>2</sub>	r <sub>2</sub>	
9	r <sub>3</sub>	r <sub>3</sub>	r <sub>3</sub>	r <sub>3</sub>	r <sub>3</sub>	r <sub>3</sub>	

1:  $E' \rightarrow E \cdot$     3:  $E \rightarrow i \cdot$     7:  $E \rightarrow E + E \cdot$     8:  $E \rightarrow E * E \cdot$     9:  $E \rightarrow (E) \cdot$



\$:  $i+i+i \#$

↑  
**F**

$\text{action}(7, +) = r_2$

7	$E$	<div> <div>← <b>P</b></div> <div>} <math>E</math></div> </div>
4	+	
1	$E$	
0	#	

\$:  $i+i(*)i \#$

↑  
**F**

$\text{action}(7, *) = s_5$

5	*	← <b>P</b>
7	$E$	
4	+	
1	$E$	
0	#	





例：设有文法G:

$$S \rightarrow iSeS \mid iS \mid a \quad \textcircled{1} \mid \textcircled{2} \mid \textcircled{3}$$

其中:

$i$  : if  $e_r$  then

$e$ : else

$a, S$ : 语句

拓广文法G:

$$S' \rightarrow S$$

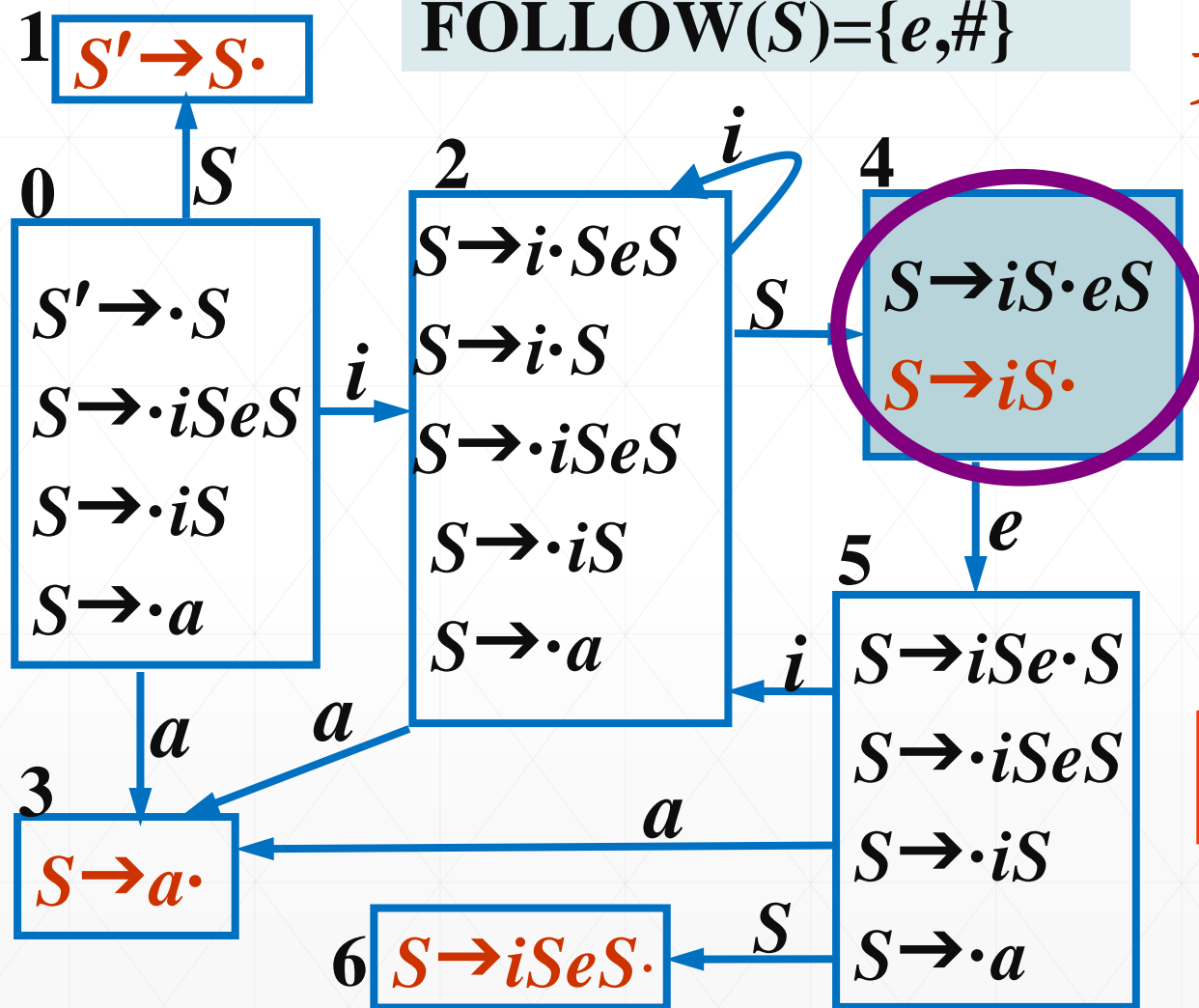
$$S \rightarrow iSeS \mid iS \mid a \quad \textcircled{1} \mid \textcircled{2} \mid \textcircled{3}$$



