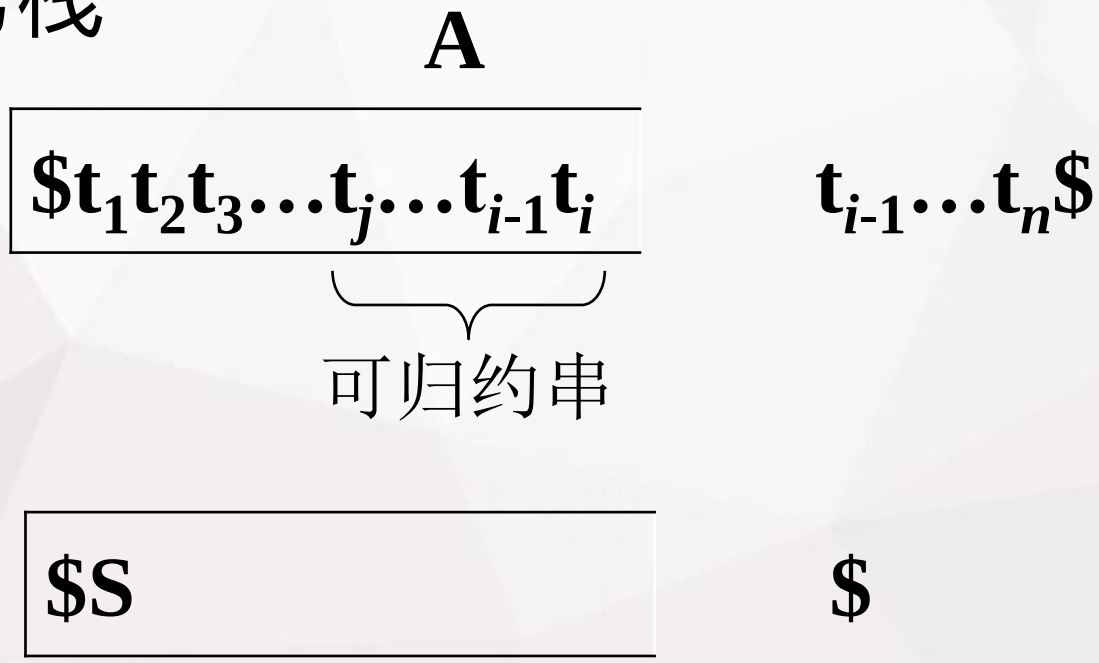
4.3 自下而上分析法的一般原理

移进-归约法(shift-reduce)

符号栈



4.3 自下而上分析法的一般原理

设有文法G[S]:

(1) S→ aAB
(2) A→ b
(3) A→ A b
(4) B→ d
(5) B→ cBe

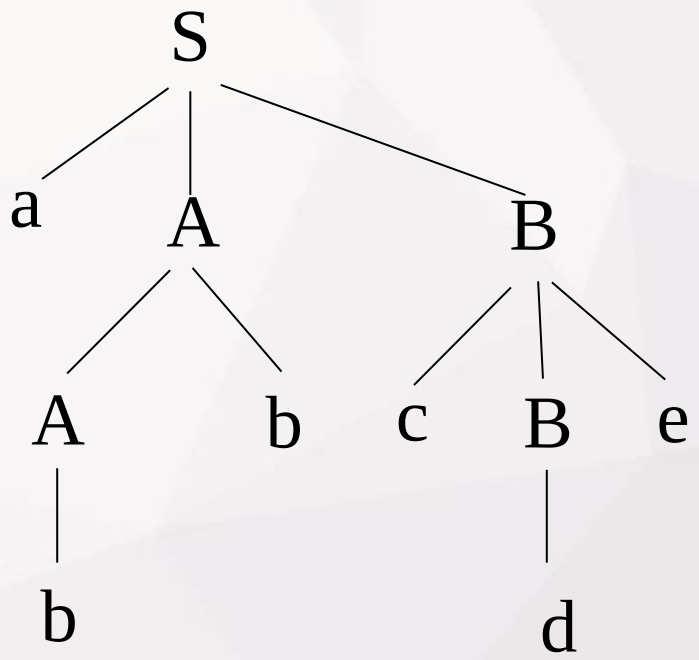
对输入串abbcde进行自下而上语法分析,检查该符号串是否该文法的正确句子。

4.3 自下而上分析法的一般原理

(1)S→ aAB　(2) A→ b　(3) A→ A b　(4) B→ d　(5) B→cBe

输入串abbcde

符号栈	输入串
$	abbcde$
$a	bbcde$
$ab	bcde$
$aA	bcde$
$aAb	cde$
$aA	cde$
$aAc	de$
$aAcd	e$
$aAcB	e$
$aAcBe	$
$aAB	$
$S	$



