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

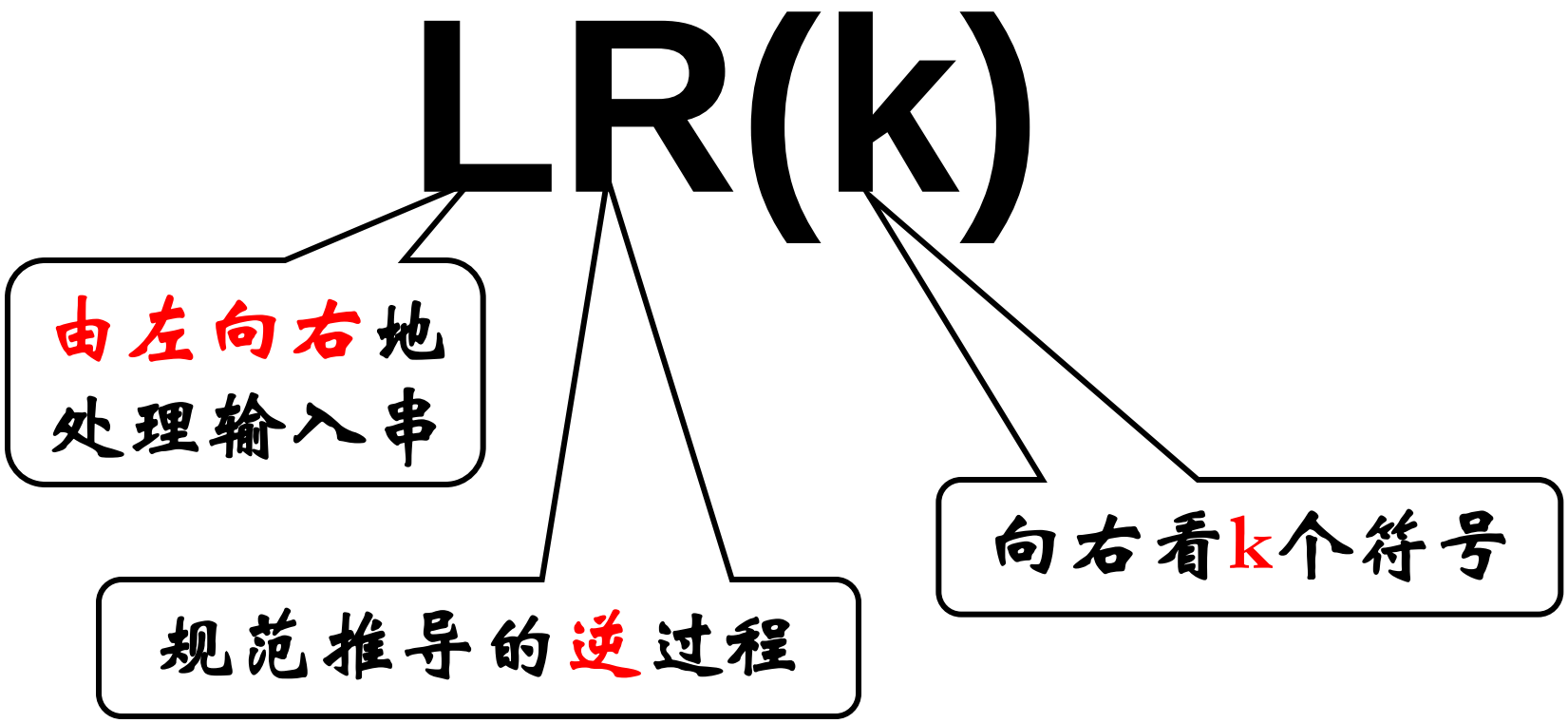
– 利于出错处理

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）

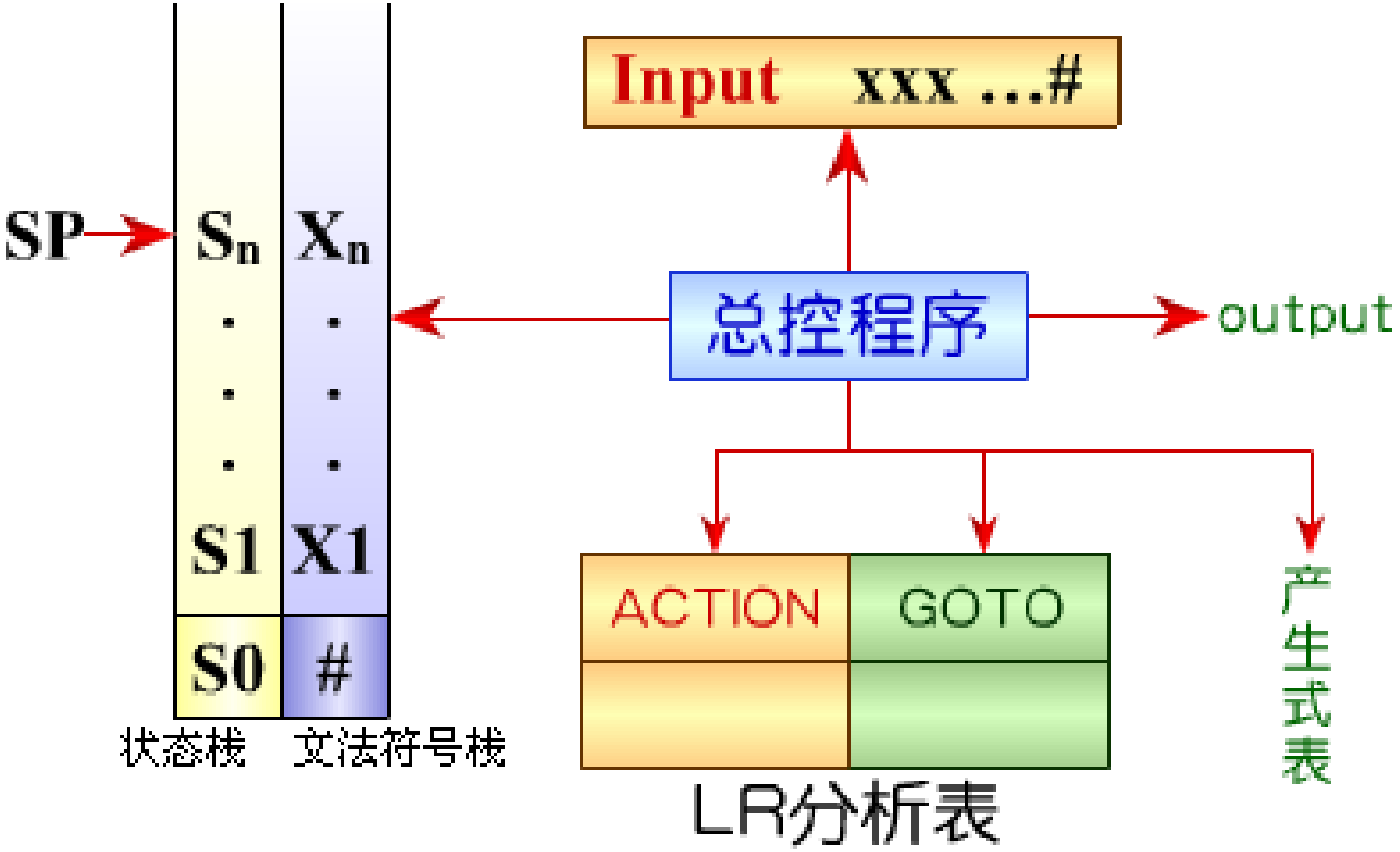


如：LR(0)、SLR(1)、LR(1)、LALR(1)

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


Input xxx …#
SP
Sn Xn
S1 X1
S0 #
状态栈
文法符号栈
总控程序
output
ACTION
GOTO
LR分析表
产生式表

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]=

- S_j：移进， e和j进栈
- r_k：用产生式 (k)A→$X_{m-r+1}X_{m-r+2}\cdots X_m$规约，且 S=GOTO[S’,A]进栈
- acc：接受
- 出错

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1)	#	abbcde#	移进	0	S_2	
2)	#a	bbcde#	移进	02	S_4	
3)	#ab	bcde#	归约(A→b)	024	r_2	3
4)	#aA	bcde#	移进	023	S_6	
5)	#aAb	cde#	归约(A→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→d)	02358	r_4	7
9)	#aAcB	e#	移进	02357	S_9	
10)	#aAcBe	#	归约(S→aAcBe)	023579	r_1	1
11)	#S	#	接受	01	acc	

对输入串abbcde#的LR分析过程

文法G[S]:

(1) S → aAcBe

(2) A → b

(3) A → Ab

(4) B → d

	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			

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→β