# 文法G的识别LR(0)项目有效可归前缀的DFA

$\text{FOLLOW}(S) = \{e, \#\}$

文法G的LR分析表



	<i>i</i>	<i>e</i>	<i>a</i>	#	<i>S</i>
0	S <sub>2</sub>		S <sub>3</sub>		1
1				acc	
2	S <sub>2</sub>		S <sub>3</sub>		4
3	r <sub>3</sub>	r <sub>3</sub>	r <sub>3</sub>	r <sub>3</sub>	
4	r <sub>2</sub>	S <sub>5</sub>	r <sub>2</sub>	r <sub>2</sub>	
5	S <sub>2</sub>		S <sub>3</sub>		6
6	r <sub>1</sub>	r <sub>1</sub>	r <sub>1</sub>	r <sub>1</sub>	

$S' \rightarrow S$

$S \rightarrow iSeS \mid iS \mid a$  ①|②|③



$$S' \rightarrow S$$

$$S \rightarrow iSeS \mid iS \mid a \quad \textcircled{1} \mid \textcircled{2} \mid \textcircled{3}$$

对  $\$ = iiaea\#$  进行分析

if $e_1$ then	if $e_2$ then	$S_1$	else	$S_2$
$i$	$i$	$a$	$e$	$a$

$\$ : i i a e a \#$

↑ **F**

$\text{action}(4, e) = S_5$

5	$e$
4	$S$
2	$i$
2	$i$
0	$\#$

**F++;P++**

← **P**



**5.1 “移近—归约”分析法**

**5.3 LR分析鸟瞰**

**5.4 LR (0)分析**

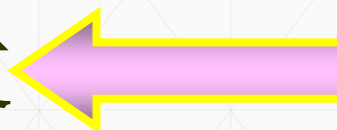
**5.5 SLR (1)分析**

**5.6 LR (1)分析**

**5.7 LALR (1)分析**

**5.8 LR分析对二义文法的应用**

**5.9 LR分析的错误处理与恢复**



**5.10 语法分析器的自动生成与YACC(自学)**



## ■ 错误存在的必然性

问题复杂性、程序员素质、输入错、系统环境生疏 ...

## ■ 编译中的错误种类

1. 词法错误；
2. 语法错误；
3. 语义错误(静态、动态)；
4. 违反环境限制的错误；



## ■ 错误处理基本目标

1. 清晰准确地**报告**错误（错误的定性和定位）；
2. 迅速从每个错误中**恢复**过来，以便诊断后面的错误；



## ■ 错误处理与恢复策略

### 1. 局部化恢复

### 2. 出错产生式

扩充语言的文法，增加产生错误结构的产生式。分析中可以直接识别处理错误。

### 3. 全局纠正

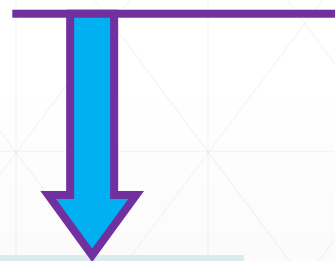
获取全局最小代价纠正。



## ■ 局部化恢复的一般性技术

### 1. 紧急恢复方式

编译器每次发现错误时，抛弃一个输入符号，直到输入符号属于某个指定的同步符号集合为止。



例如，界限符：

； 、 }、 end ...