4.5 LR分析法

1965，Donald E. Knuth

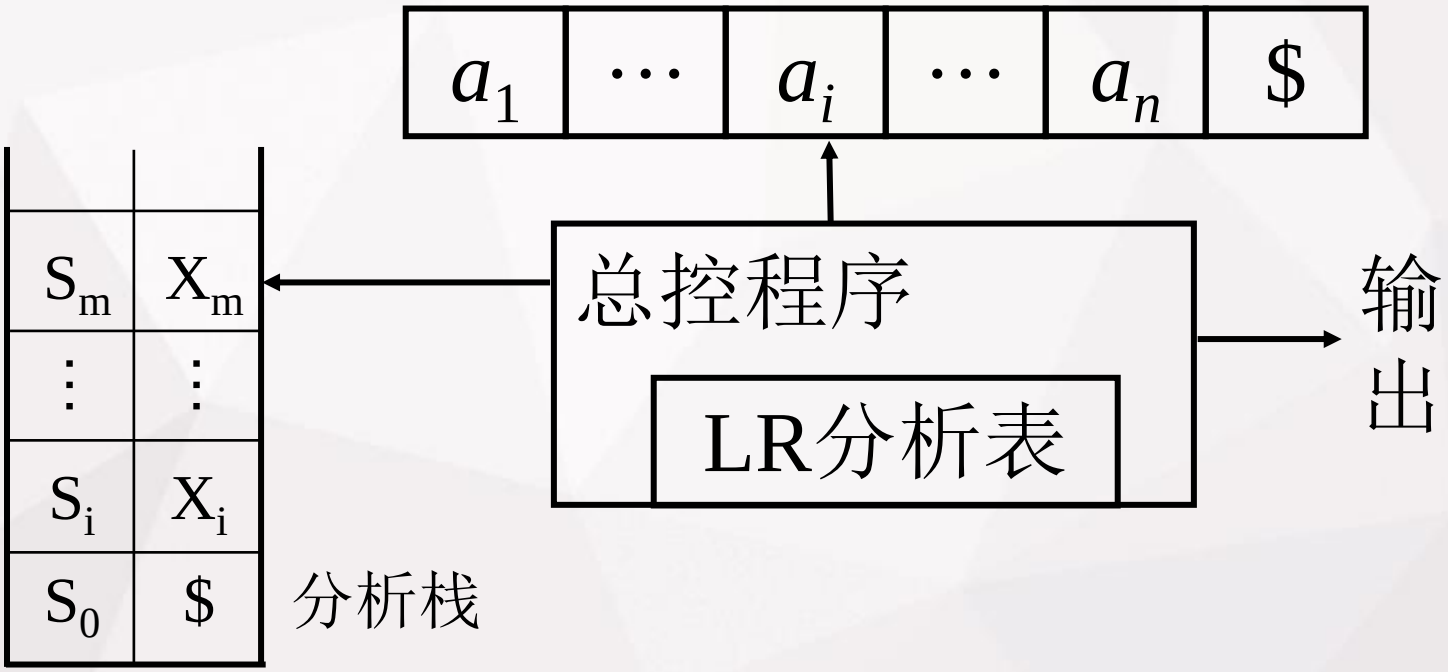
LR分析法是一种自下而上进行规范归约的语法分析方法。

大多数用无二义性上下文无关文法描述的语言都可以用LR分析法进行有效的无回溯分析。分析速度快，并能准确及时地指出输入串的语法错误和出错位置。

主要缺点：手工构造LR分析器的工作量相当大，具体实现较困难。

4.5.1 LR分析器的逻辑结构和工作过程

LR分析器的逻辑结构



4.5.1 LR分析器的逻辑结构和工作过程

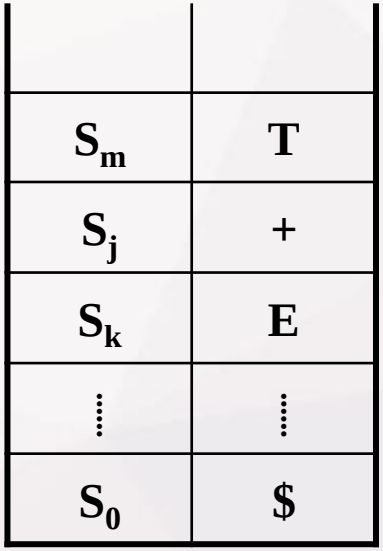
1. 分析栈：分析栈用来存放分析过程中的历史和展望信息。

例如，对文法

G[E]: E→E+T| T

T→T*F | F

F→(E) | id



4.5.1 LR分析器的逻辑结构和工作过程

2. LR分析表： LR分析器的核心部分

0. S’→S
1. S→A
2. S→B
3. A→*a*A*b*
4. A→*c*
5. B→*a*B*b*
6. B→*d*

状态	ACTION					GOTO		
	a	*b*	*c*	*d*	$	S	A	B
0	s_4		s_5	s_6		1	2	3