            pop |β| 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$S' \overset{*}{\underset{R}{\Rightarrow}} \alpha A\omega \underset{R}{\Rightarrow} \alpha\beta\omega$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'→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' → S[0]

(1) S → aAcBe[1]

(2) A → b[2]

(3) A → Ab[3]

(4) B → d[4]

S '⇒S[0]

⇒aAcBe[1]

⇒aAcd[4]e[1]

⇒aAb[3]cd[4]e[1]

⇒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

ε,S

ε,a,ab

ε ,a,aA,aAb

ε ,a,aA,aAc,aAcd

ε ,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

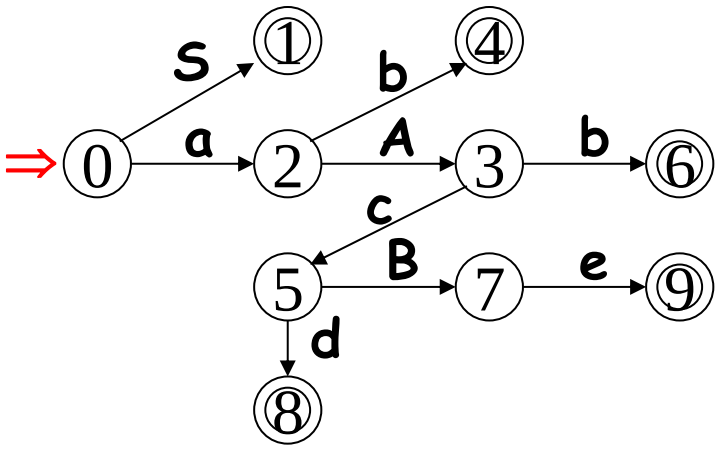
ε,S

ε,a,ab

ε ,a,aA,aAb

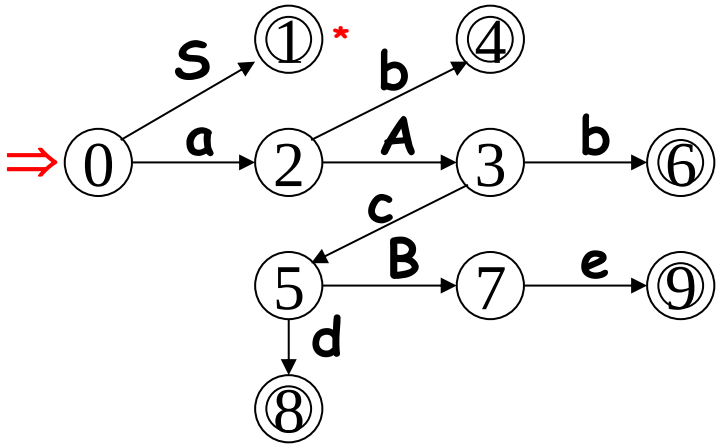
ε ,a,aA,aAc,aAcd

ε ,a,aA,aAc,aAcB,aAcBe



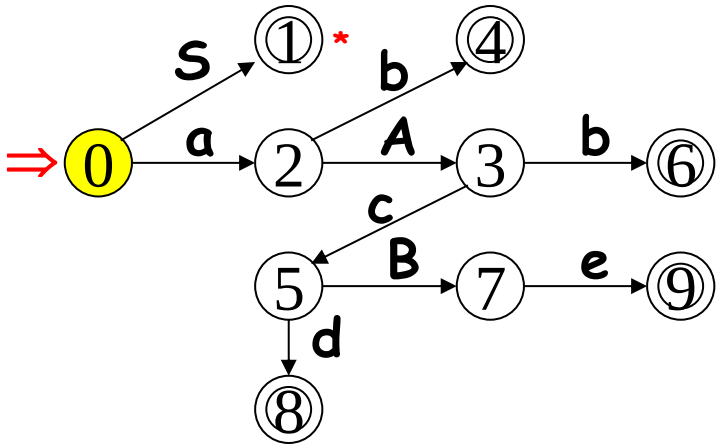
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→Ab)	0236	r3	3
6）	#aA	cde#	移进	023	S_5	
7）	#aAc	de#	移进	0235	S_8	
8）	# aAcd	e#	归约(B→d)	02358	r_4	7
9）	#aAcB	e#	移进	02357	S_9	
10）	#aAcBe	#	归约(S→aAcBe)	023579	r_1	1
11）	#S	#	接受	01	acc	