**思想：** 在分析到某一含有错误的**短语**时，分析程序认为含有错误的短语是由某一非终结符 $A$ 所推导出的，短语的一部分已处理，放在栈顶部，剩下未处理的在输入串，分析程序跳过这些剩余符号，直至找到 $A$ 的跟随字符为止，同时把栈顶内容逐个移去，直至找到某个状态 $q$ ， $GOTO(q,A)$ 对应一新状态，将 $GOTO(q,A)$ ， $A$ 压入栈。



## 2. 短语级恢复

发现错误时，对剩余符号串作局部校正。

使用可以使编译器继续工作的输入串代替剩余输入的前缀。

在输入串的出错点采用插入、删除或修改的方法。

\* 关键：选择合适的替换串。

$(i+i;$   $\Rightarrow$   $(i+i)$ ;

$i\ i*i;$   $\Rightarrow$   $i+i*i;$



## ■ 错误处理基本目标

1. 清晰准确地**报告**错误（错误的定性和定位）；
2. 迅速从每个错误中**恢复**过来，以便诊断后面的错误；

## ■ 错误恢复的困难

1. 源程序书写的灵活性；
2. 错误发生的随机性；
3. 不使正确程序的处理效率降低。
4. 可能校正途径的多样性。



**例:**设有文法 $G(E)$ :

$$E' \rightarrow E$$

$$E \rightarrow E + E \quad \textcircled{1}$$

$$E \rightarrow E * E \quad \textcircled{2}$$

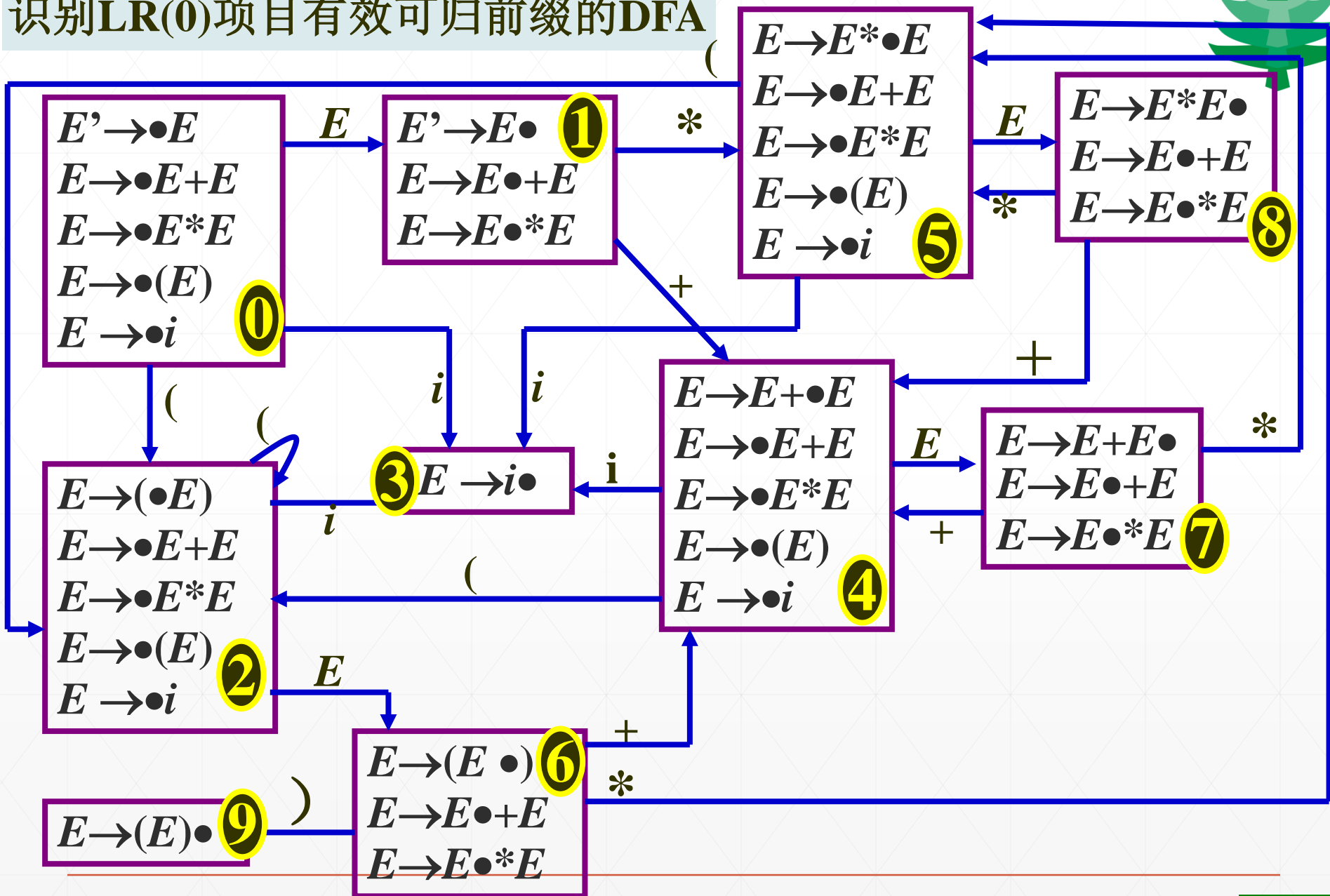
$$E \rightarrow (E) \quad \textcircled{3}$$