1					acc			
2	r_1	r_1	r_1	r_1	r_1			
3	r_2	r_2	r_2	r_2	r_2			
4	s_4		s_5	s_6			7	9
5	r_4	r_4	r_4	r_4	r_4			
6	r_6	r_6	r_6	r_6	r_6			
7		s_8						
8	r_3	r_3	r_3	r_3	r_3			
9		s_{10}						
10	r_5	r_5	r_5	r_5	r_5			

4.5.2 LR(0)分析法

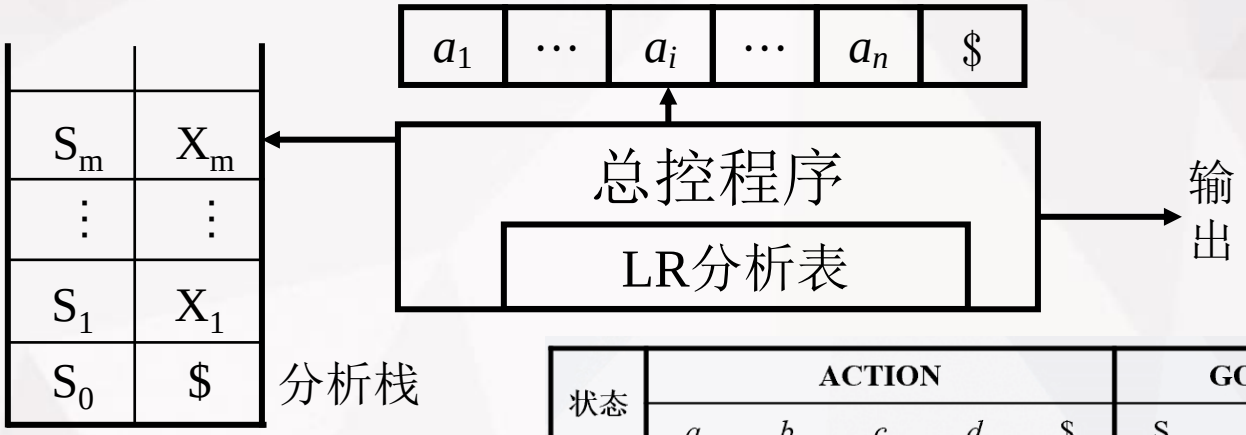
步骤	栈中状态	栈中符号	输入串	分析动作
1	0	$	*aacbb*$	s_4
2	04	$*a*	*acbb*$	s_4
3	044	$*aa*	*cbb*$	s_5
4	0445	$*aac*	*bb*$	用第4条规则A→*c*归约
5	0447	$*aa*A	*bb*$	s_8
6	04478	$*aa*A*b*	*b*$	用第3条规则A→*a*A*b*归约
7	047	$*a*A	*b*$	s_8
8	0478	$*a*A*b*	$	用第3条规则A→*a*A*b*归约
9	02	$A	$	用第1条规则S→A归约
10	01	$S	$	acc

