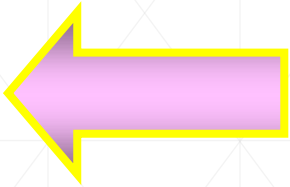
5.7 LALR (1)分析

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： LR(1) ⊃ SLR(1) ⊃ LR(0)

存储开销： LR(1) ↑ SLR(1) LR(0) ↓

存储开销 ⇨ LR(0)、SLR(1)

分析功能 ⇨ LR(1)

⇨ LALR(1)

LALR(1)

Look −Ahead LR(1)

LALR(1)分析表状态数 = LR(0)的状态数

SLR(1)⊆ LALR(1)分析功能⊆ LR(1)

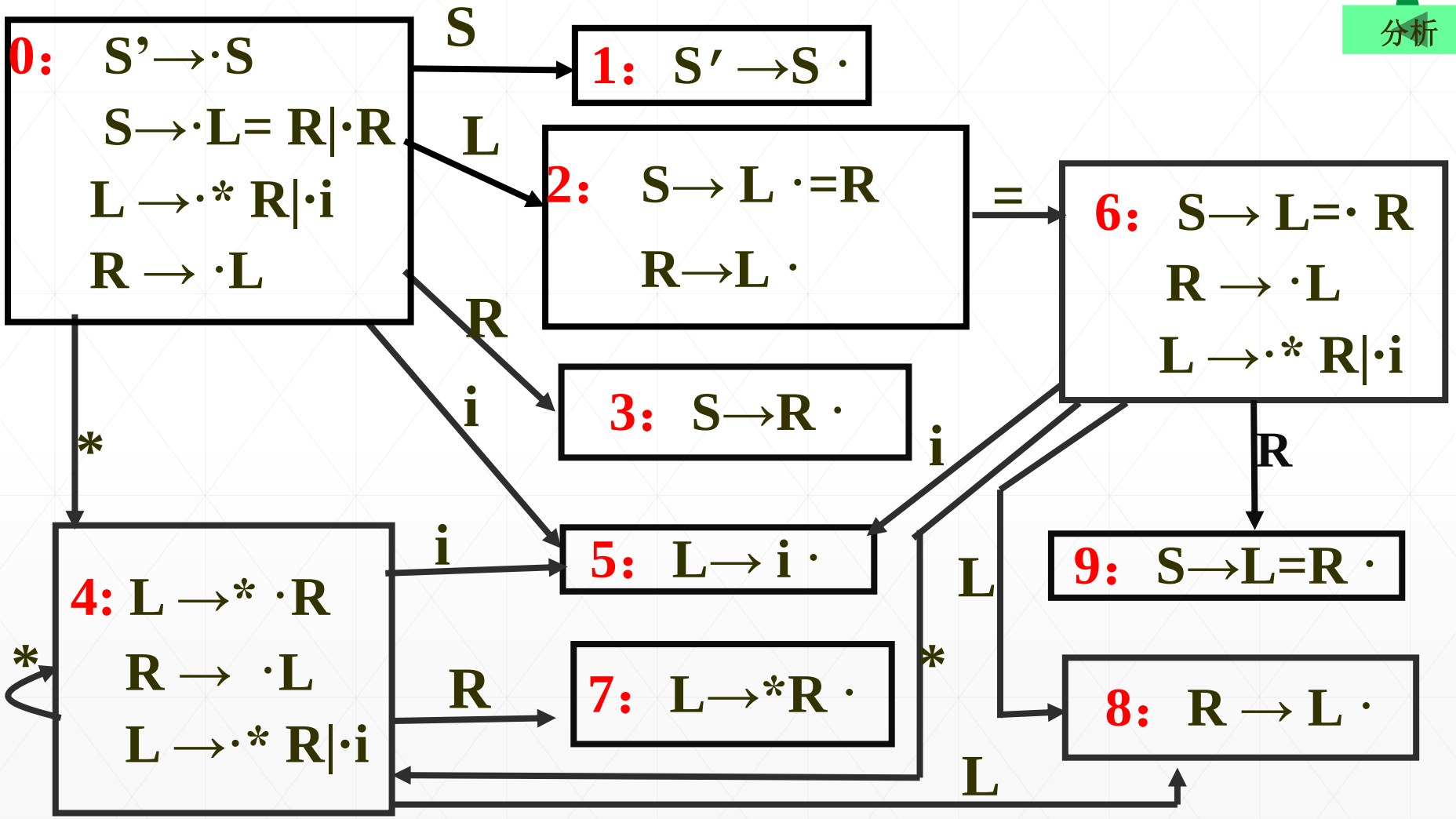
例： 设有文法G(*S*)为

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

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

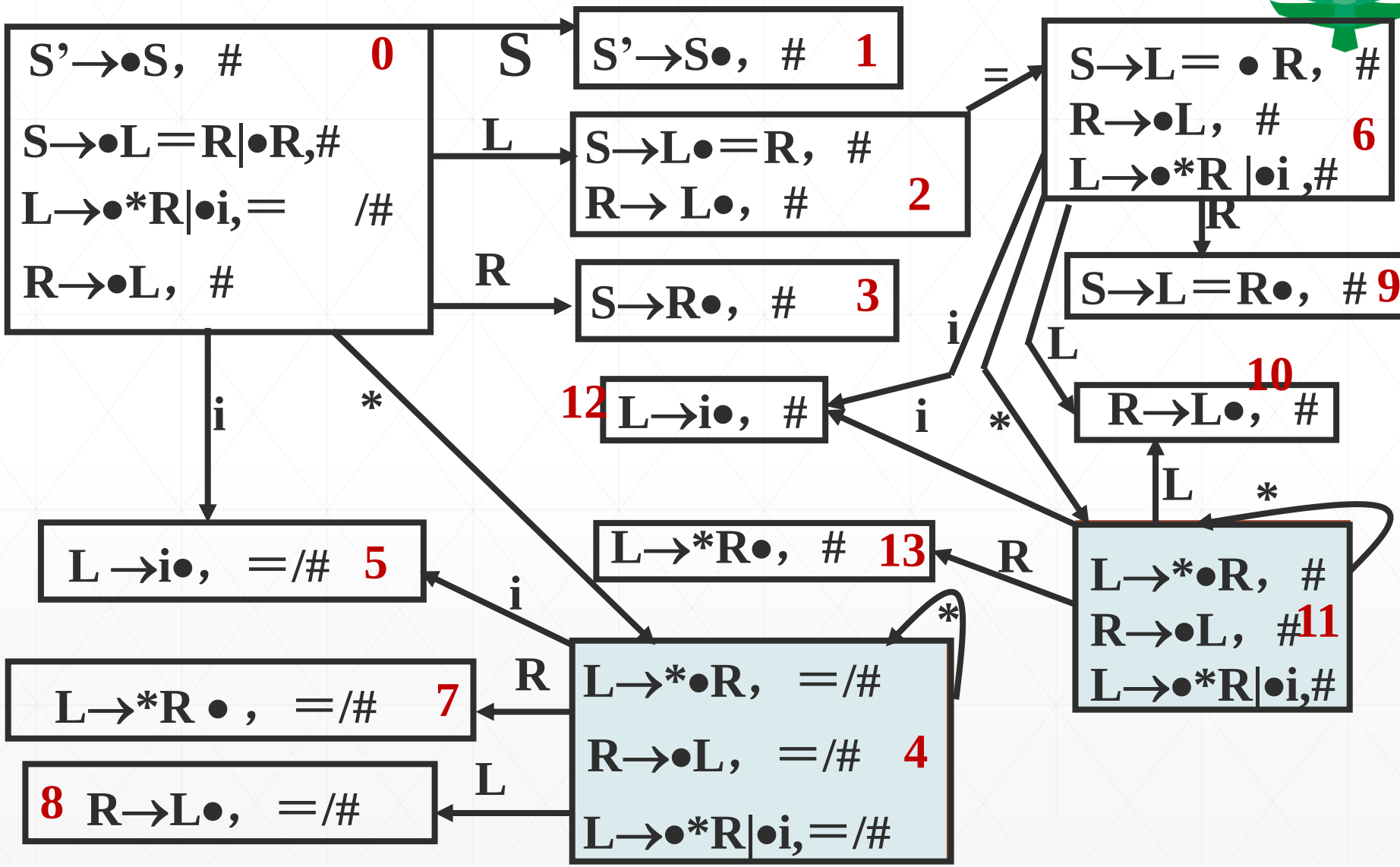
$$R \rightarrow L$$

G(S)的识别LR(0)项目有效可归前缀的DFA：

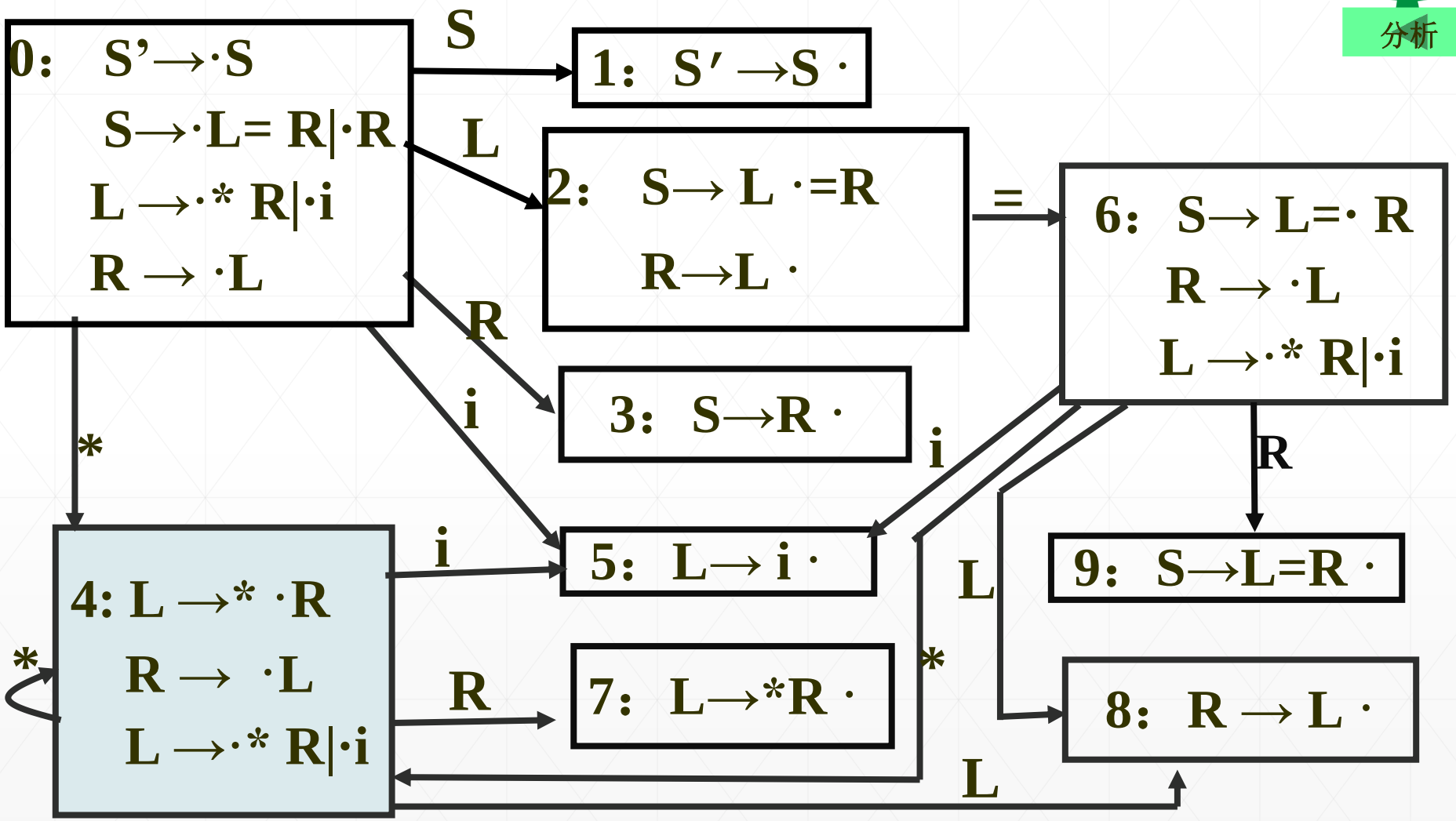


*S'→S S→ L=R ①| R ② L →*R ③| i④ R → L⑤*

G(S)的识别LR(1)项目有效可归前缀的DFA:

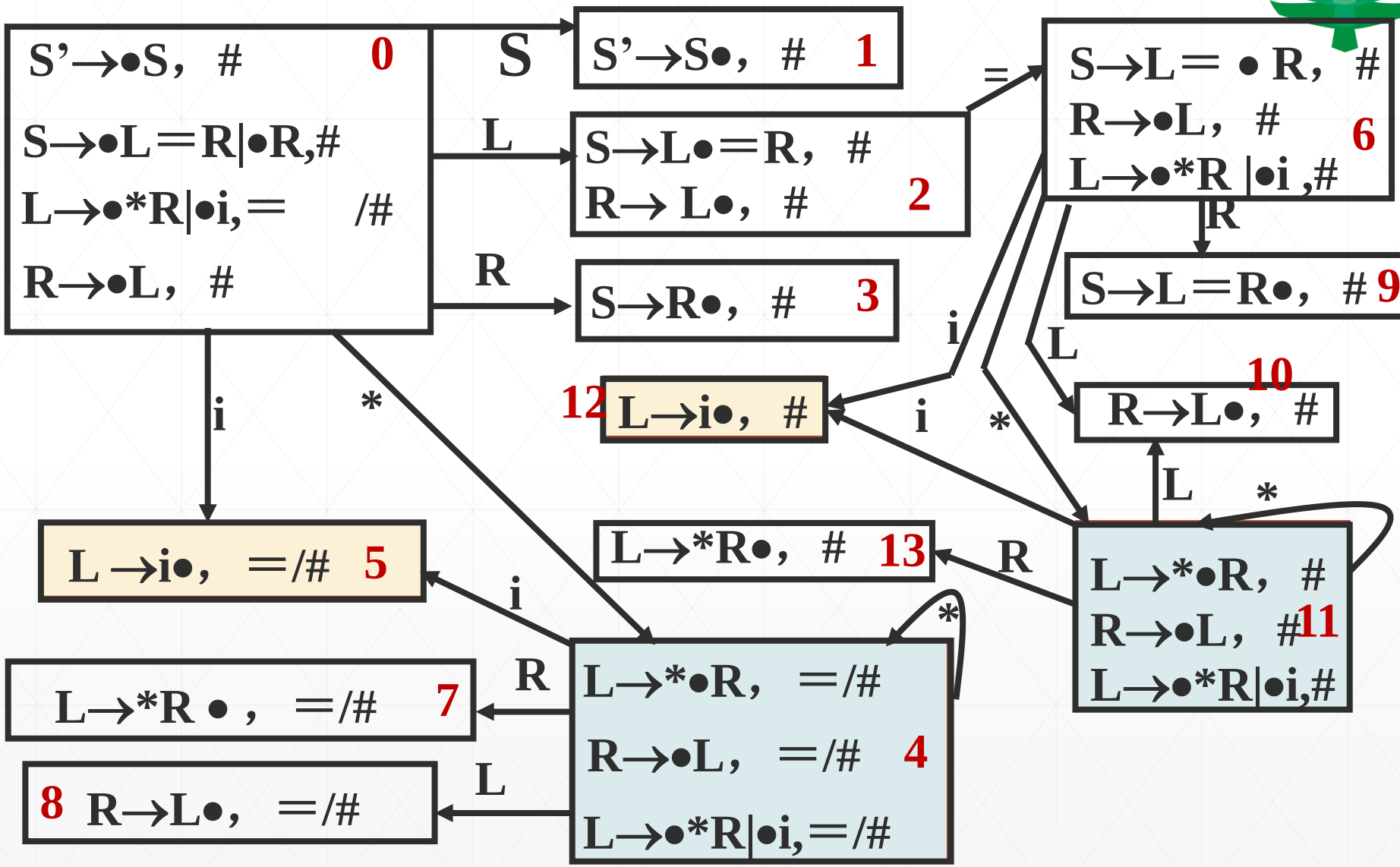


G(S)的识别LR(0)项目有效可归前缀的DFA：

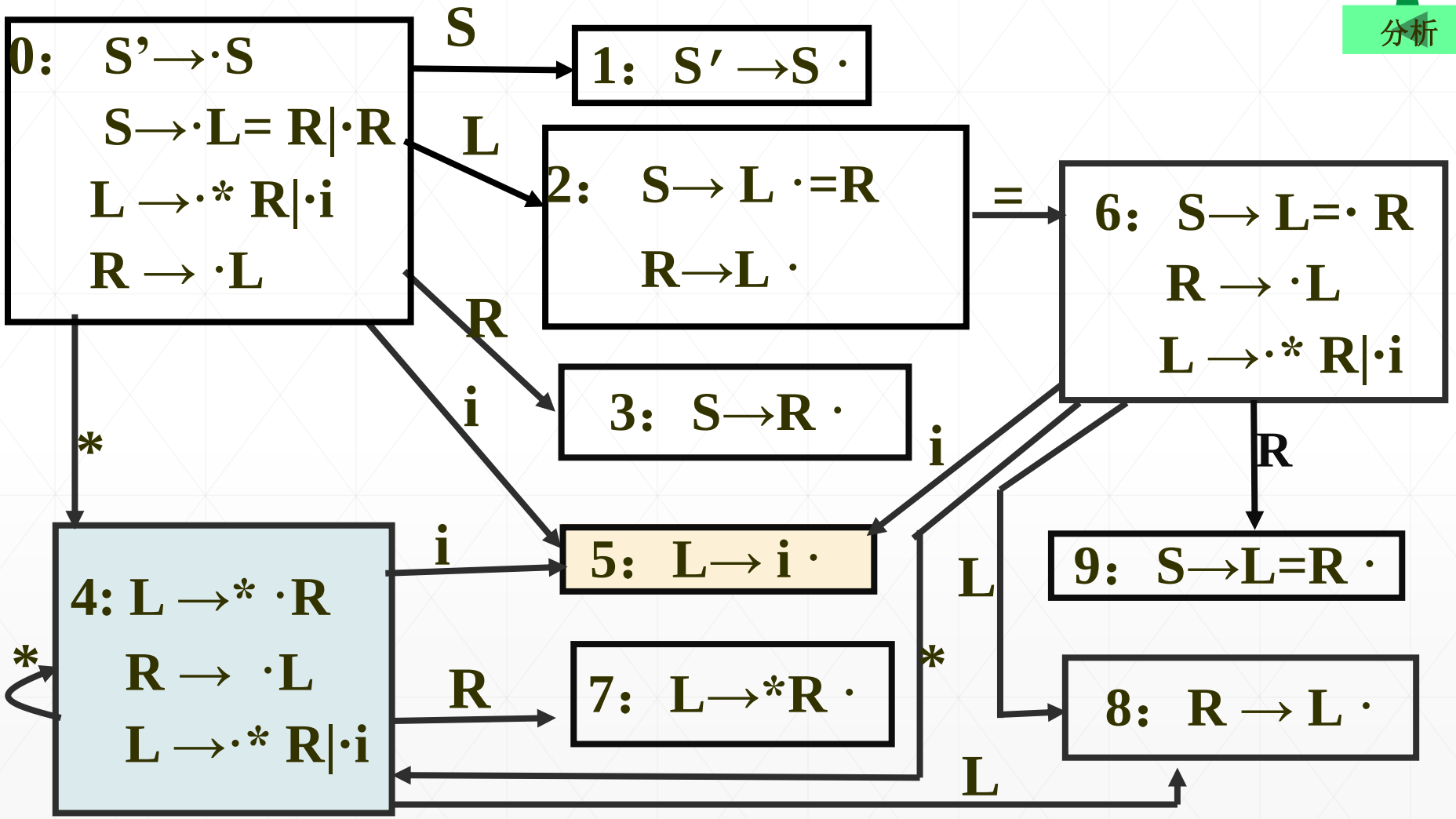


*S'→S S→ L=R ①| R ② L →*R ③| i④ R → L⑤*

G(S)的识别LR(1)项目有效可归前缀的DFA:

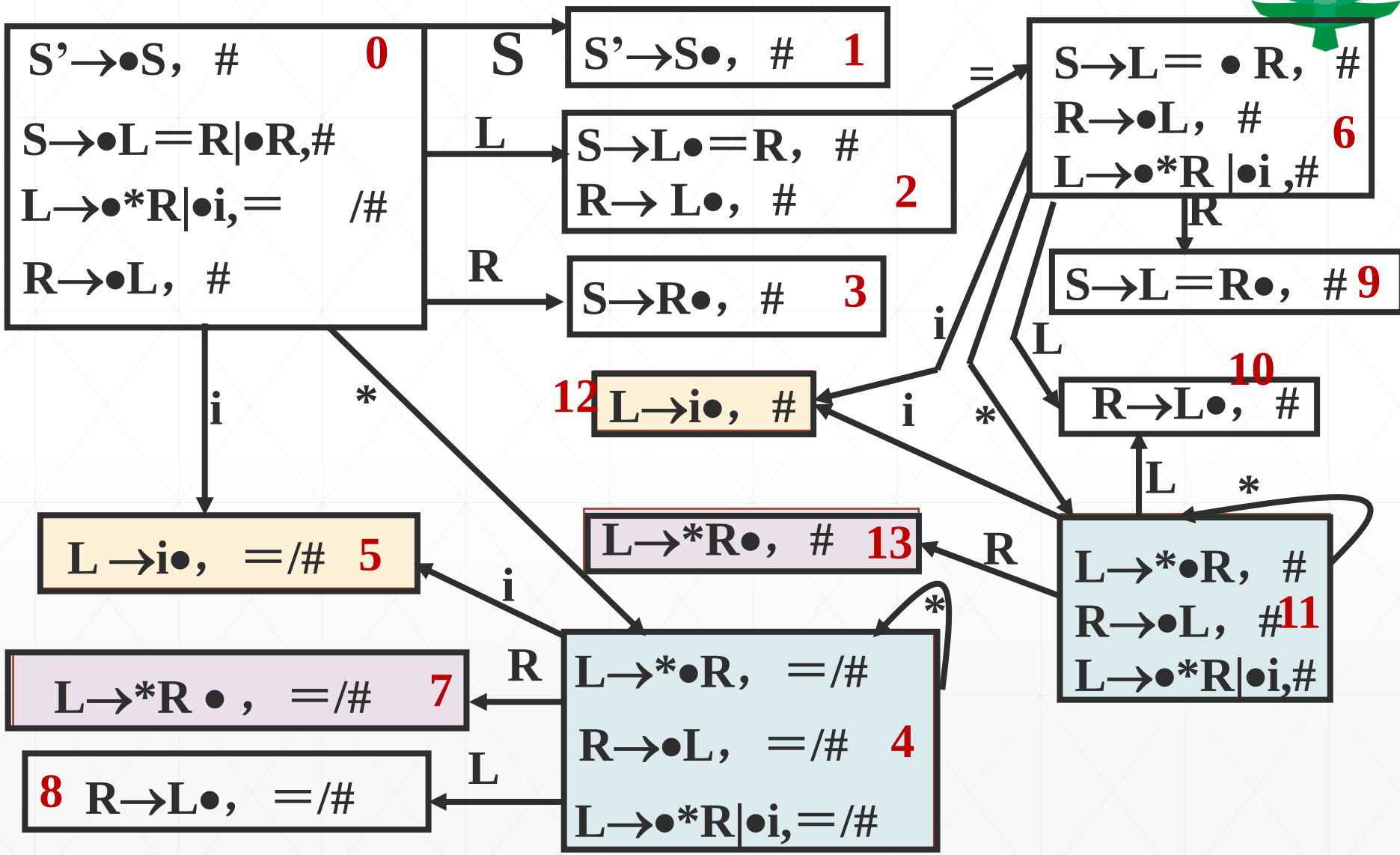


G(S)的识别LR(0)项目有效可归前缀的DFA：

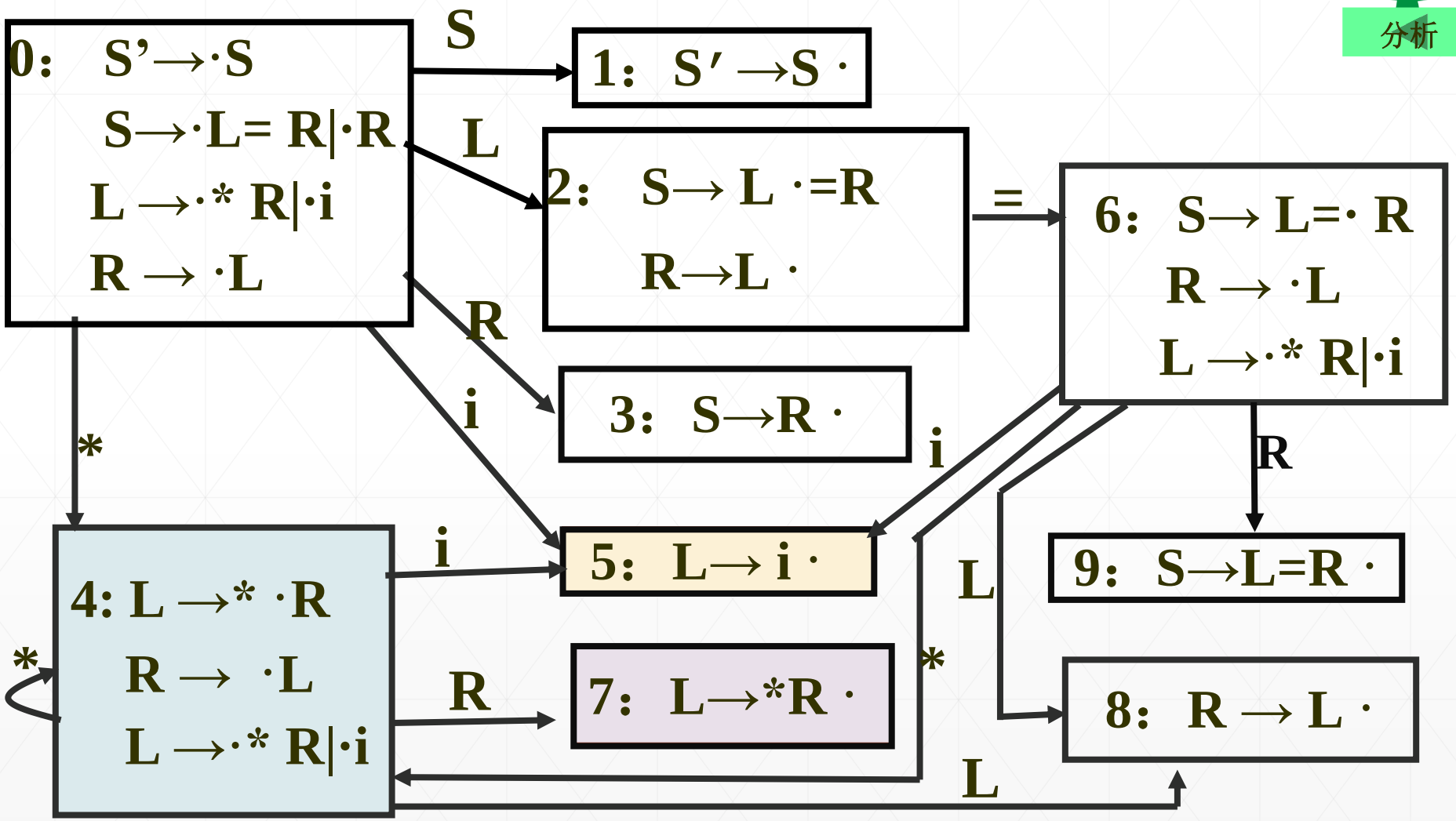


*S'→S S→ L=R ①| R ② L →*R ③| i④ R → L⑤*

G(S)的识别LR(1)项目有效可归前缀的DFA:

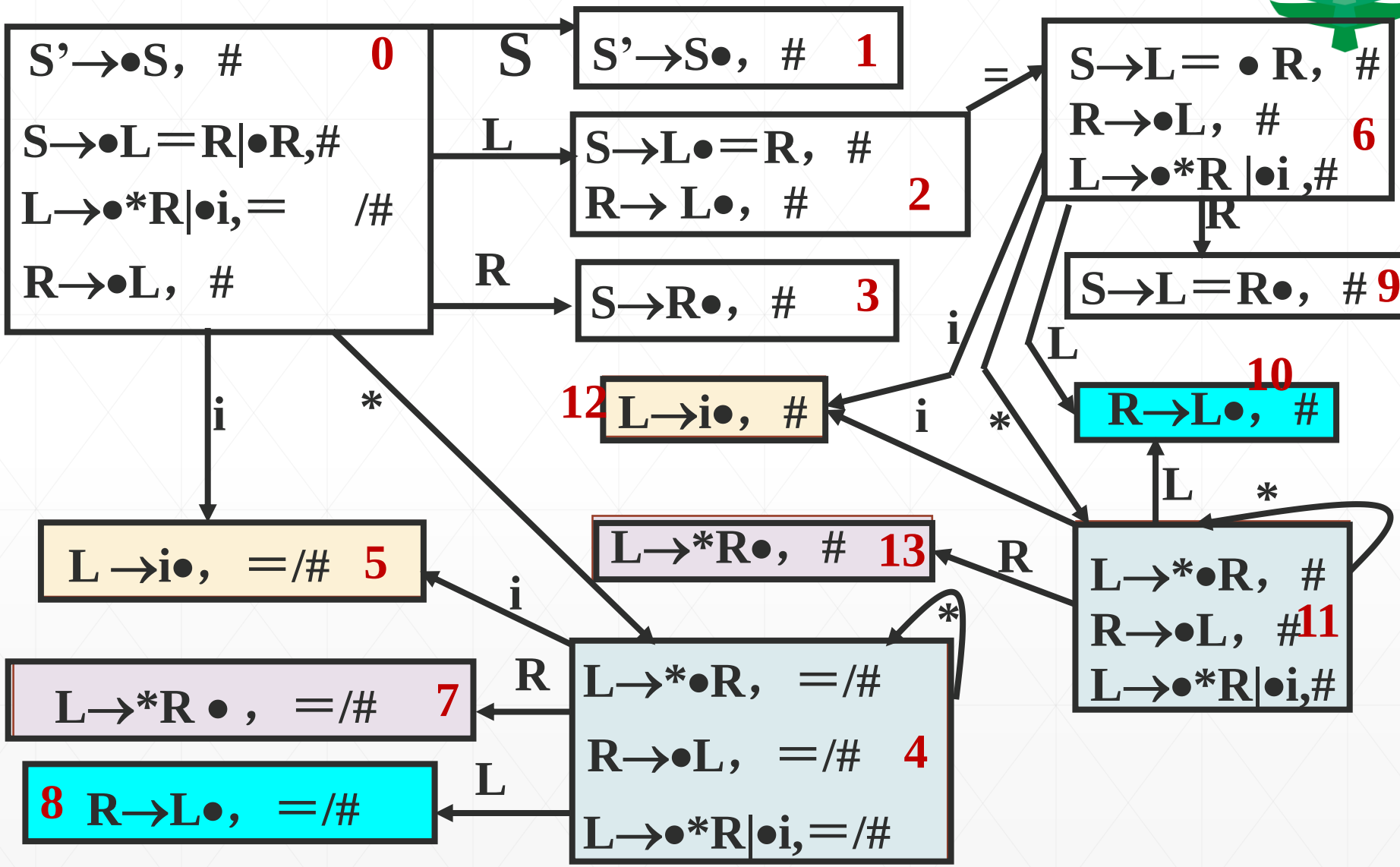


G(S)的识别LR(0)项目有效可归前缀的DFA：

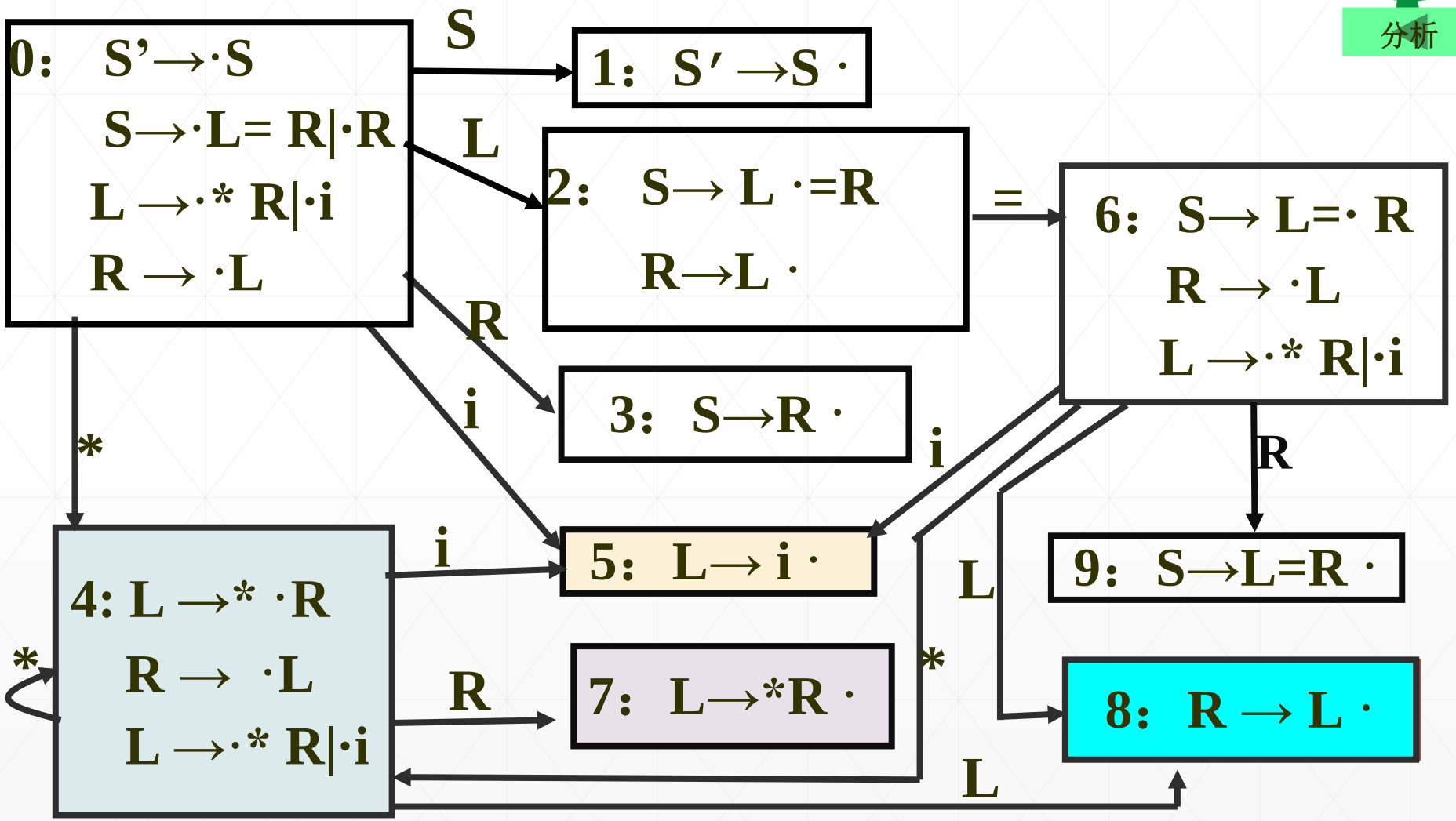


*S'→S S→ L=R ①| R ② L →*R ③| i④ R → L⑤*

G(S)的识别LR(1)项目有效可归前缀的DFA:



G(S)的识别LR(0)项目有效可归前缀的DFA：



*S'→S S→ L=R ①| R ② L →*R ③| i④ R → L⑤*

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		LR(1)的*DFA*状态
4	分裂为	4、11
5	分裂为	5、12
7	分裂为	7、13
8	分裂为	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）