$$E \rightarrow i \quad \textcircled{4}$$

构造文法 $G(E)$ 的带出错处理的LR分析表



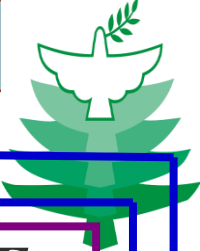
# 识别LR(0)项目有效可归前缀的DFA



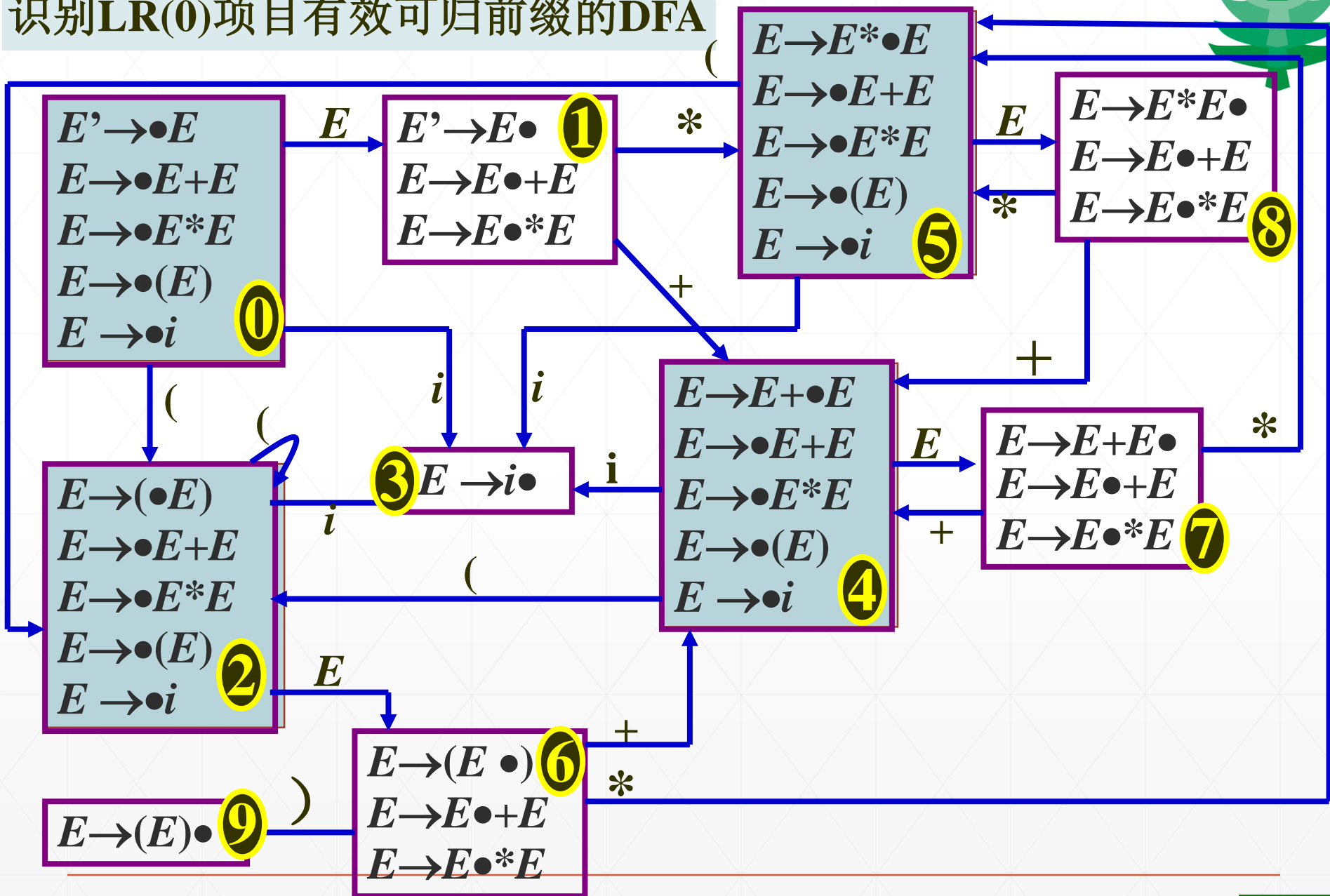


# 文法G(E)的LR分析表

状态	ACTION						GOTO
	<i>i</i>	+	*	(	)	#	<i>E</i>
0	S3			S2			1
1		S4	S5			acc	
2	S3			S2			6
3	r4	r4	r4	r4	r4	r4	
4	S3			S2			7
5	S3			S2			8
6		S4	S5		S9		
7	r1	r1	S5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	



# 识别LR(0)项目有效可归前缀的DFA





**e1:**

处于状态0、2、4、5时，要求输入符号为运算对象 *i* 或 (*E*)，此时若遇到+、\*、#，则调用e1。

**e1功能:**

将假设的 *i* 和状态3入栈；( shift )