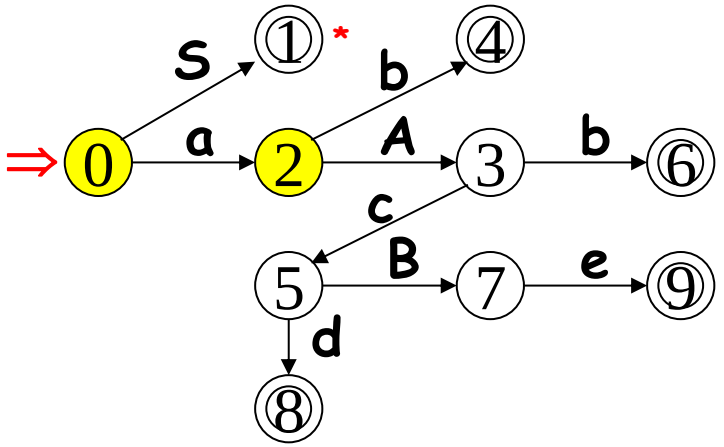
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S2	



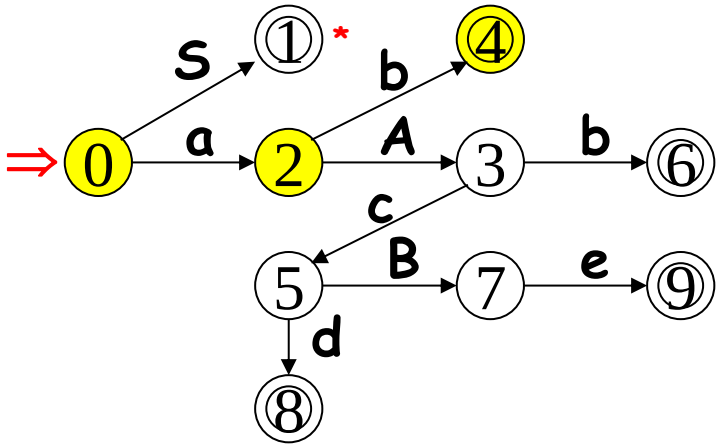
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S4	



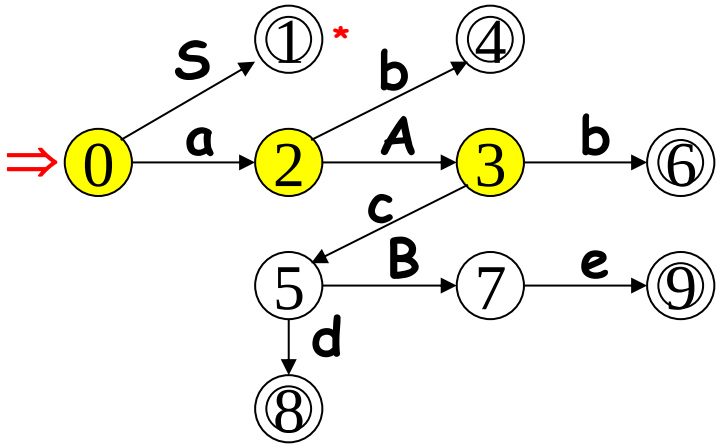
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r2	3



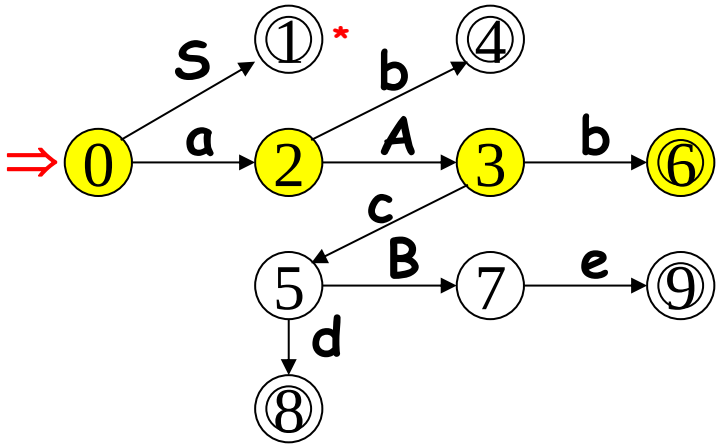
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r2	3
4）	#aA	bcde#	移进	023	S6	



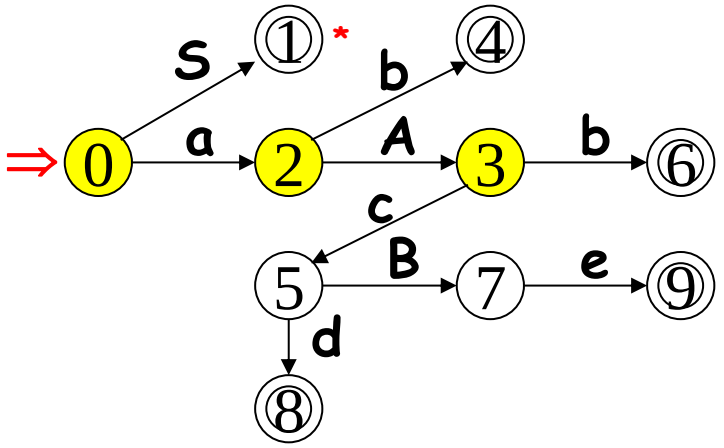
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→Ab)	0236	r3	3



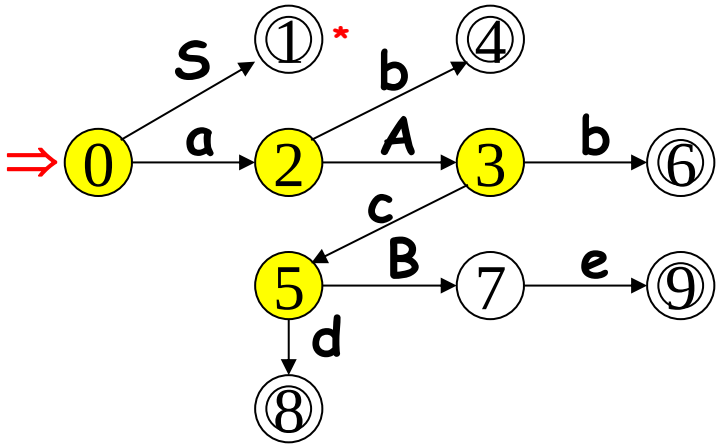
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→Ab)	0236	r3	3
6）	#aA	cde#	移进	023	S5	