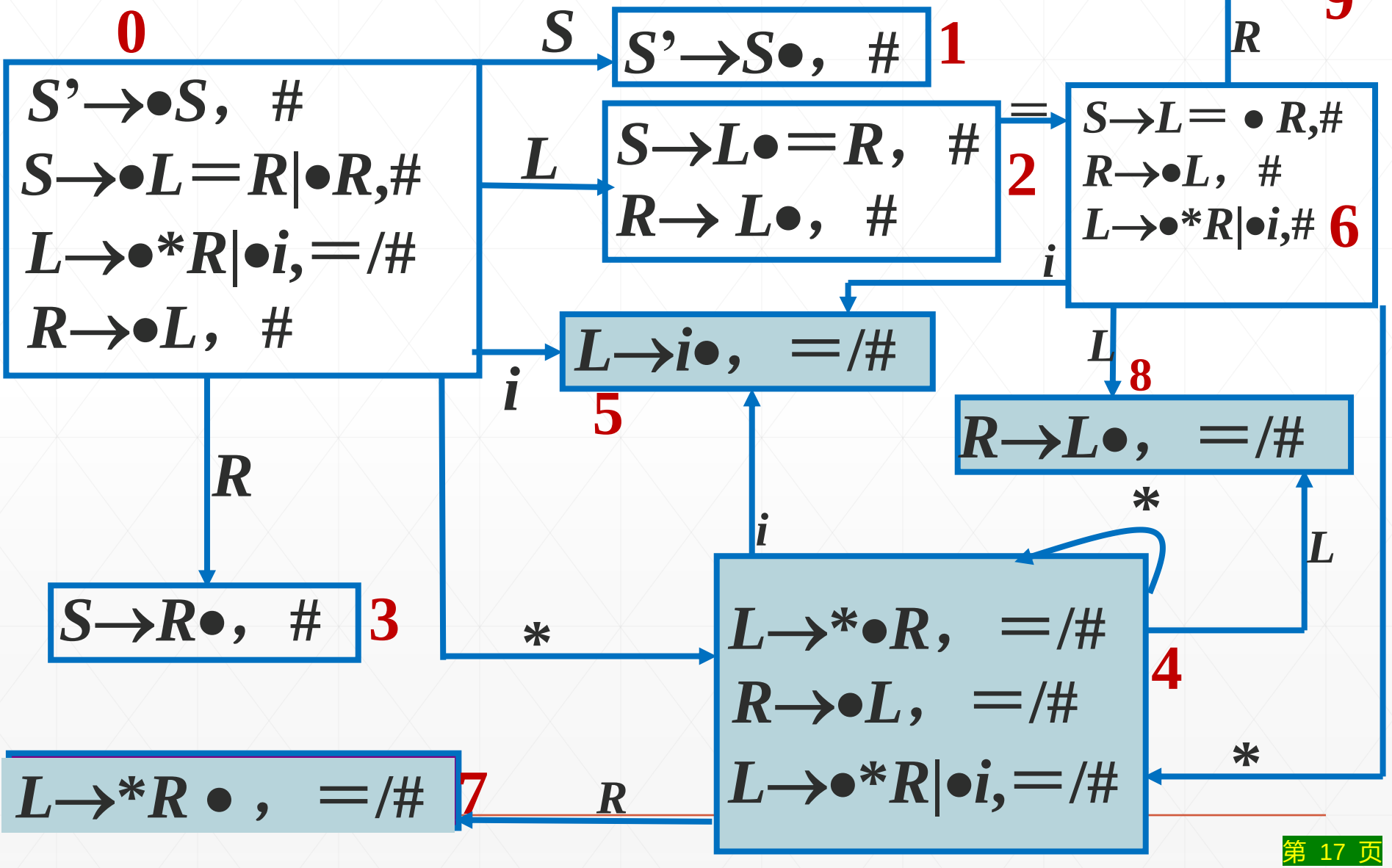
对文法*G*的LR(1)项目集规范族(识别可归前缀的DFA)，若存在两个(或两个以上)项目集(状态)I_0、I_1，其中的LR(0)项目相同,仅搜索符不同,则称I_0 、I_1为*G*的LR(1)的同心项目集(同心状态).或称I_0 、I_1具有相同的心.

合并LR(1)的同心项目集(状态)=> LALR (1)的*C*或*DFA*

个数减少 (DFA的状态、项目集)

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



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

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;

③ 若接受项目［$S'\rightarrow S \cdot$， ＃］$\in I_k$，则置Action $(K,$ ＃$)=$acc;

④ Action表中不能用①至③规则填入信息的, 则置“出错标志”

- 算法（ LALR(1)分析表构造）

输入：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，...， n｝＝分析表的状态

① 若$f(k, a)=j$，$a \in V_T$， 则置 Action $(k, a)=S_j$;

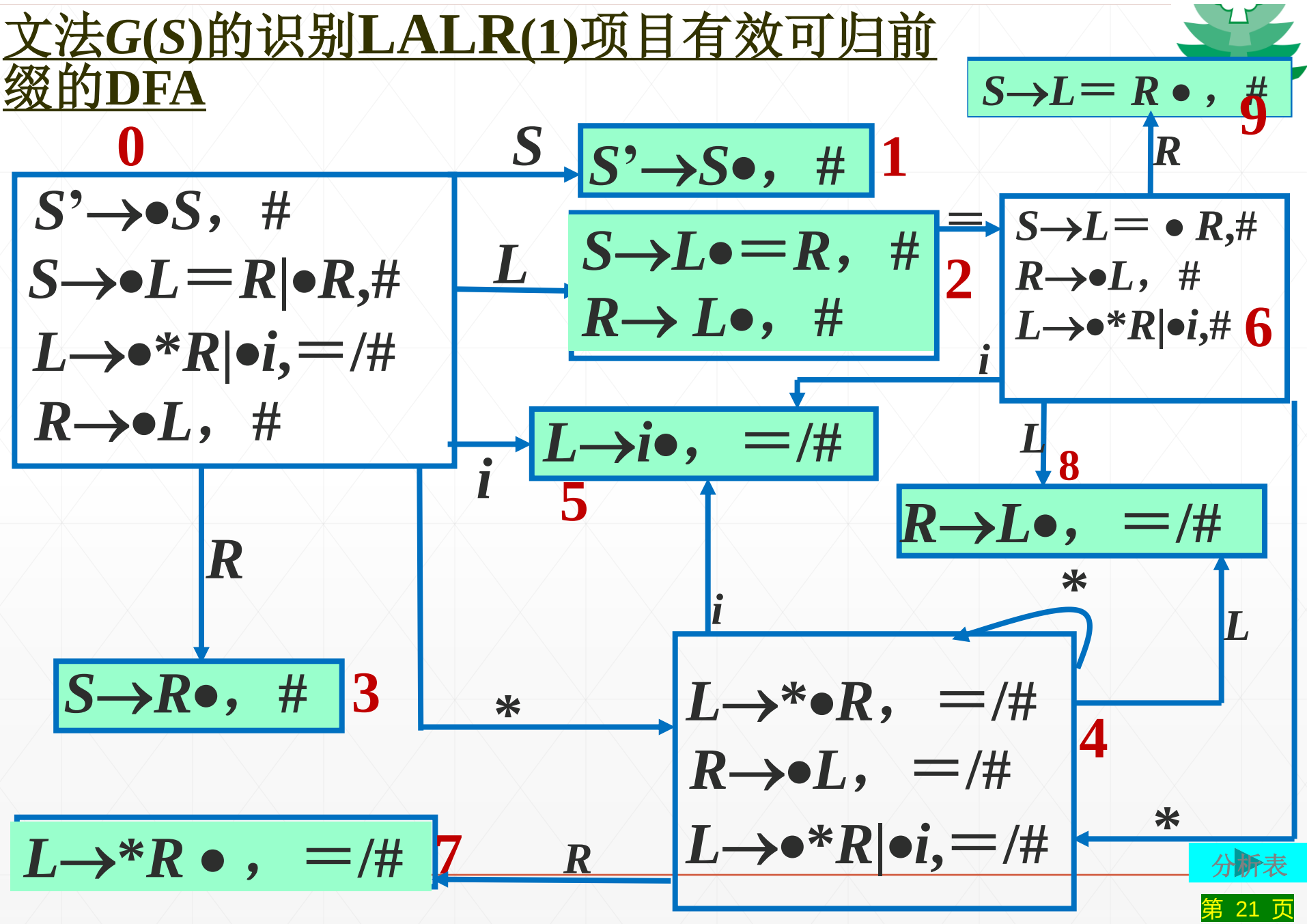
若$f(k, A)=j$，$A \in V_N$，则置Goto$[K, A]=j$。

②若［$A \rightarrow \alpha \cdot, a$］$\in k$，则置Action$(K, a)=r_j$，其中$A \rightarrow \alpha$为文法$G$的第$j$个产生式;

③ 若接受项目［$S' \rightarrow S \cdot$， ＃］$\in k$，则置Action $(K,$ ＃$)=$acc;

④ Action表中不能用①至③规则填入信息的，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*	#	*L*	*R*	*S*
0		S_4	S_5		2	3	1
1				acc			
2	S_6			r_5			
3				r_2			
4		S_4	S_5		8	7	
5	r_4			r_4			
6		S_4	S_5		8	9	
7	r_3			r_3			
8	r_5			r_5			
9				r_1			

S’→*S*

S→*L*=*R* ①|*R*②

L →**R* ③| *i*④

R → *L*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。

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，分析表构造同LR(1)。

确定串中错误前有时会比LR(1)多产生若干步归约。

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

I_k: $[A \to \alpha\bullet, W_1]$　　I_j: $[A \to \alpha\bullet, W_2]$

$[B \to \beta\bullet a\,\gamma, b]$　　$[B \to \beta\bullet a\,\gamma, c]$

由于I_k与I_j均无冲突，故有

$W_1 \cap \{a\} = \varnothing$　　$W_2 \cap \{a\} = \varnothing$

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

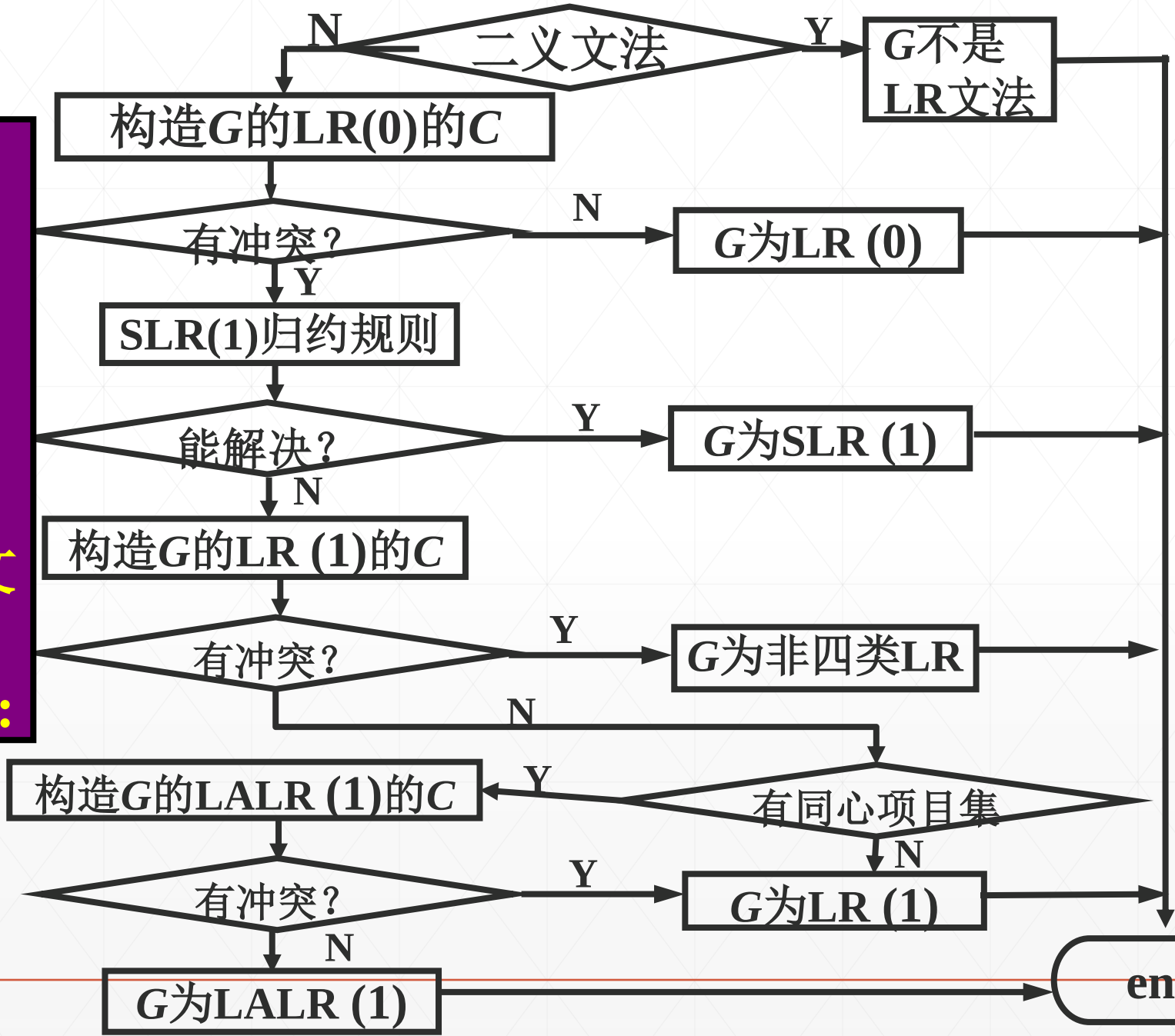
I_k与I_j合并，有$I_{k/j}$：$[A \to \alpha\bullet, W_1 \cup W_2]$