**e1出错信息:**

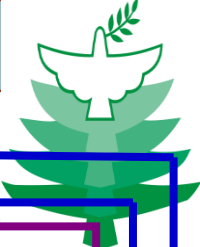
缺少运算对象。



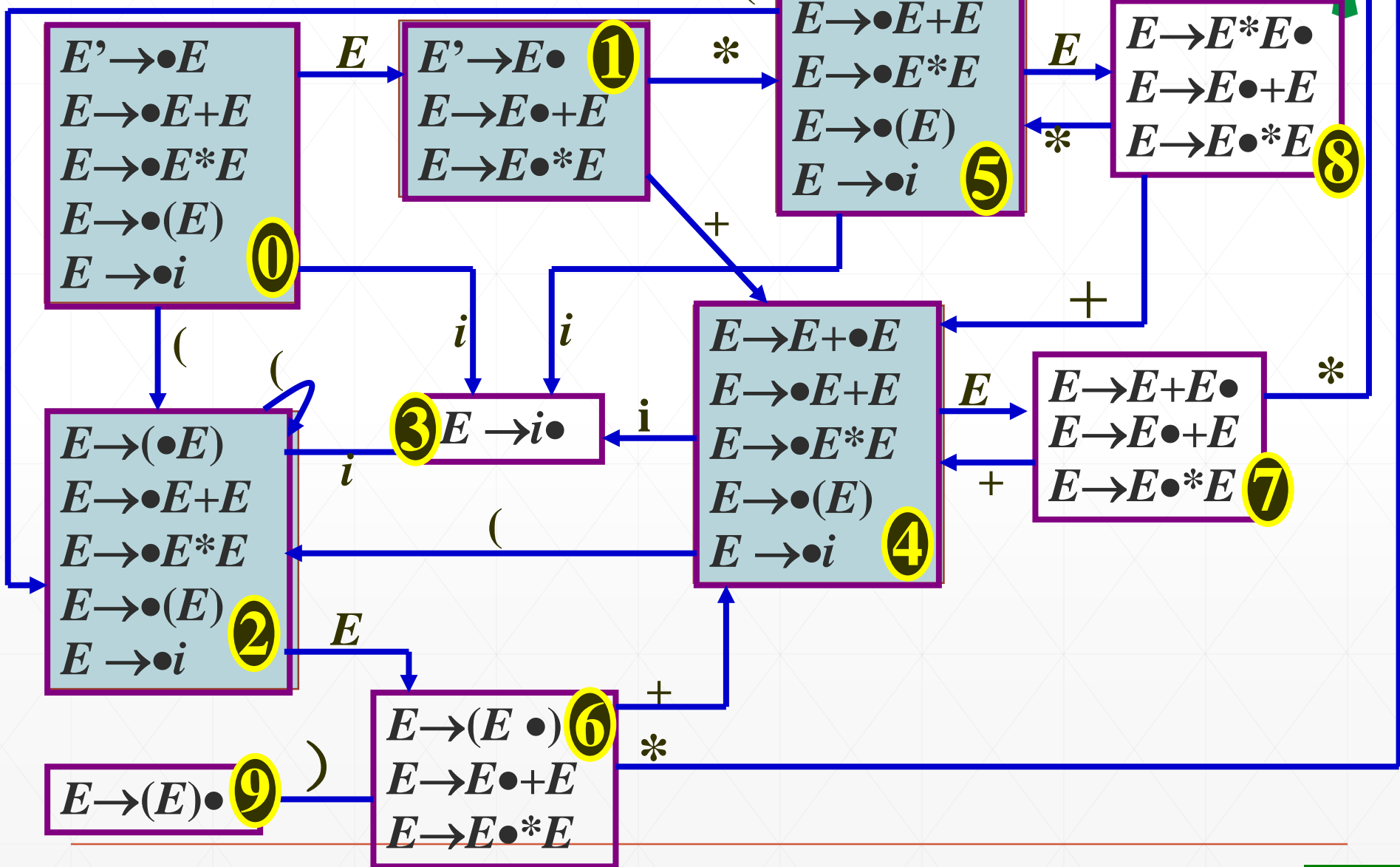


# 文法G(E)的LR分析表

状态	ACTION						GOTO
	<i>i</i>	+	*	(	)	#	<i>E</i>
0	S3	e1	e1	S2		e1	1
1		S4	S5			acc	
2	S3	e1	e1	S2		e1	6
3	r4	r4	r4	r4	r4	r4	
4	S3	e1	e1	S2		e1	7
5	S3	e1	e1	S2		e1	8
6		S4	S5		S9		
7	r1	r1	S5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	