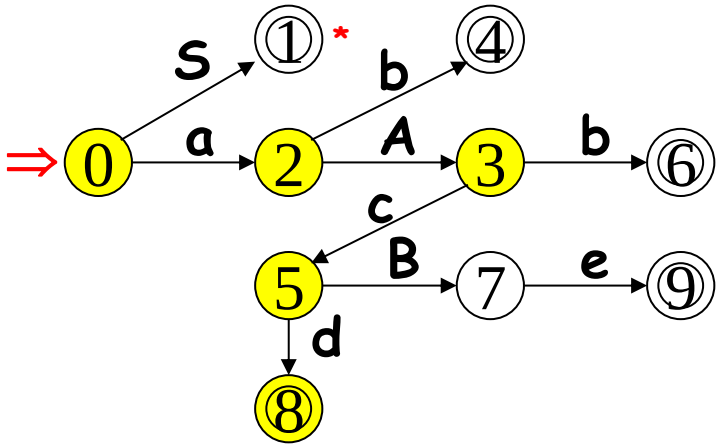
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r_2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→Ab)	0236	r_3	3
6）	#aA	cde#	移进	023	S_5	
7）	#aAc	de#	移进	0235	S8	



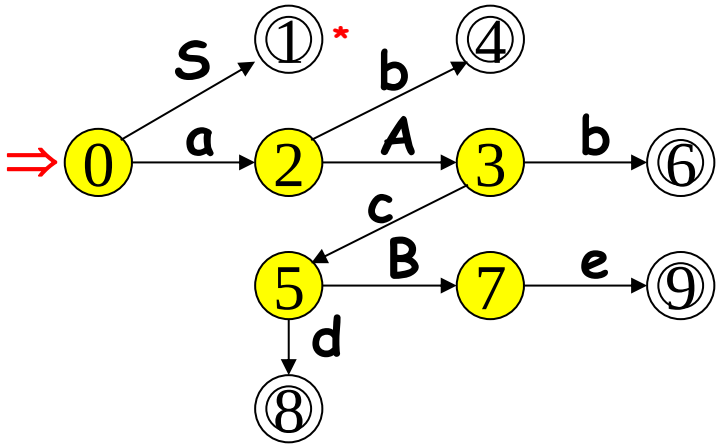
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r_2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→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→d)	02358	r4	7



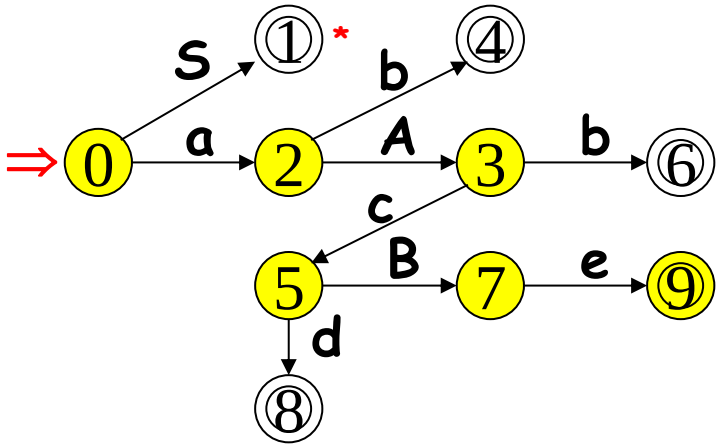
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r_2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→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→d)	02358	r_4	7
9）	#aAcB	e#	移进	02357	S9	



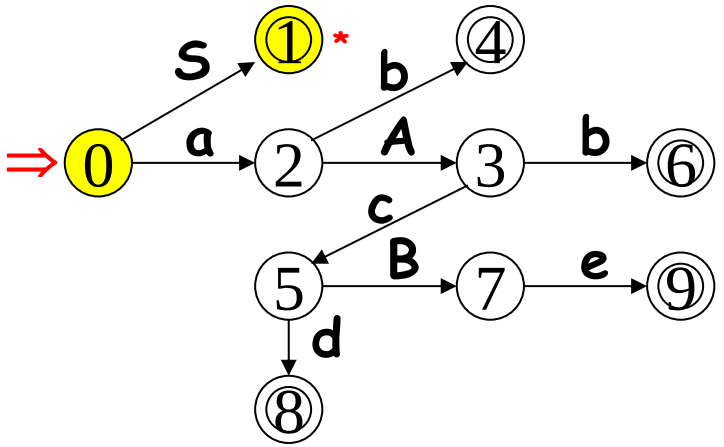
对输入串abbcde的LR分析过程

步骤	符号栈	输入符号串	动作	状态栈	ACTION	GOTO
1）	#	abbcde#	移进	0	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r_2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→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→d)	02358	r_4	7
9）	#aAcB	e#	移进	02357	S_9	
10）	#aAcBe	#	归约(S→aAcBe)	023579	r1	



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	S_2	
2）	#a	bbcde#	移进	02	S_4	
3）	#ab	bcde#	归约(A→b)	024	r_2	3
4）	#aA	bcde#	移进	023	S_6	
5）	#aAb	cde#	归约(A→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→d)	02358	r_4	7
9）	#aAcB	e#	移进	02357	S_9	
10）	#aAcBe	#	归约(S→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α不含有句柄的任何符号

    可记作：$A \rightarrow .\beta$
  - 活前缀αβ1含有句柄的部分符号

    可记作：$A \rightarrow \beta_1 . \beta_2$
  - 活前缀αβ已含有句柄的全部符号

    可记作：$A \rightarrow \beta .$

- 文法的每一个产生式 **A→XY** 定义有下面三个项目：

  A→ •XY

  A→ X•Y

  A→ XY•

  对于**A→ε**的项目只有**A→•**

  以上项目称作**LR(0)项目**。

- 文法G’[S’]:
  (0) S’ → S
  (1) S → aAcBe
  (2) A → b
  (3) A → Ab
  (4) B → d

S’→•S
S’→S•
S→•aAcBe
S→a•AcBe
S→aA•cBe
S→aAc•Be
S→aAcB•e
S→aAcBe•
A→•b
A→b•
A→•Ab
A→A•b
A→Ab•
B→•d
B→d•