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

若此时 $I_{k/j}$有“移进 — 归约 ”冲突，则

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

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

$S' \rightarrow \bullet S$

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

$A \rightarrow \bullet d$

$B \rightarrow \bullet d$

d

1

$A \rightarrow d \bullet$

$B \rightarrow d \bullet$

归约—归约冲突

G非LR(0)文法

用SLR(1)归约规则

FOLLOW(A) = {a,c}

FOLLOW(B) = {a,c}

$\cap \neq \varnothing$

G非SLR(1)文法

$S' \rightarrow S$

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

$A \rightarrow d$

$B \rightarrow d$

$S' \to S$
$S \to Aa \mid bAc \mid Bc \mid bBa$
$A \to d$
$B \to d$

0
$S' \to \bullet S, \#$
$S \to \bullet Aa \mid \bullet bAc \mid$
$\bullet Bc \mid \bullet bBa, \#$
$A \to \bullet d, a$
$B \to \bullet d, c$

1 $S' \to S\bullet, \#$

2 $S \to A\bullet a, \#$

3
$S \to b \bullet Ac \mid$
$b \bullet Ba, \#$
$A \to \bullet d, c$
$B \to \bullet d, a$

4 $S \to B\bullet c, \#$

5
$A \to d \bullet, a$
$B \to d \bullet, c$

6 $S \to Aa\bullet, \#$

7 $S \to bA\bullet c, \#$

8 $S \to bB\bullet a, \#$

9
$A \to d \bullet, c$
$B \to d \bullet, a$

10 $S \to Bc\bullet, \#$

11 $S \to bAc\bullet, \#$

12 $S \to bBa\bullet, \#$

S A B d b a c A B d c a

同心项目集

G为LR(1)

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)

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：$E \to E+E \mid E*E \mid (E) \mid i$

G'：　$E \to T \mid E+T$

$T \to F \mid T*F$

$F \to (E) \mid i$

例如，　G：$S \to iSeS \mid iS \mid a$

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

G'：$S \to A \mid B$

$A \to iAeA \mid a$

$B \to iS \mid iAeB$

例：设有文法$G(E)$：

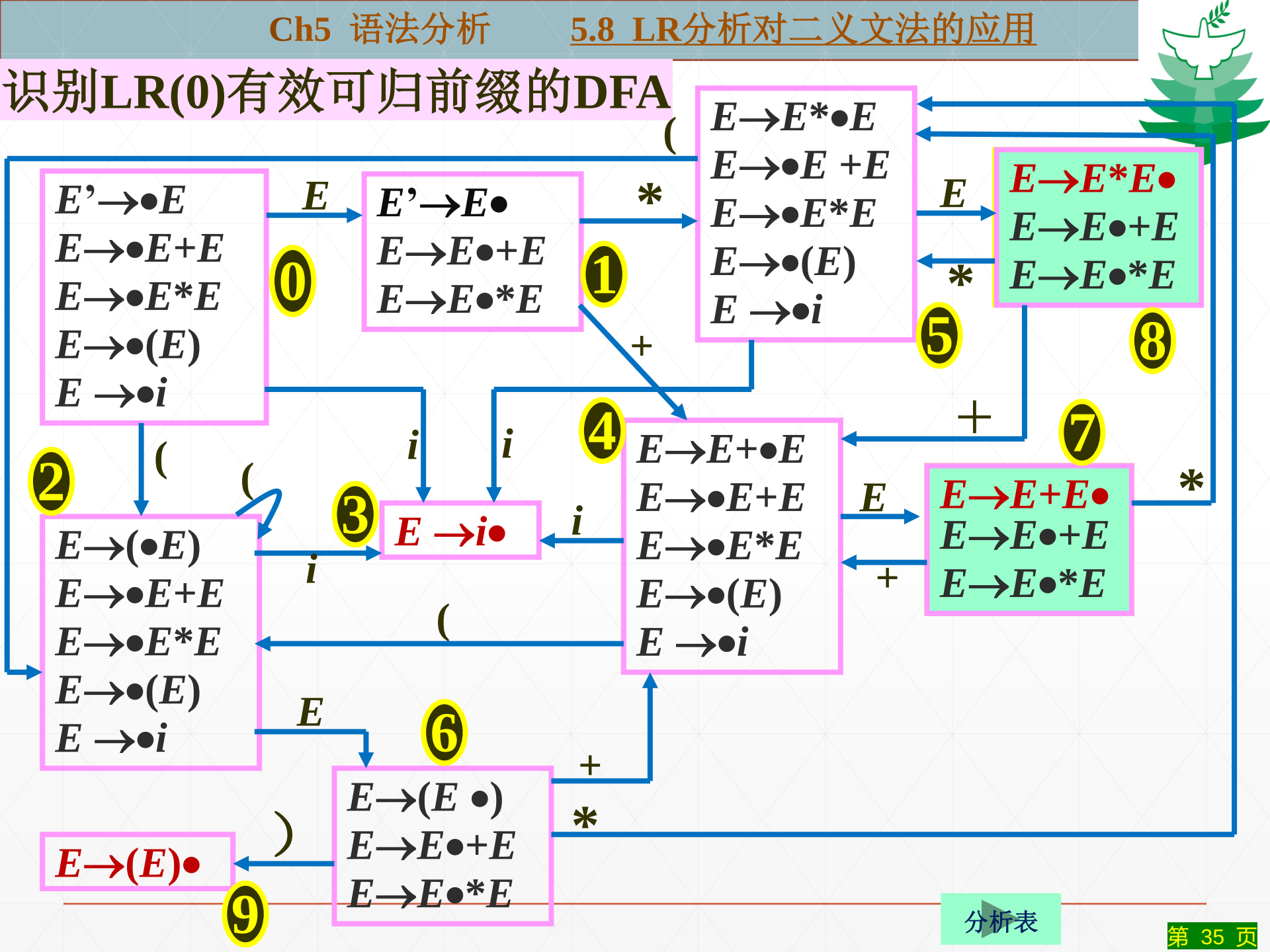
$E \rightarrow E+E|E*E|(E)|i$

拓广文法：

$E' \rightarrow E$

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

识别LR(0)有效可归前缀的DFA



⑦

$E \to E+E\bullet$

$E \to E\bullet+E$

$E \to E\bullet*E$

⑧

$E \to E*E\bullet$

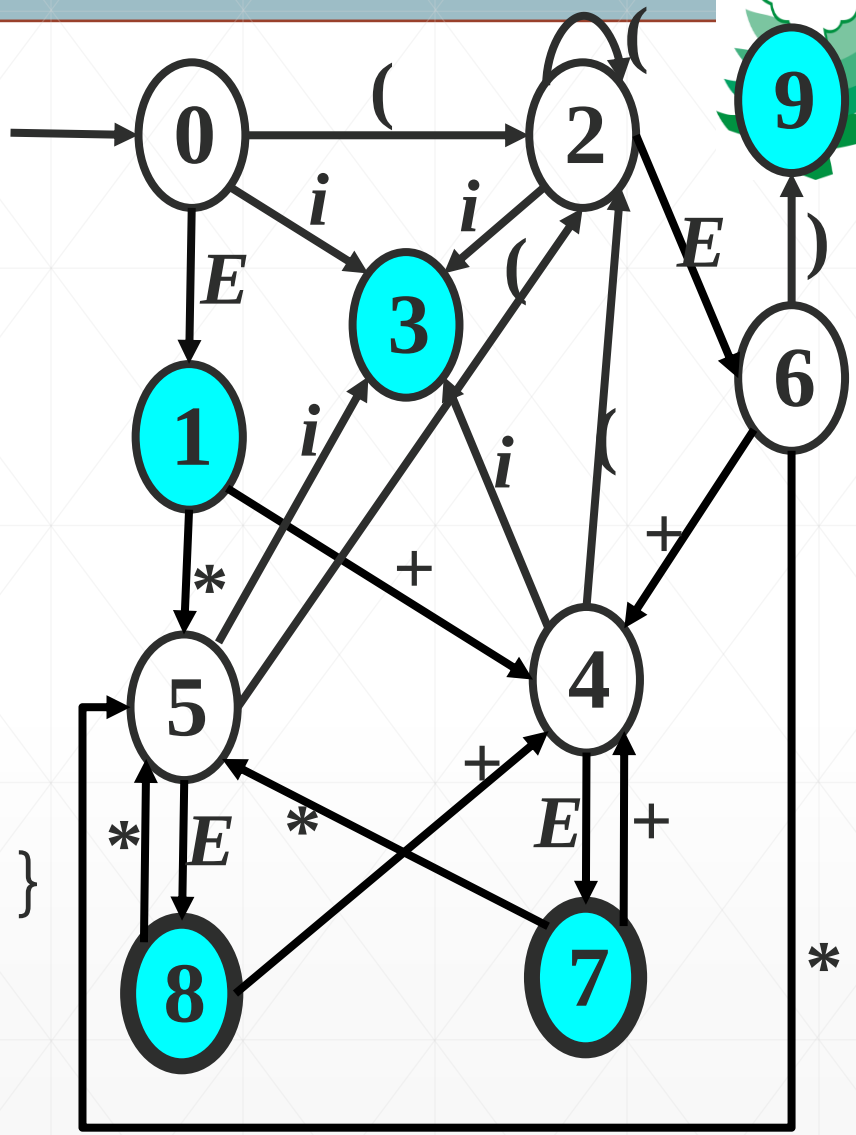
$E \to E\bullet+E$

$E \to E\bullet*E$

FOLLOW(E)＝

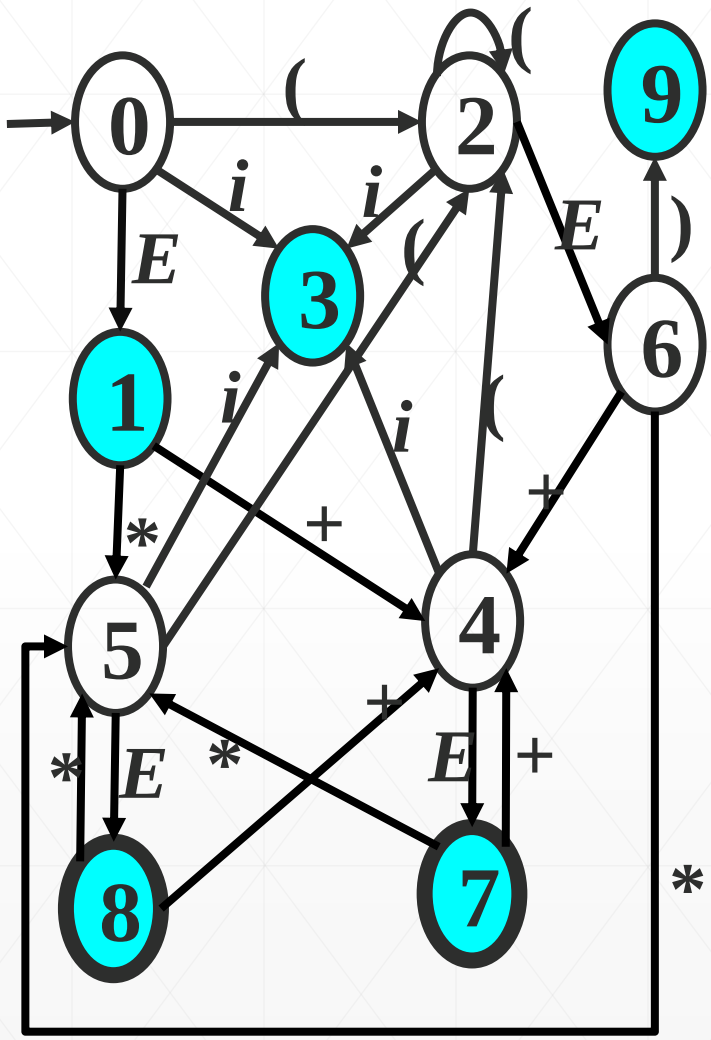
{＋, ＊,), # } ∩ {{＋} ∪ {*}}

≠ Φ



$E' \to E$　　$E \to E+E$ ① $|E*E$ ②$|(E)$ ③$|i$ ④

$E' \to E$　　$E \to E+E$ ① $|E*E$ ②$|(E)$ ③$|i$ ④



文法G(E)的LR分析表

状态	STATION						GOTO
	+	*	(	)	i	#	E
0			s_2		s_3		1
1	s_4	s_5				acc	
2			s_2		s_3		6
3	r_4	r_4	r_4	r_4	r_4	r_4	
4			s_2		s_3		7
5			s_2		s_3		8
6	s_4	s_5		s_9			
7	r_1	s_5	r_1	r_1	r_1	r_1	
8	r_2	r_2	r_2	r_2	r_2	r_2	
9	r_3	r_3	r_3	r_3	r_3	r_3	