# 识别LR(0)项目有效可归前缀的DFA





**e2:**

处于状态0、1、2、4、5时，若遇到“）”，则调用e2。

**e2功能:**

删除输入的“）”； **(F++)**

**e2出错信息:**

“）”不配对。

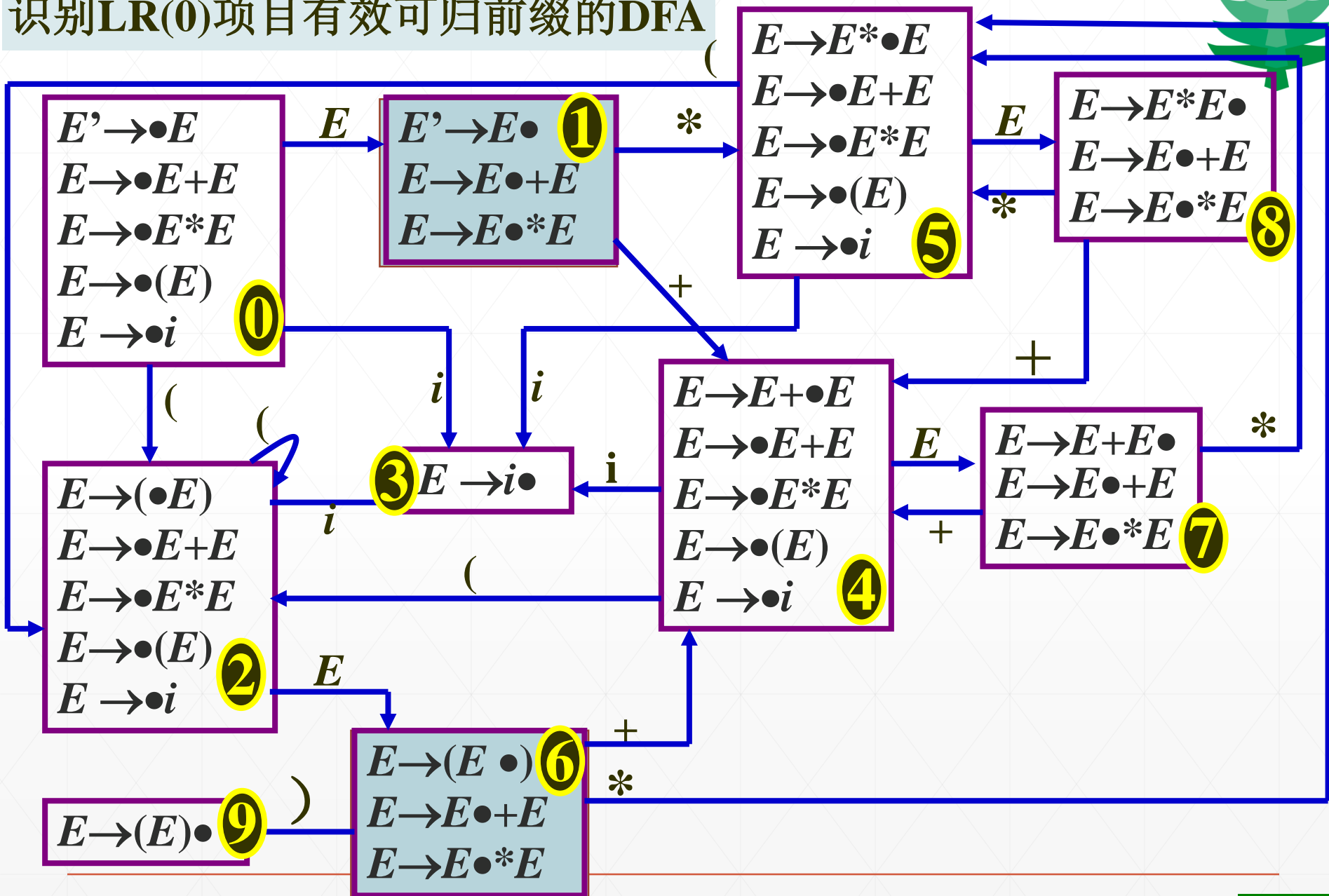


# 文法G(E)的LR分析表

状态	ACTION						GOTO
	<i>i</i>	+	*	(	)	#	<i>E</i>
0	S3	e1	e1	S2	e2	e1	1
1		S4	S5		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		S4	S5		S9		
7	r1	r1	S5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	



# 识别LR(0)项目有效可归前缀的DFA





**e3:**

处于状态1或6时，期望下面输入符为运算符或“)”，但遇到“*i*”或“(”时，则调用e3。

**e3功能:**

将假设的“+”和状态4入栈, (**P++**) ;