4.5.1 LR分析器的逻辑结构和工作过程

3. 总控程序

while (ACTION[S,*a*]!=acc)
{ *if* (ACTION[S,*a*]==s_i)
{ 状态*i*和输入符号*a*进栈；
将下一个输入符号读入*a*中；}
else if (Action[S,*a*]==r_i)
{ 用第*i*条规则A→*α*归约；
将|*α*|个状态和|*α*|个符号退栈；
当前栈顶状态为S’，状态GOTO[S’,A]=S”和A进栈；}
else if （ Action[S,*a*]== ERROR）
error( )；
}

4.5.2 LR(0)分析法



构造LR分析表的基本思想:从给定的上下文无关文法直接构造识别文法所有规范句型活前缀的DFA，然后再将DFA转换成一张LR分析表。

状态	ACTION					GOTO		
	a	*b*	*c*	*d*	$	S	A	B
0	s_4		s_5	s_6		1	2	3
1					acc			
2	r_1	r_1	r_1	r_1	r_1			
3	r_2	r_2	r_2	r_2	r_2			
4	s_4		s_5	s_6			7	9
5	r_4	r_4	r_4	r_4	r_4			
6	r_6	r_6	r_6	r_6	r_6			
7		s_8						
8	r_3	r_3	r_3	r_3	r_3			
9		s_{10}						
10	r_5	r_5	r_5	r_5	r_5			

4.5.2 LR(0)分析法

步骤	栈中状态	栈中符号	输入串	分析动作
1	0	$	*aacbb*$	s_4
2	04	$*a*	*acbb*$	s_4
3	044	$*aa*	*cbb*$	s_5
4	0445	$*aac*	*bb*$	用第4条规则A→*c*归约
5	0447	$*aa*A	*bb*$	s_8
6	04478	$*aAb*	*b*$	用第3条规则A→*a*A*b*归约
7	047	$*a*A	*b*$	s_8
8	0478	$*a*A*b*	$	用第3条规则A→*a*A*b*归约
9	02	$A	$	用第1条规则S→A归约
10	01	$S	$	acc

4.5.2 LR(0)分析法

- 文法规范句型的活前缀
  - 字符串的前缀是指字符串的任意首部：例如，字符串*abc*的前缀有*ε,a,ab,abc*。
  - 规范句型活前缀是指规范句型中不包含句柄右边任何符号的前缀。

  注意，活前缀可以是一个或者是若干个规范句型的前缀。
- LR(0)项目

活前缀与句柄之间的关系：⑴已含有句柄的全部符号；⑵只含有句柄的一部分符号；⑶不含有句柄的任何符号。

$$A\rightarrow\alpha\cdot \quad A\rightarrow \alpha_1\cdot\alpha_2 \quad A\rightarrow\cdot\alpha$$

我们把文法G中右部标有圆点的规则称为G的一个LR(0)项目。 注意：对空规则 A→ε，仅有LR(0)项目A→• 。

4.5.2 LR(0)分析法

由于不同的LR(0)项目反映了在分析过程中栈顶的不同情况，因此，我们可以根据圆点位置和圆点后是终结符还是非终结符，将一个文法的全部LR(0)项目进行分类。

- 归约项目
- 移进项目
- 待约项目
- 接受项目

4.5.2 LR(0)分析法

① **归约项目**，形如$A\rightarrow\alpha\bullet$，其中$\alpha\in(V_N\cup V_T)^*$，即圆点在最右端的项目，它表示一个规则的右部已分析完，句柄已形成，应该按此规则进行归约。

② **移进项目**，形如$A\rightarrow\alpha\bullet\ a\beta$，其中$\alpha$，$\beta\in(V_N\cup V_T)^*$，即圆点后面为终结符的项目，它表示期待从输入串中移进一个符号，以待形成句柄。

③ **待约项目**，形如$A\rightarrow\alpha\bullet\ B\beta$，其中$\alpha$，$\beta\in(V_N\cup V_T)^*$，$B\in V_N$，即圆点后面为非终结符的项目，它表示期待从余留的输入串中进行归约得到B，然后分析A的右部。