- **LR(0)项目分为以下几种：**

设**a**是终结符，**B**是非终结符，α，β ∈**V***

移进项目，形如 **A** →α•**a**β

待约项目，形如 **A** →α•**B**β

归约项目，形如 **A** →α•或 **A**→•

接受项目，形如 **S’** →**S**•

- 识别活前缀的DFA

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

- 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→α·Bβ ∈Closure(I)且B→η∈P，则将B → ·η加进Closure(I)中；
3. 重复执行(2)直到Closure(I)不再增大为止。

I_0 =Closure(S'→•S)：

S'→•S

S→•aAcBe

I_1=Closure(S→a•AcBe)：

S→a•AcBe

A→•b

A→•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) =Closure(J),

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

I_0 =Closure(S'→•S):

S'→•S

S→•aAcBe

a

I_1=Closure(S→a•AcBe):

S→a•AcBe

A→•b

A→•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]:

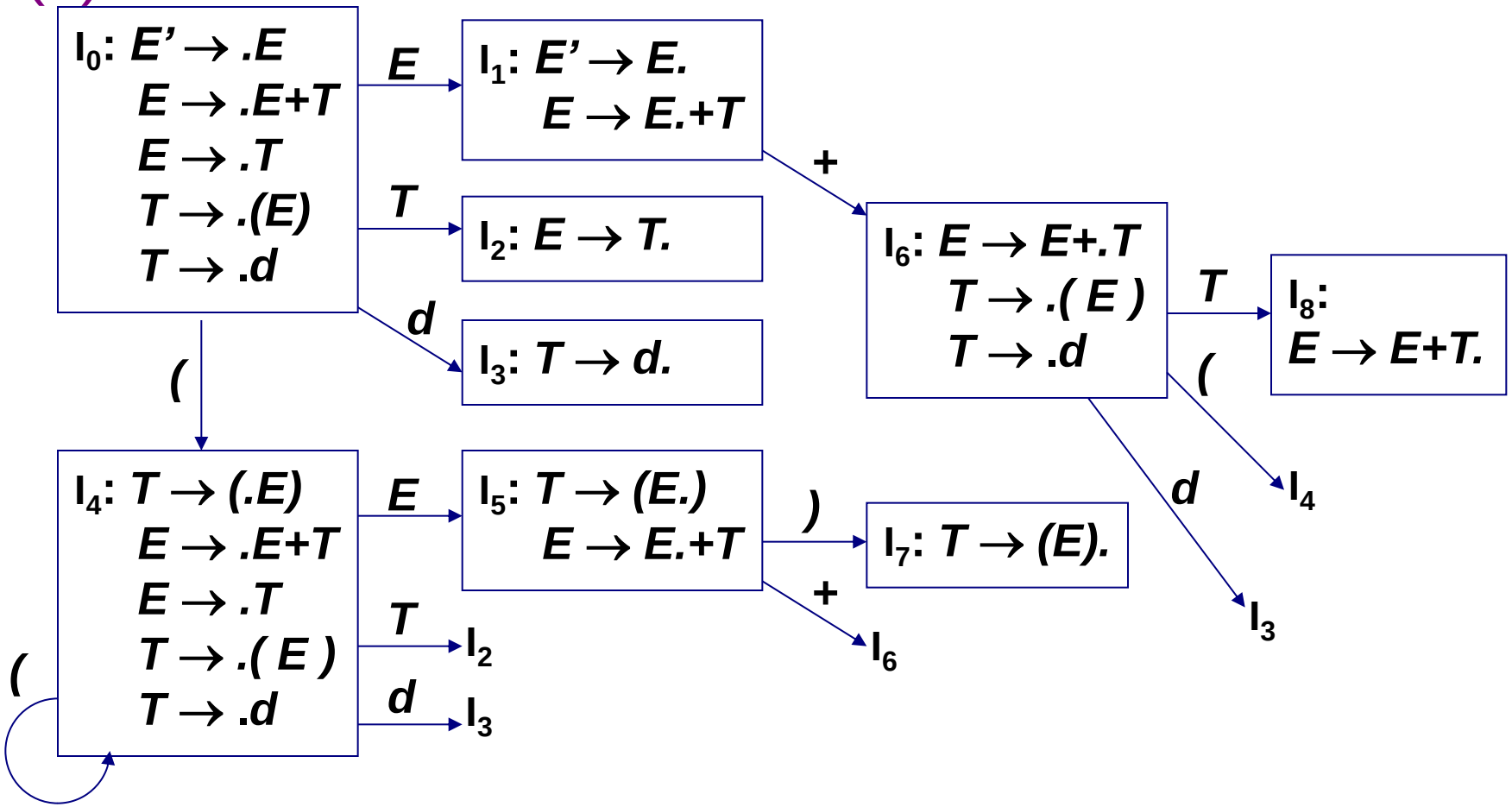
(1) $E \rightarrow E+T$

(2) $E \rightarrow T$

(3) $T \rightarrow (E)$

(4) $T \rightarrow d$

$E' \rightarrow E$



- 例

  G’[S’]:

  (0) S’ → S

  (1) S → aAcBe

  (2) A → b

  (3) A → Ab

  (4) B → d

LR(0)分析表

- 若项目A→α•aβ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→α•属于I_k, 那么，对任何终结符a, 置ACTION[k, a]为“用产生式A→α(文法G`的第j个产生式)进行规约”，简记为“r_j”;
- 若项目S`→S•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