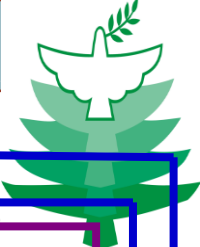
**e3出错信息:**

缺少运算符。

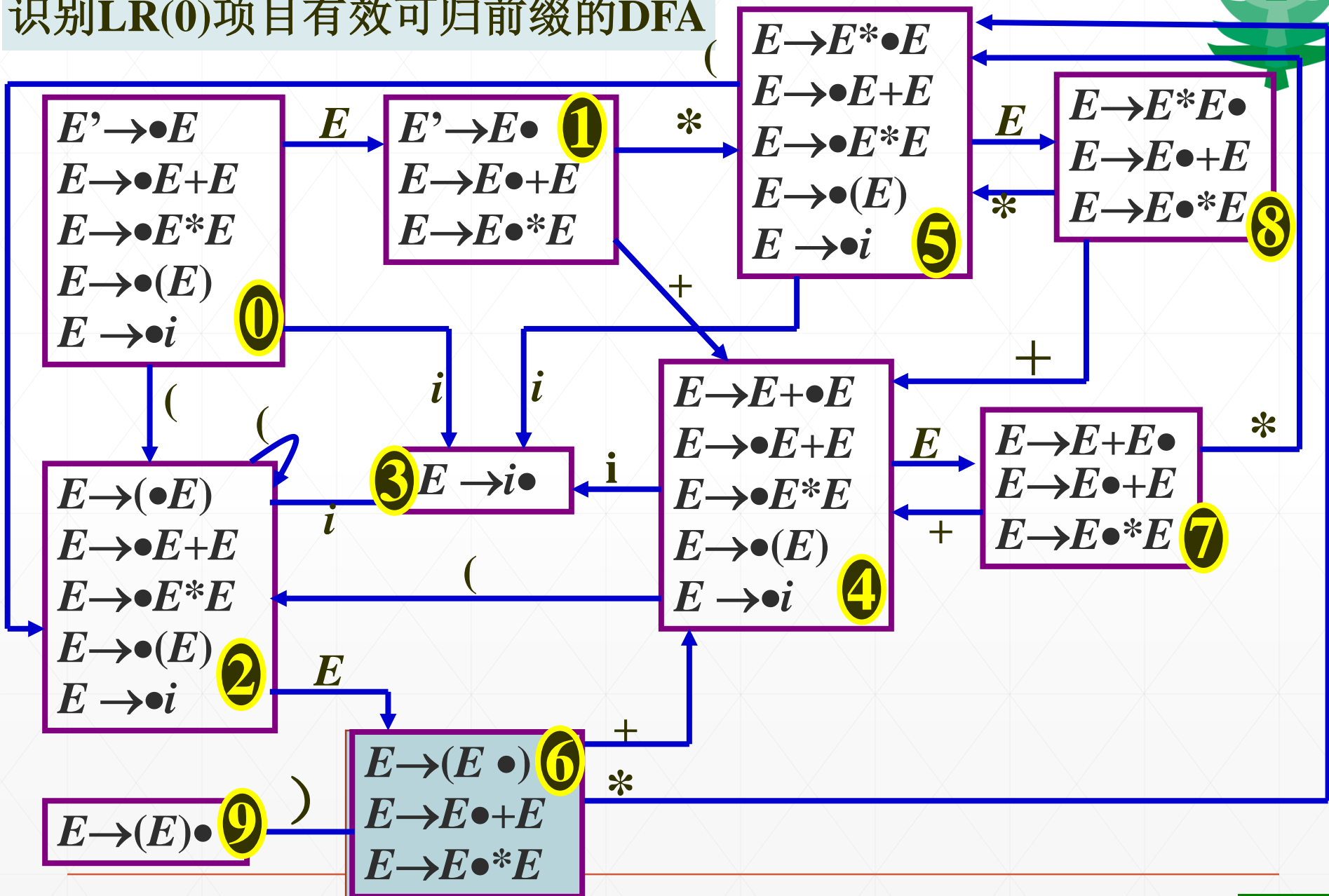


# 文法G(E)的LR分析表

状态	ACTION						GOTO
	<i>i</i>	+	*	(	)	#	<i>E</i>
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		
7	r1	r1	S5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	



# 识别LR(0)项目有效可归前缀的DFA







**e4:**

处于状态6时，期望下面输入符为运算符或“）”，但遇到“#”时，则调用e4。

**e4功能:**

将假设的“）”和状态9入栈, (**P++**) ;

**e4出错信息:**

缺少“）”。



# 文法G(E)的LR分析表

状态	ACTION						GOTO
	<i>i</i>	+	*	(	)	#	
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	



\$1: ( *i+i* #  $\xrightarrow{\text{blue arrow}}$  \$: ( *i+i* ) #  $\xrightarrow{\text{yellow arrow}}$  action(9, #) = r<sub>3</sub>

$\uparrow$   
**F**

$\uparrow$   
**F**

**E**

$\leftarrow$  **P**

<b>E</b> {	7	<i>E</i>	$\leftarrow$ <b>P</b>
	4	+	
	6	<i>E</i>	
	2	(	
	0	#	

9	)	$\leftarrow$ <b>P</b>
6	<i>E</i>	
2	(	
0	#	

goto(2, *E*) = 6

action(6, #) = **e4**

action( 7, #) = r<sub>1</sub>



\$2:  $i i \# \xrightarrow{\quad} \$: i+i \# \xrightarrow{\quad} \text{action}(4, i) = S_3$

$\uparrow$   
**F**

<b>3</b>	$i$
<b>4</b>	+
<b>1</b>	$E$
<b>0</b>	#

**P**

$\text{action}(1, i) = e3$



# end

结束即是新的开始  
THE END IS THE BEGINNING