1:$E' \to E$•　3:$E \to i$•　7: $E \to E+E$•　8: $E \to E*E$•　9:$E \to (E)$•

$: *i*+*i*+*i* #

F

action(7 , +) = r_2

7	*E*
4	+
1	*E*
0	#

P

E

$: *i*+*i* * *i* #

F

action(7 , *) = S_5

5	*
7	*E*
4	+
1	*E*
0	#

P

例：设有文法G:

$S \rightarrow iSeS$ ①| iS ②| a ③

其中：

i : if e_r then

e: else

a, S: 语句

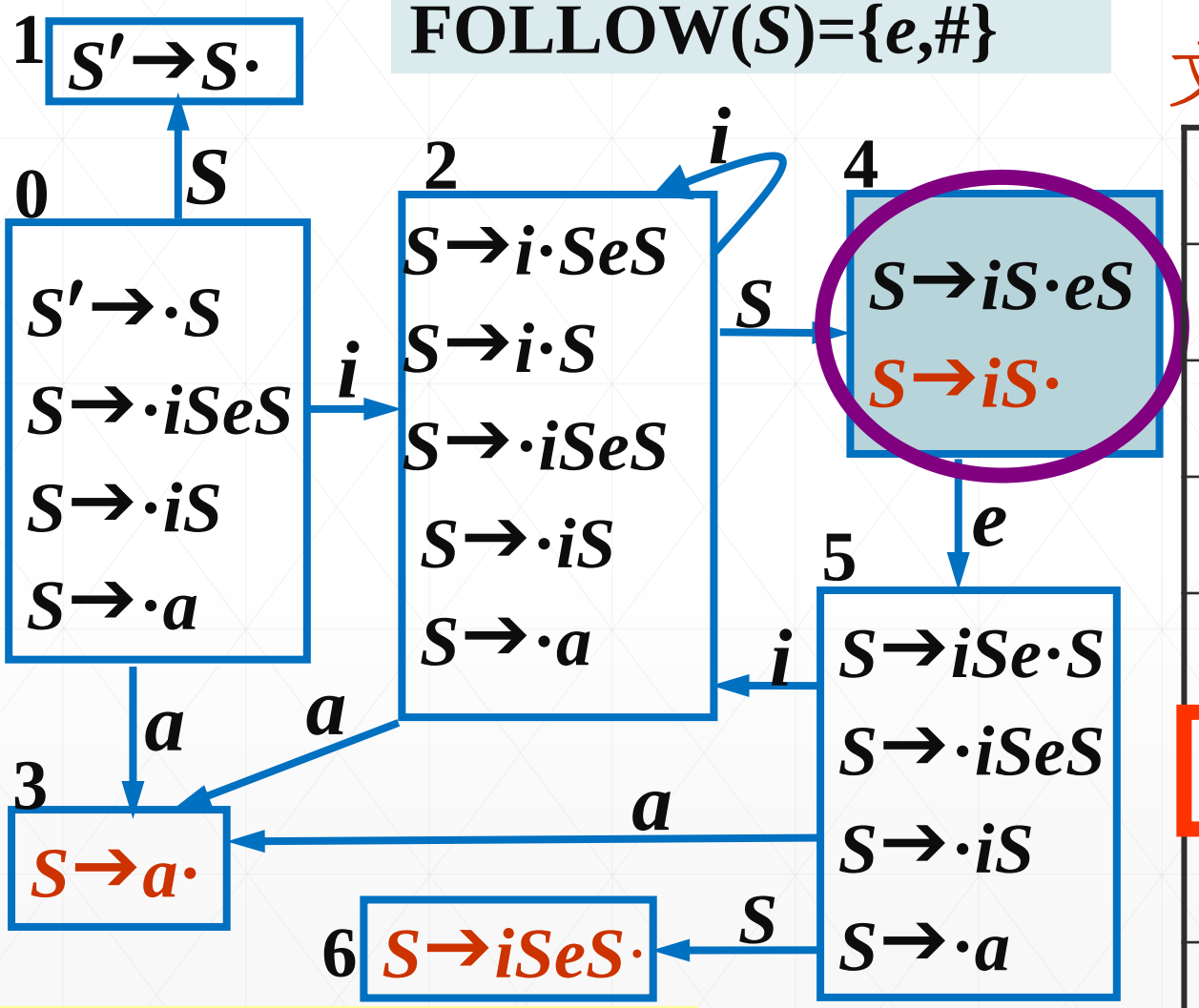
拓广文法G:

$S' \rightarrow S$

$S \rightarrow iSeS$ ①| iS ②| a ③

文法G的识别LR(0)项目有效可归前缀的DFA

FOLLOW(S)={e,#}



文法G的LR分析表

	i	e	a	#	S
0	S_2		S_3		1
1				acc	
2	S_2		S_3		4
3	r_3	r_3	r_3	r_3	
4	r_2	S_5	r_2	r_2	
5	S_2		S_3		6
6	r_1	r_1	r_1	r_1	

S’→ S

S→ iSeS ①| iS ②| a ③

$S' \rightarrow S$

$S \rightarrow iSeS$ ①| iS ②| a③

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

action(4 , *e*) = S_5

5	*e*
4	*S*
2	*i*
2	*i*
0	#

F++;P++

P

- 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* ;　⇨　(*i*+*i*) ;

*i i***i*;　⇨　*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。

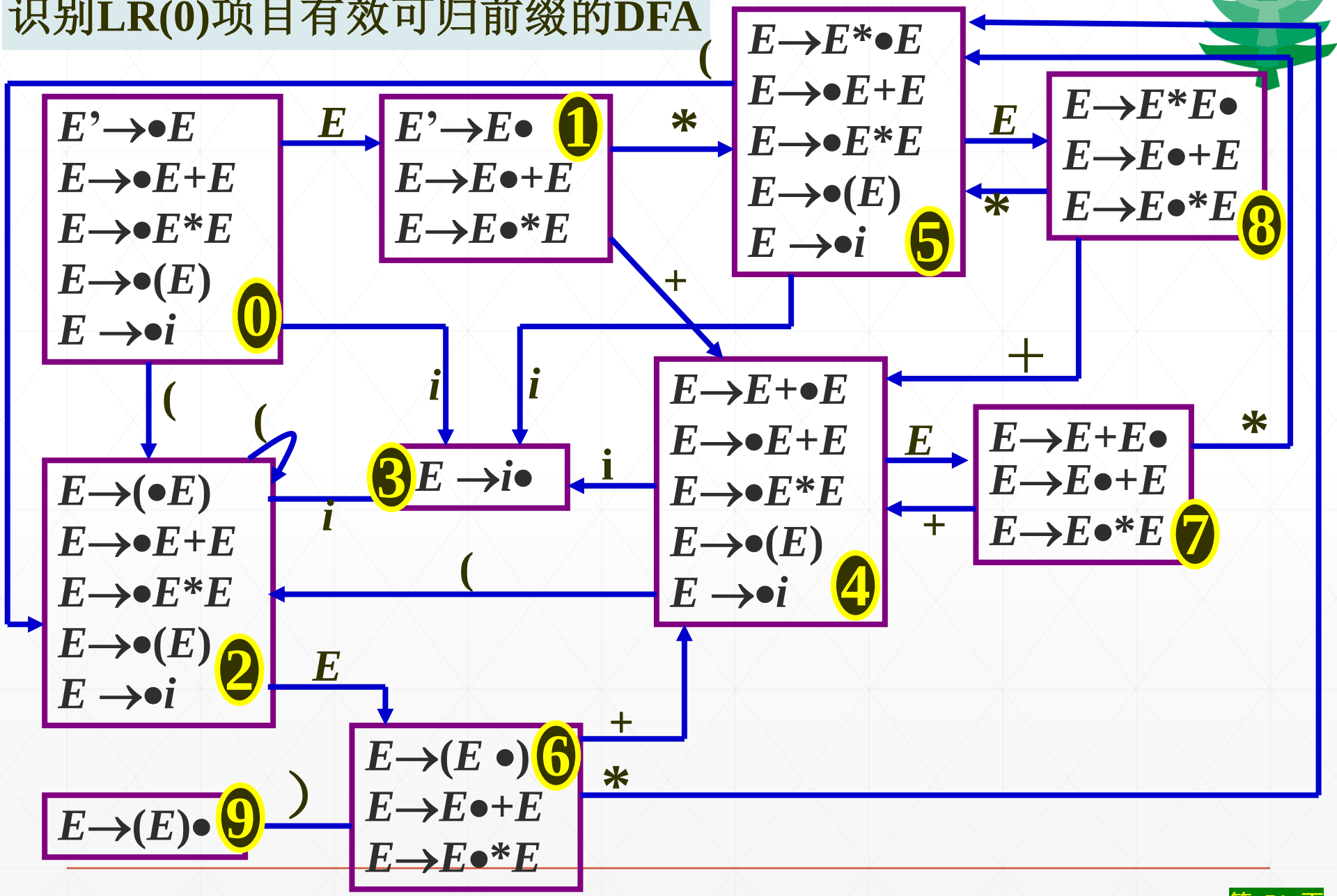
例:设有文法$G(E)$:

$E' \rightarrow E$

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

构造文法$G(E)$的带出错处理的LR分析表

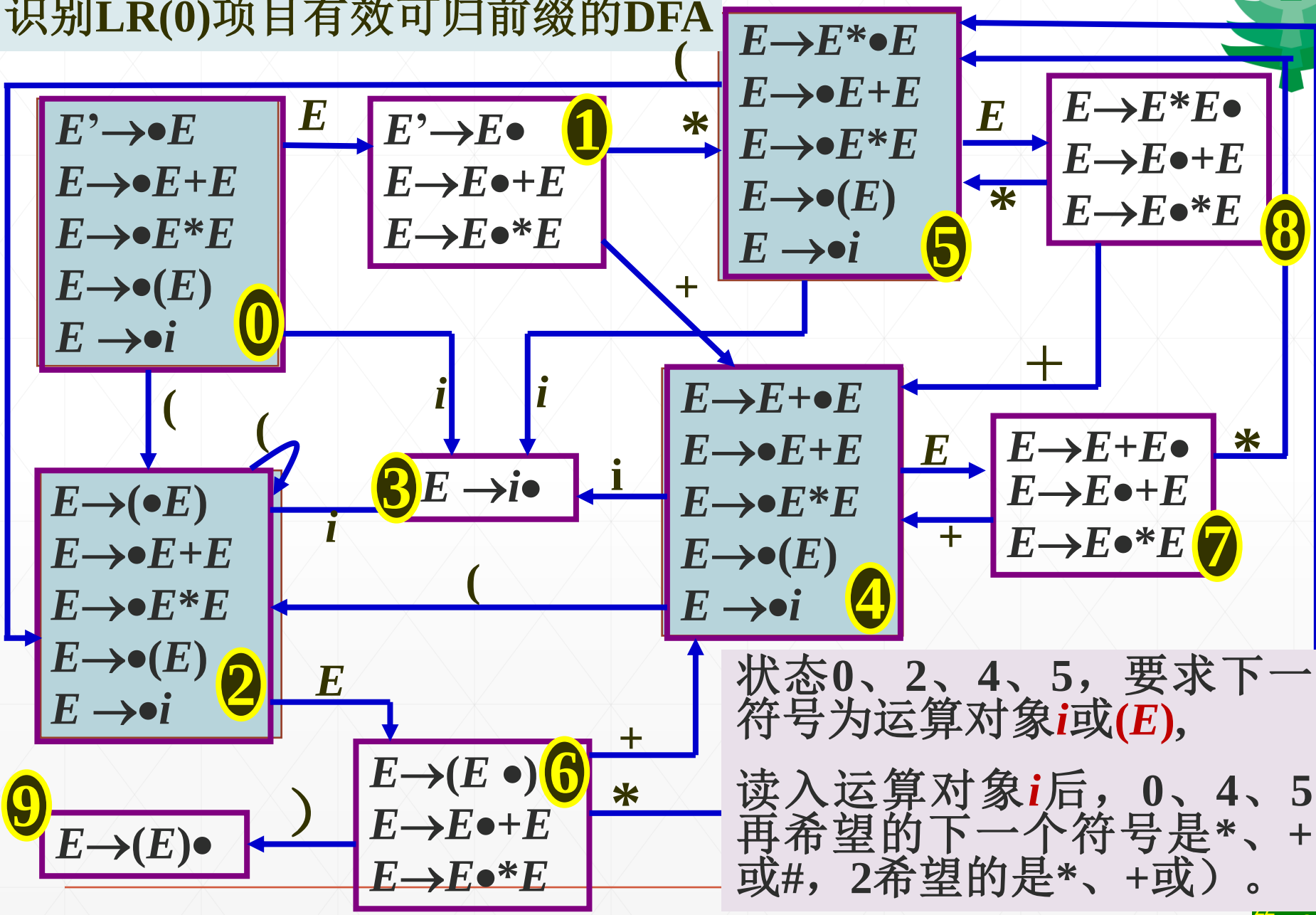
识别LR(0)项目有效可归前缀的DFA



文法G(E)的LR分析表

状态	ACTION						GOTO
	i	+	*	(	)	#	*E*
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



状态0、2、4、5，要求下一符号为运算对象*i*或(*E*)，

读入运算对象*i*后，0、4、5再希望的下一个符号是*、+或#，2希望的是*、+或）。

e1错误信息：

缺少运算对象。

e1错误位置：

处于状态0、4、5时，若遇到+、*、#；

处于状态2：若遇到+、*、），

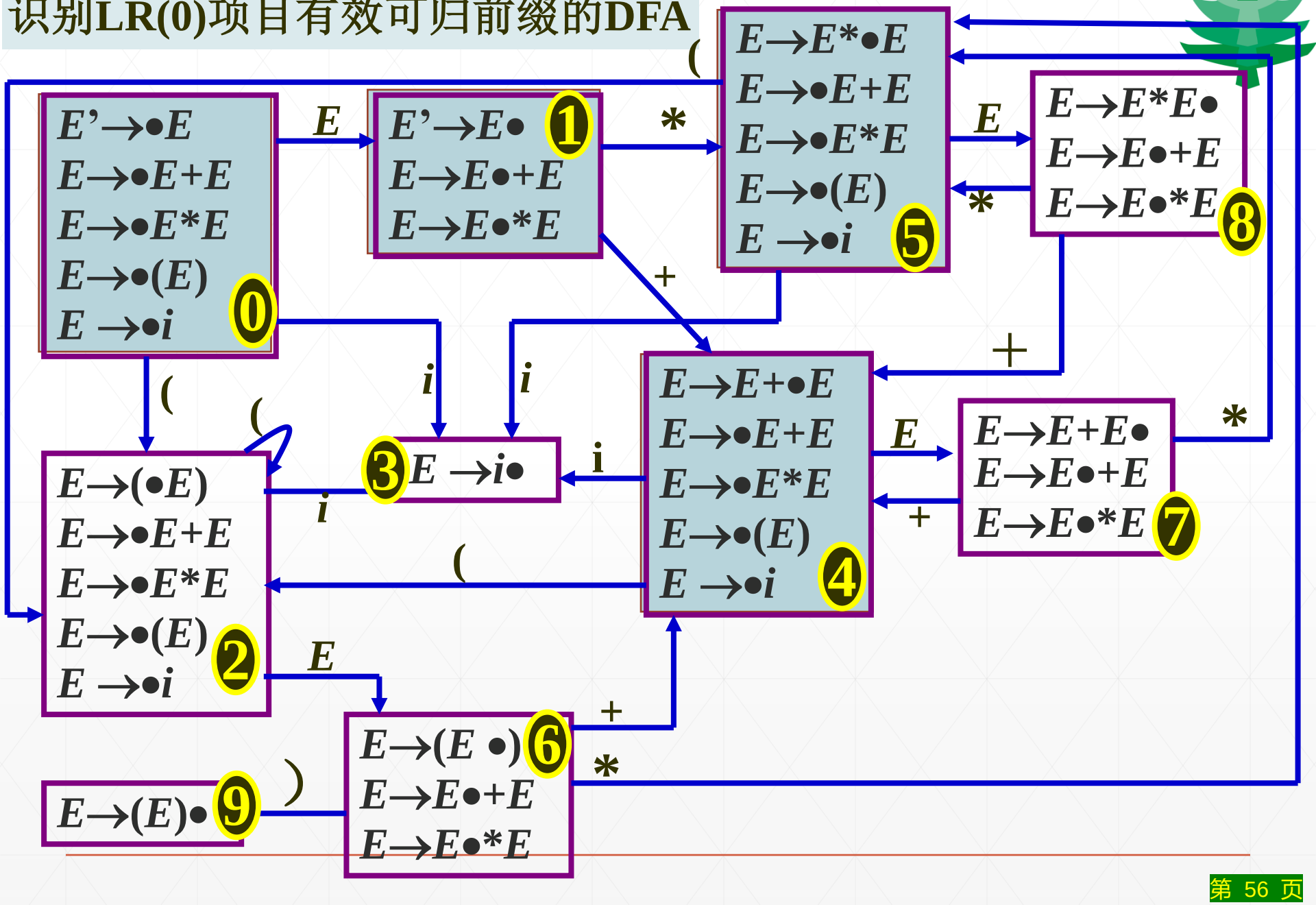
e1错误功能：

将假设的 *i* 和状态3入栈；(shift)

文法G(E)的LR分析表

状态	ACTION						GOTO
	i	+	*	(	)	#	*E*
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错误信息:

“ ）”不配对。

e2错误位置:

处于状态0、1、4、5时，若遇到）；

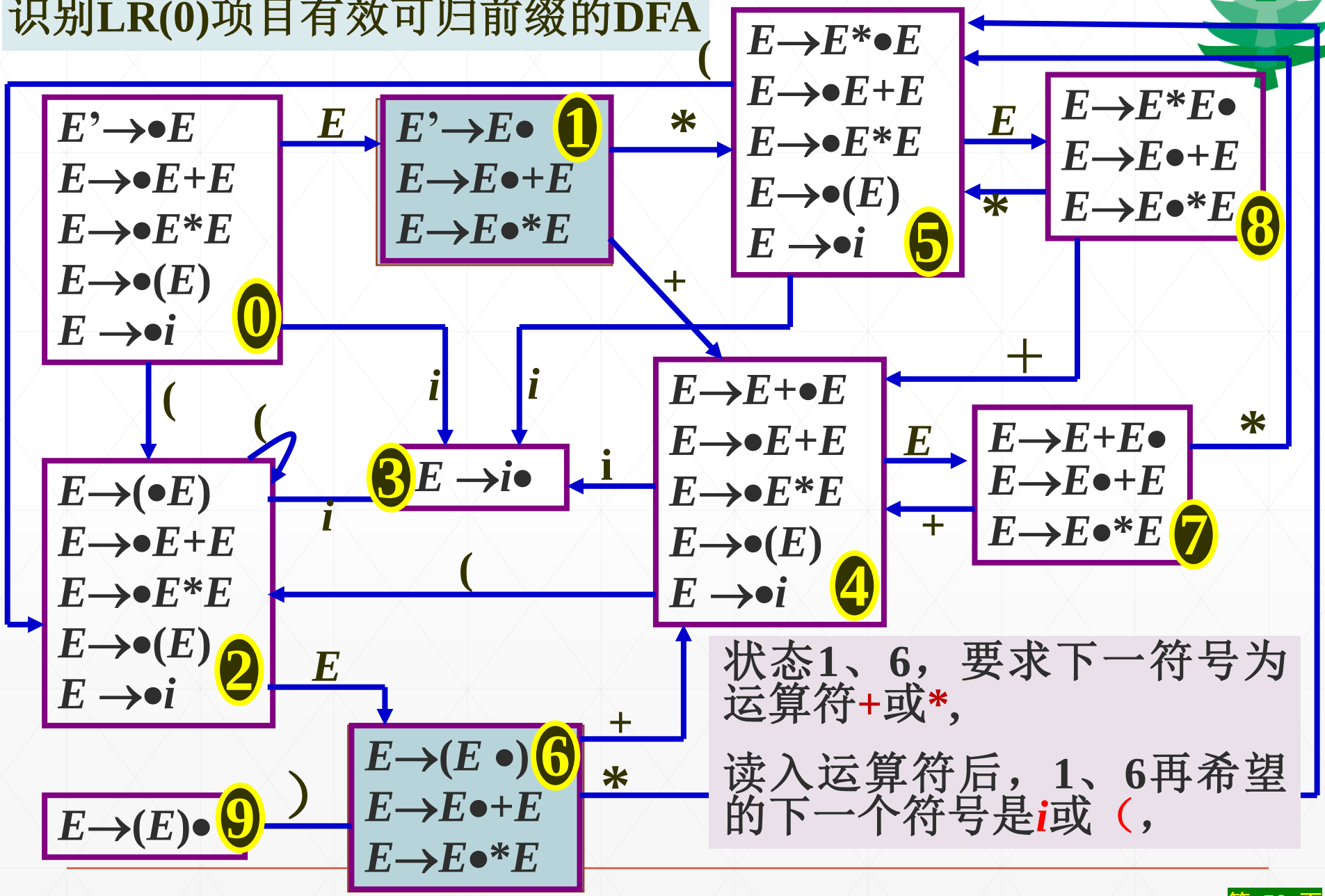
e2错误功能:

删除输入的“）”；　**(F++)**

文法G(E)的LR分析表

状态	ACTION						GOTO
	i	+	*	(	)	#	*E*
0	S3	e1	e1	S2	e2	e1	1
1		S4	S5		e2	acc	
2	S3	e1	e1	S2	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



状态1、6，要求下一符号为运算符+或*，

读入运算符后，1、6再希望的下一个符号是*i*或（，

e3错误信息:

缺少运算符。

e3错误位置:

处于状态1、6时，若遇到*i*或（；

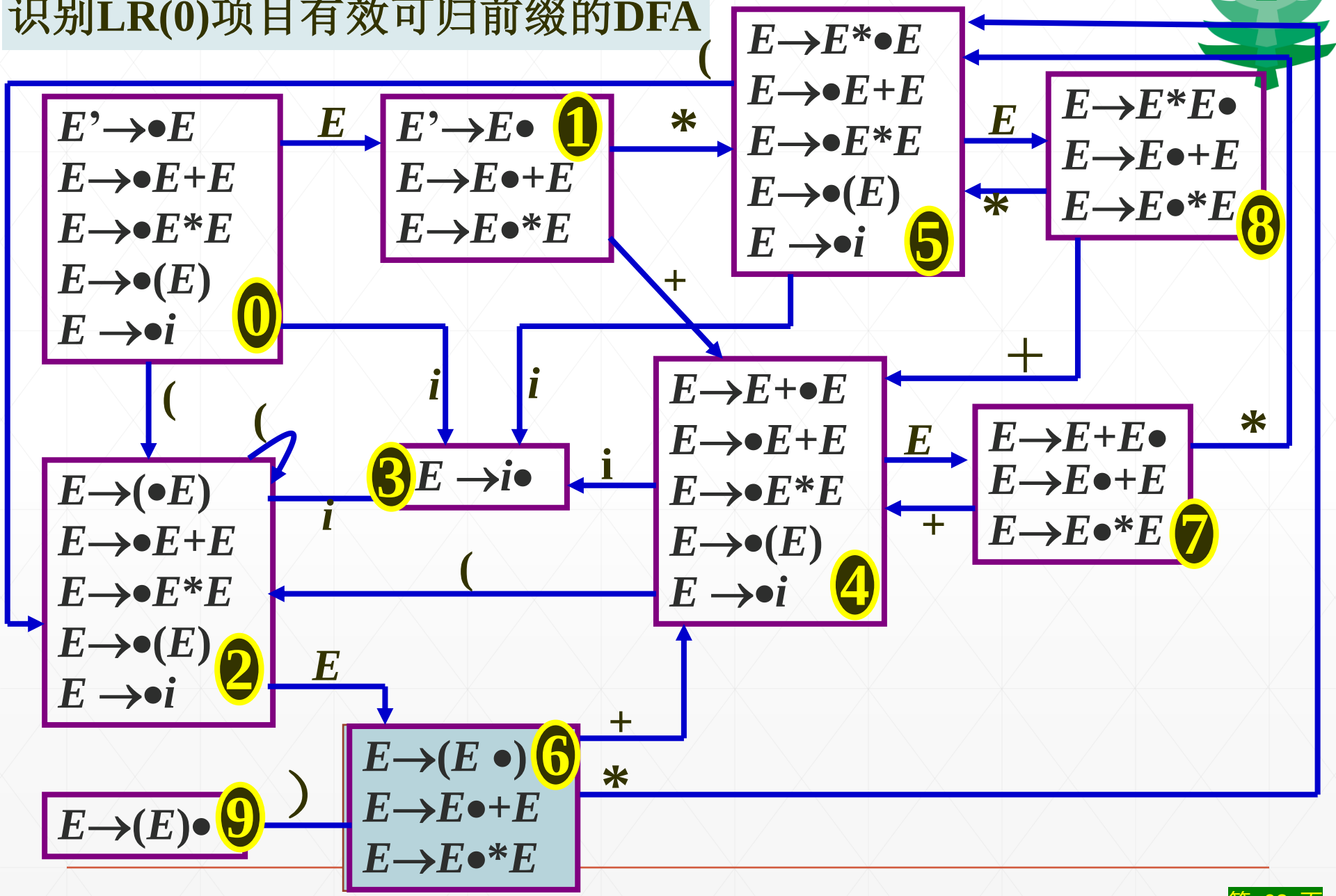
e3错误功能:

将假设的“+” 和状态 4入栈, **(P++)** ；

文法G(E)的LR分析表

状态	ACTION						GOTO
	i	+	*	(	)	#	*E*
0	S3	e1	e1	S2	e2	e1	1
1	e3	S4	S5	e3	e2	acc	
2	S3	e1	e1	S2	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错误信息：

缺少“ ）”。

e4错误位置：：

处于状态6时，若遇到“#”

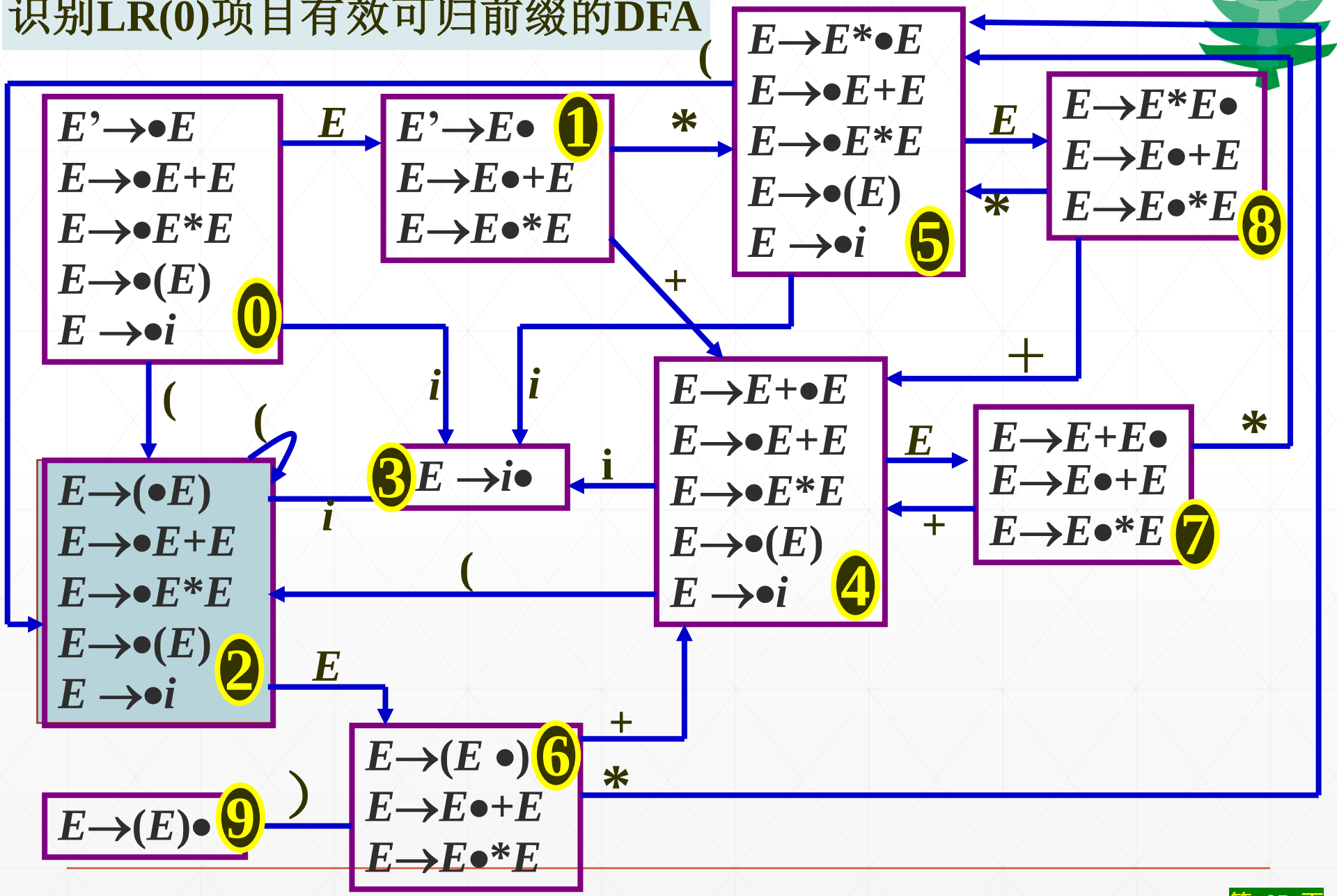
e4错误功能：

将假设的“）”和状态9入栈, (P++)；

文法G(*E*)的LR分析表

状态	ACTION						GOTO
	i	+	*	(	)	#	*E*
0	S3	e1	e1	S2	e2	e1	1
1	e3	S4	S5	e3	e2	acc	
2	S3	e1	e1	S2	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	

识别LR(0)项目有效可归前缀的DFA



e5错误信息：

“(”不配对且缺少运算对象。

e5错误位置：：

处于状态2时，若遇到“#”

e5错误功能：

将栈顶的“（”和状态2弹出，*i*和状态3入栈。

文法G(*E*)的LR分析表

状态	ACTION						GOTO
	i	+	*	(	)	#	*E*
0	S3	e1	e1	S2	e2	e1	1
1	e3	S4	S5	e3	e2	acc	
2	S3	e1	e1	S2	e1	e5	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* #　⇒　$: (*i*+*i*) #　⇒　action(9 , #) = r_3

F

F

E

P

9	)
6	*E*
2	(
0	#

goto(2, *E*) =6

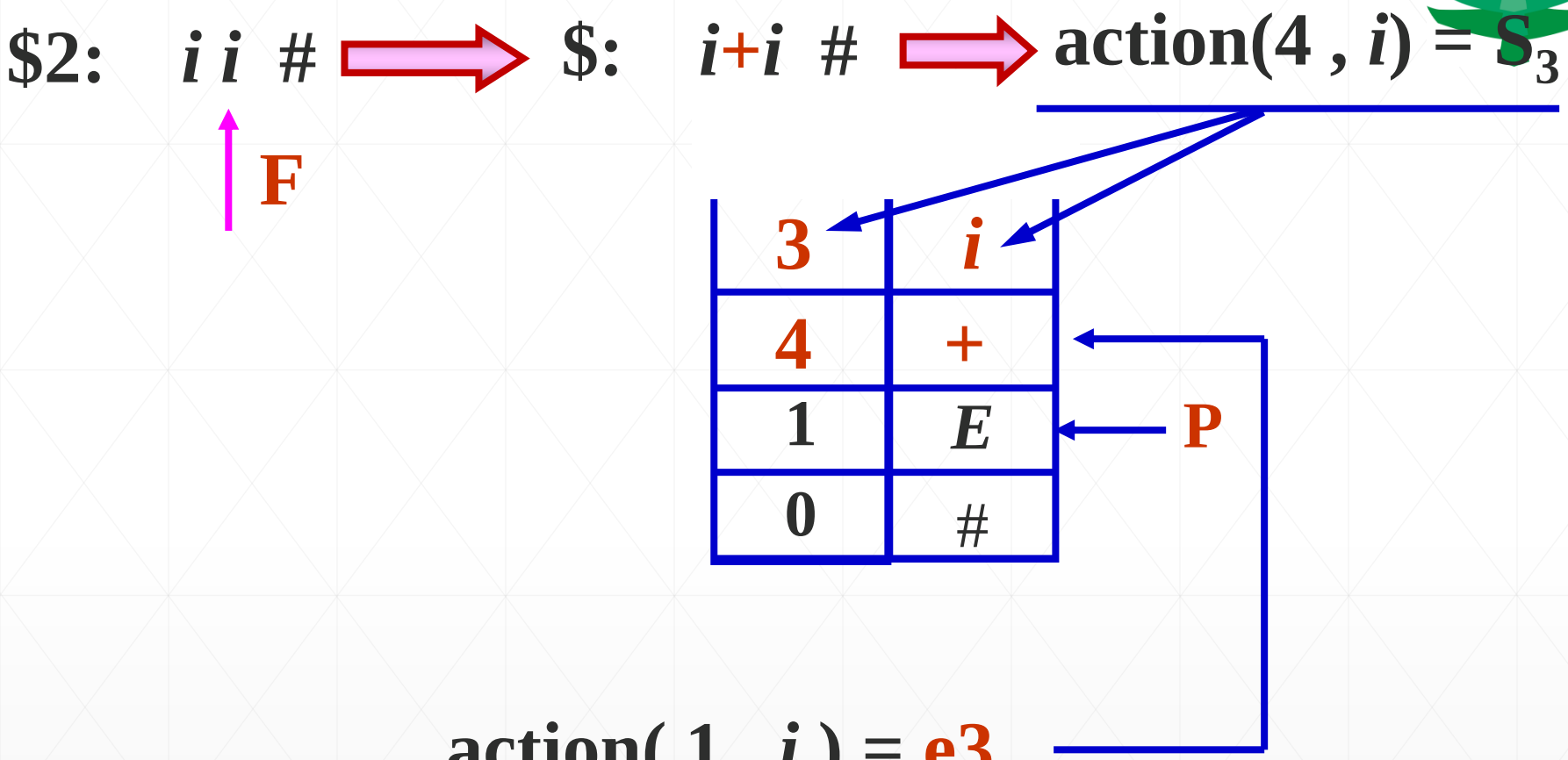
E

P

7	*E*
4	+
6	*E*
2	(
0	#

action(6 , #) = e4

action(7, #) = r_1


$2: i i #
$: i+i #
action(4 , i) = S3
F
3
i
4
+
1
E
P
0
#
action(1 , i) = e3

end