	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			

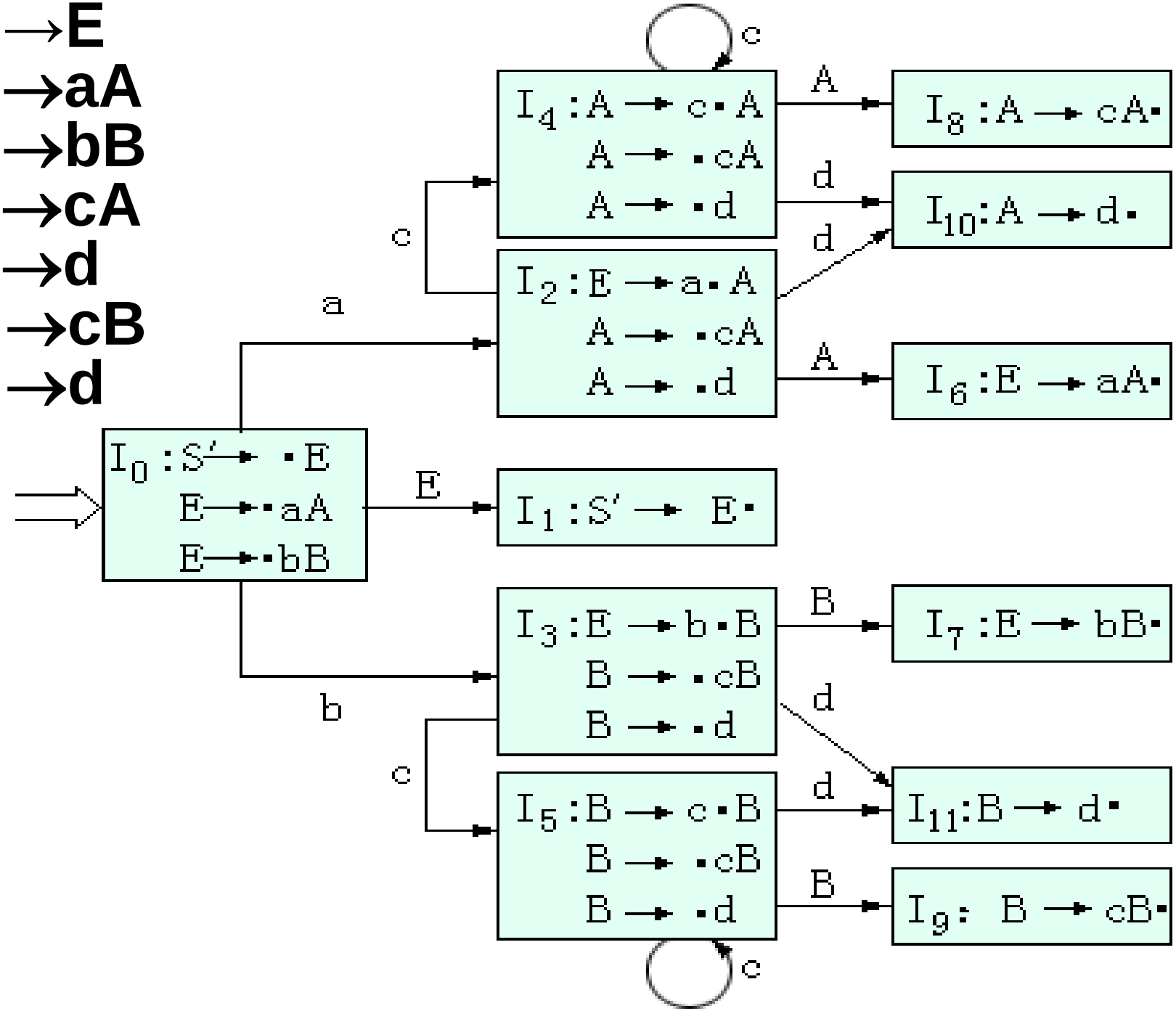
- **LR(0)**文法

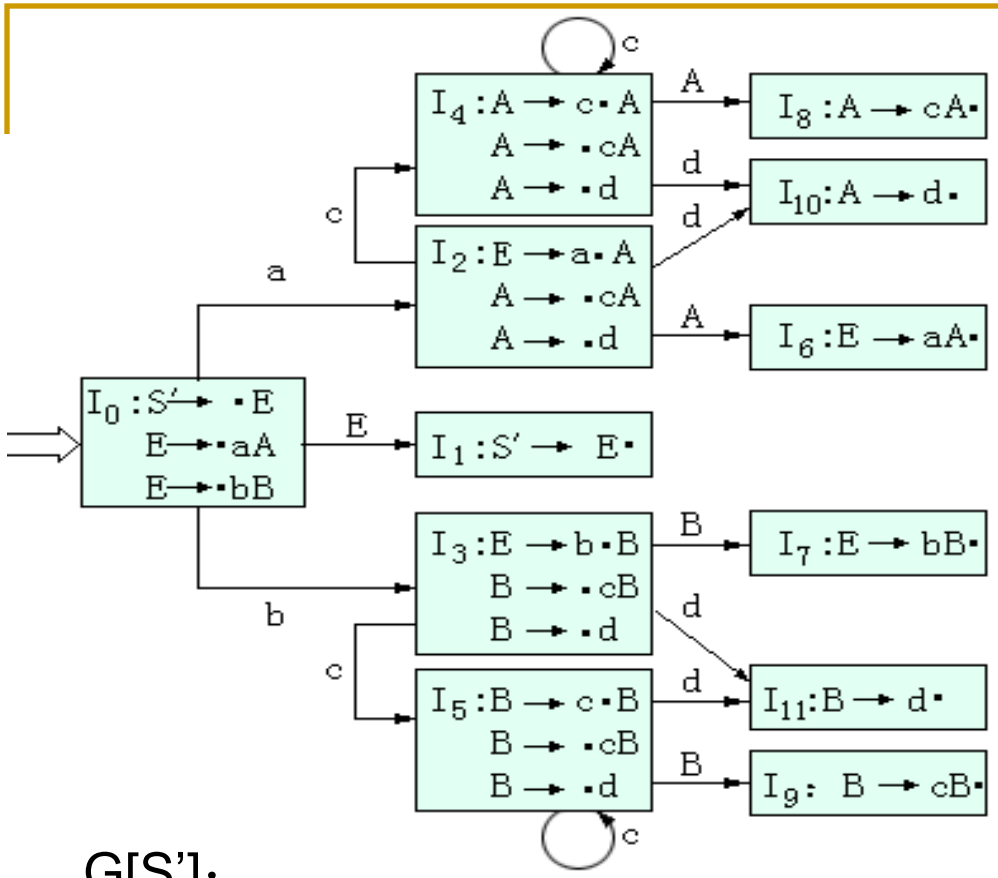
对于一个上下文无关文法，如果能够构造一张**LR(0)**分析表，使得它的每一个条目均是唯一的，则称该上下文无关文法为**LR(0)**文法。