④ **接受项目**，形如$S'\rightarrow S\bullet$，其中S'为即文法开始符号的归约项目。 S'为左部的规则只有一个，它是归约项目的特殊情况，它表示整个句子已经分析完毕，可以接受。

4.5.2 LR(0)分析法

分析表的每个状态表示一类活前缀，可以通过DFA来描述状态之间的关系。

对于构成识别文法规范句型活前缀DFA，其每一个状态是由若干个LR(0)项目所组成的集合，称为LR(0)项目集。

为了使“接受”项目唯一，我们对文法G[S]进行拓广，引入新规则S' →S，得到拓广文法G’[S’]。

有效项目：如果存在最右推导过程$S \overset{*}{\Rightarrow} \alpha\beta_1\beta_2\omega$，称项目A→ $\beta_1 \cdot \beta_2$ 对活前缀$\alpha\,\beta_1$有效。

4.5.2 LR(0)分析法

(1)定义闭包函数

设I是拓广文法G'的一个LR(0)项目集，定义和构造I的闭包CLOSURE(I)如下：

(a)I中的任何一个项目都属于CLOSURE(I)。

(b)若A→α• Bβ属于CLOSURE(I)，则每一形如 B →•γ的项目也属于CLOSURE(I)。

(c) 重复(b)直到CLOSURE(I)不再增大为止。

例 **0.** S’→S **1.** S→A **2.** S→B **3.** A→**a**Ab
4. A→c **5.** B→**a**Bb **6.** B→d

令I={S'→·S }，则CLOSURE(I)

={ S'→·S, S→·A, S→·B, A→·**a**Ab, A→·c, B→·**a**Bb，B→·d }

=I_0

4.5.2 LR(0)分析法

(2) 定义状态转移函数GO

设I是拓展文法G'的任一个项目集，X为一文法符号，定义状态转移函数GO(I,X)如下：

GO(I , X)=CLOSURE(J)，J={A→aX·β | A→a·X β∈I}

I_0={ S'→·S , S→·A , S→·B, A→·aAb , A→·c , B→·aBb , B→·d }

GO(I_0 , S)= CLOSURE({S'→S· }) ={ S'→S· } = I_1

GO(I_0 , a) =CLOSURE({A→a·Ab , B→a·Bb})