- 练习

G[S’]：
(0) S’→E
(1) E →aA
(2) E →bB
(3) A →cA
(4) A →d
(5) B →cB
(6) B →d

(0) S'→E
(1) E →aA
(2) E →bB
(3) A →cA
(4) A →d
(5) B →cB
(6) B →d





G[S']:

(0) S'→E

(1) E →aA

(2) E →bB

(3) A →cA

(4) A →d

(5) B →cB

(6) B →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′→ E

（1） E→ E+T

（2） E→ T

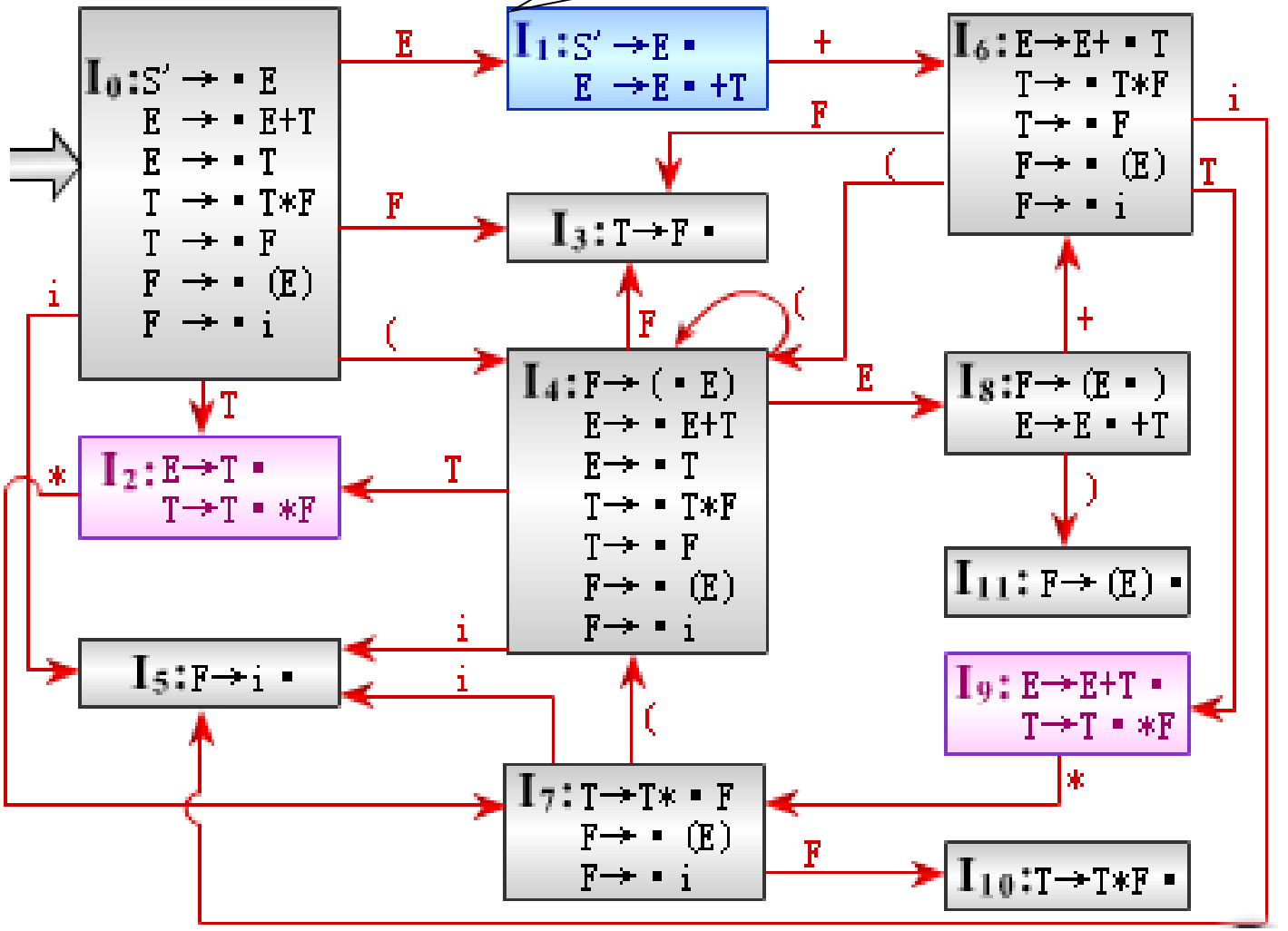
（3） T→ T*F

（4） T→ F

（5） F→ (E)

（6） F→ 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)
  $I = \{X \rightarrow \alpha \bullet b\beta\ ,\ A \rightarrow \gamma \bullet,\ B \rightarrow \delta \bullet\ \}$
  若有$FOLLOW(A) \cap FOLLOW(B) = \varnothing$
  $FOLLOW(A) \cap \{b\} = \varnothing$
  $FOLLOW(B) \cap \{b\} = \varnothing$
  则当状态I面临输入符号a时，
  1) 若a=b，则移进；
  2) 若$a \in FOLLOW(A)$, 则用产生式 $A \rightarrow \gamma$ 进行归约；
  3) 若$a \in FOLLOW(B)$, 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→α•aβ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→α•属于I_k，则对a为任何终结符或‘#’，且满足a∈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‘→S•属于I_k，则置ACTION[k,#] = acc
  - 其它填上“报错标志”