={ A→a·Ab , A→·aAb , A→·c
B→a·Bb , B→·aBb , B→·d }

=I_4

4.5.2 LR(0)分析法

(3) 构造识别文法规范句型活前缀DFA的方法

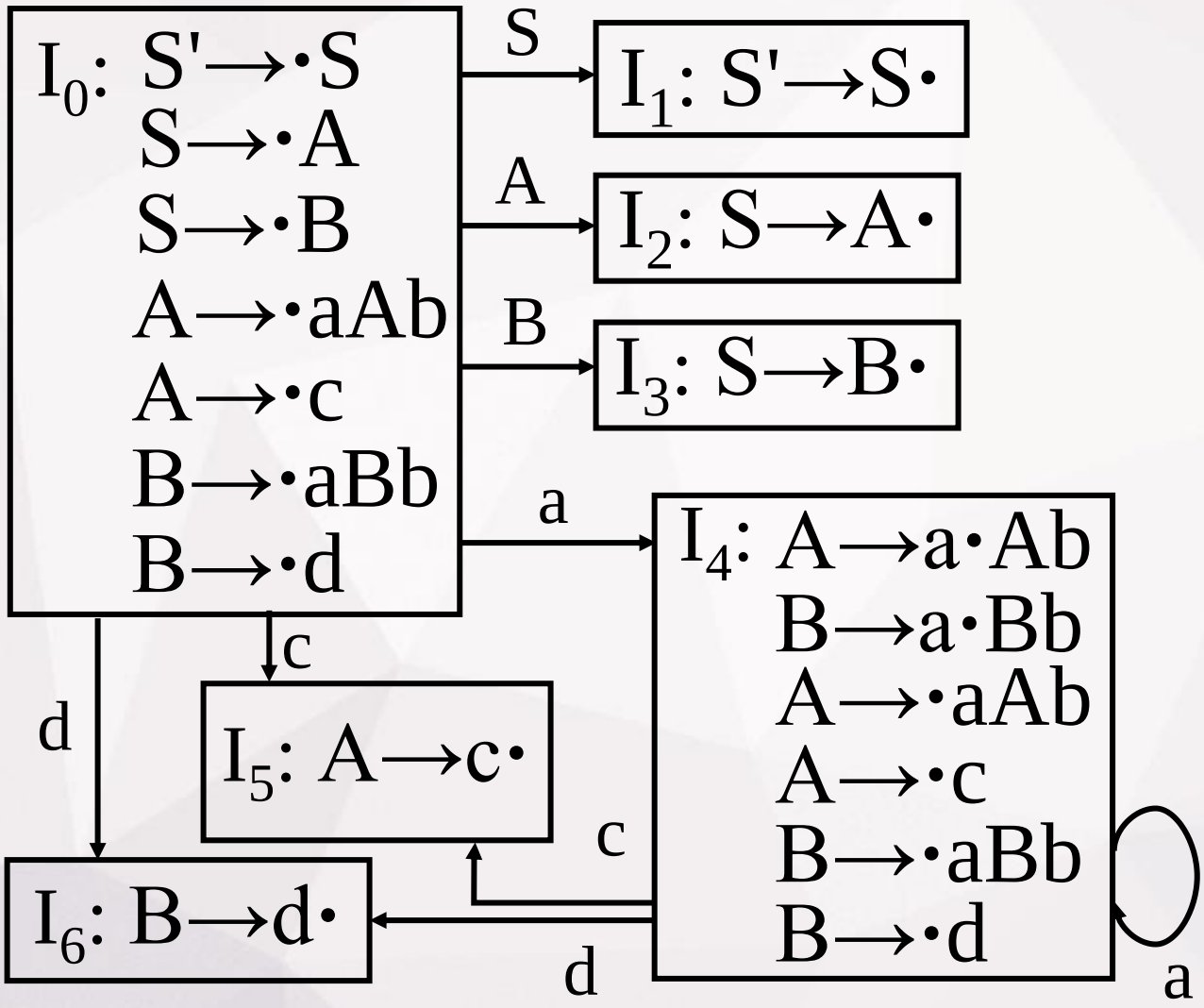
(a) 求CLOSURE{S'→•S}，得到初态项目集。

(b) 对初态项目集或其它已构造的项目集，应用状态转移函数GO(I,X)求出新的项目集（后继状态）。

(c) 重复(b)直到不出现新的项目集（新状态）为止。

(d) 转移函数GO建立状态之间的连结关系。

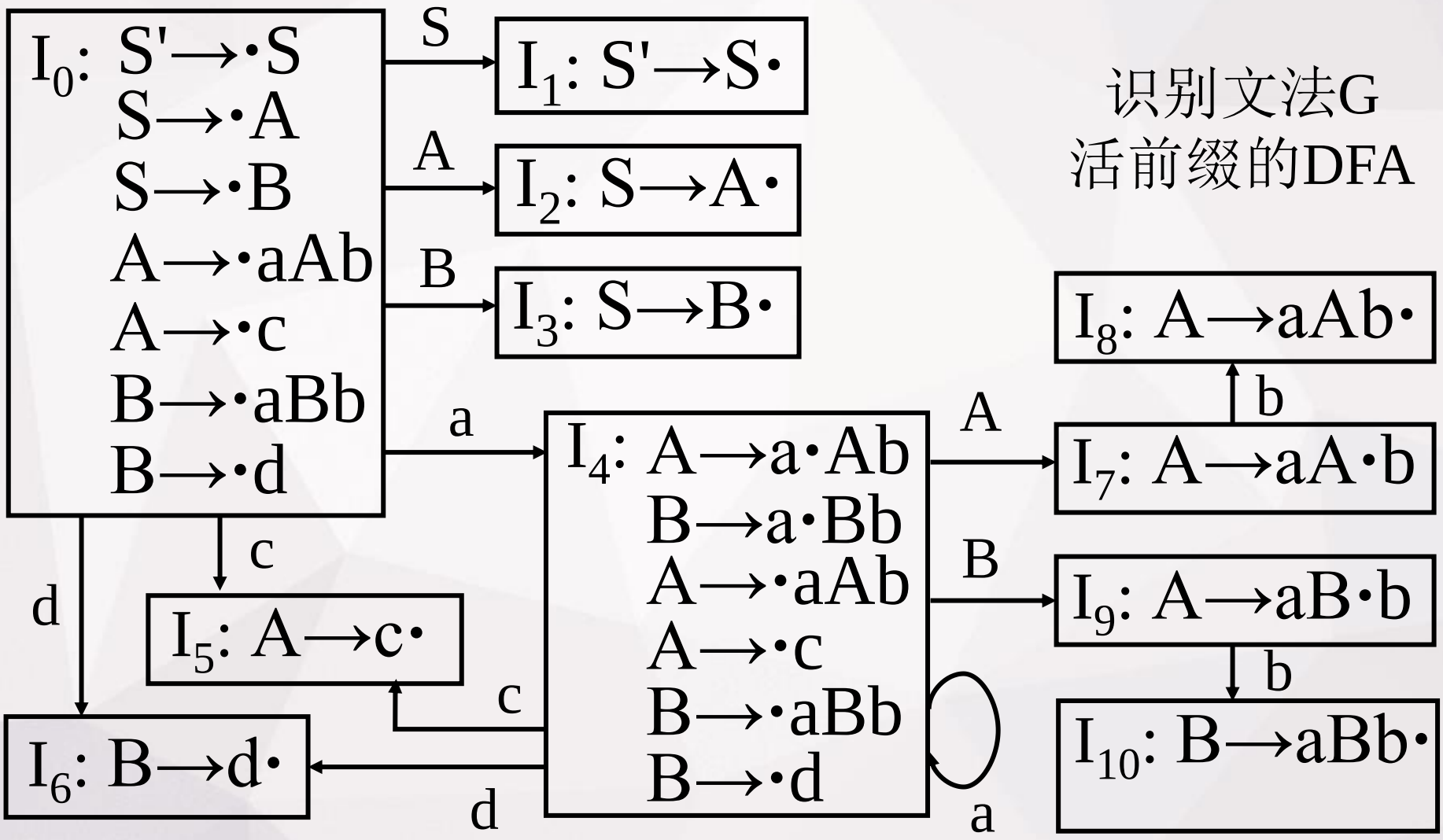
4.5.2 LR(0)分析法



识别文法G
活前缀的DFA

0. S'→S
1. S→A
2. S→B
3. A→aAb
4. A→c
5. B→aBb
6. B→d

4.5.2 LR(0) 分析法


I0: S'→·S
S→·A
S→·B
A→·aAb
A→·c
B→·aBb
B→·d
S
I1: S'→S·
A
I2: S→A·
B
I3: S→B·
a
I4: A→a·Ab
B→a·Bb
A→·aAb
A→·c
B→·aBb
B→·d
c
I5: A→c·
d
I6: B→d·
c
d
A
I7: A→aA·b
b
I8: A→aAb·
B
I9: A→aB·b
b