(0) S′→ E

(1) E→ E+T

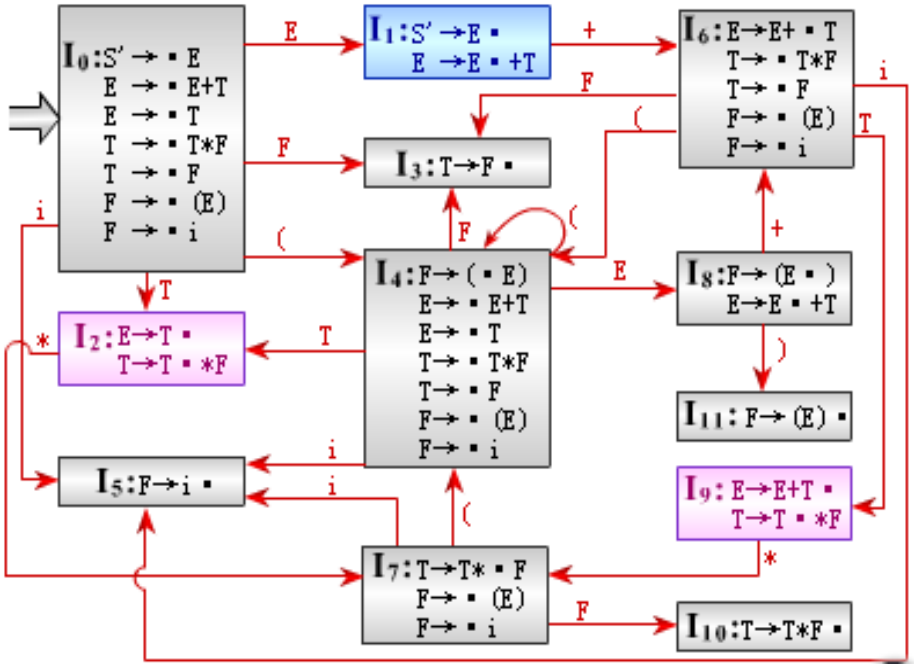
(2) E→ T

(3) T→ T*F

(4) T→ F

(5) F→ (E)

(6) F→ 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 →aAd
(2) S →bAc
(3) S →aec
(4) S →bed
(5) A →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 →x.y,a]

a为向前搜索符，它只对归约项目[U →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

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

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

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

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

6.6 二义性文法在LR分析中的应用

- 例

  G'[E']:

  (0) E' → E

  (1) E → E+E

  (2) E→ E*E

  (3) E → (E)

  (4) E → id

I_0: $E' \to \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot (E)$
$E \to \cdot \mathbf{id}$

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

I_2: $E \to (\cdot E)$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot (E)$
$E \to \cdot \mathbf{id}$

I_3: $E \to \mathbf{id}\cdot$

I_4: $E \to E + \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot (E)$
$E \to \cdot \mathbf{id}$

I_5: $E \to E * \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot (E)$
$E \to \cdot \mathbf{id}$

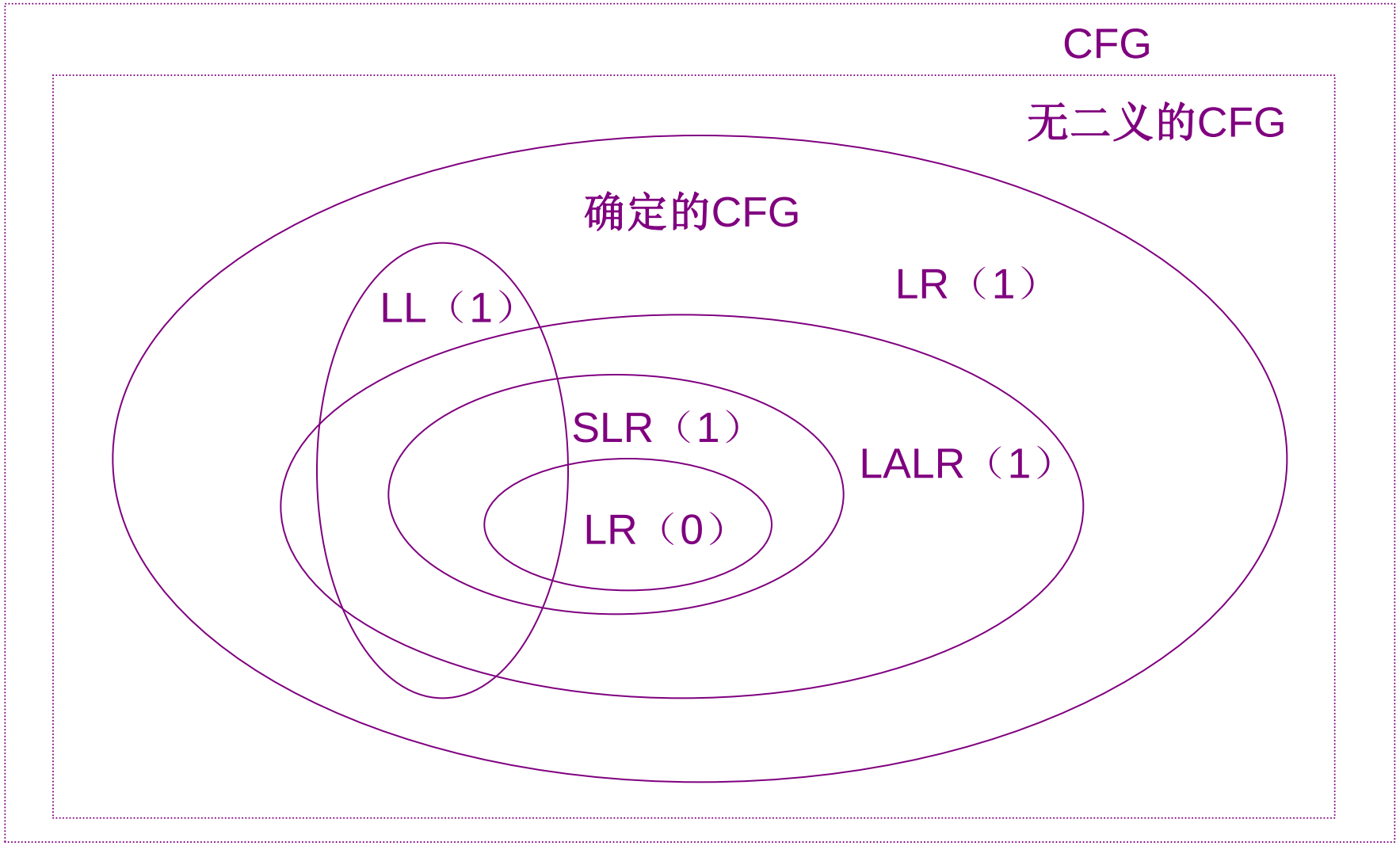
I_6: $E \to (E\cdot)$
$E \to E\cdot + E$
$E \to E\cdot * E$

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

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

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

几类分析文法之间的关系



练习

- 文法D[S]为：
- (1) S → AB
  (2) A→ aBa|ε
  (3) B →bAb|ε
- 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'→S

(1) S→SbAb

(2) S→A

(3) A→a

(4) A→ε

- **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。**