I10: B→aBb·
a
识别文法G
活前缀的DFA

4.5.2 LR(0)分析法

构成识别一个文法活前缀的DFA的状态（项目集）的全体称为这个文法的LR(0)**项目集规范族**。

若一个文法G的拓广文法G'的LR(0)项目集规范族中的每个项目集不存在移进项目和归约项目同时并存或多个归约项目同时并存，则称G为LR(0)文法。

(4) LR(0)分析表的构造

输入：识别LR(0)文法G规范句型活前缀的DFA

输出：文法G的LR(0)分析表

方法：用整数 0, 1, 2, …，n 分别表示状态 $I_0, I_1, I_2 \ldots, I_n$，令包含项目S'→•S 的集合 I_k 的下标为分析器的初始状态。

4.5.2 LR(0)分析法

1.若项目A→α•xβ属于 I_k, 且转换函数GO(I_k, x)=I_i，当 x 为终结符时，则置ACTION[*k*, x]=s_i。 见图 见表

2.若GO(I_k, x)=I_j, x为非终符，则置 GOTO[*k*, x]=*j*。 见图 见表

3.若项目A→α•属于I_k，则对任何终结符和结束符$（统一记为a）则置ACTION[*k*, a]=$r_j$（假定A→α为文法的第*j*条规则）

0. S'→S **1.** S→A **2.** S→B **3.** A→aAb
4. A→c **5.** B→aBb **6.** B→d 见图 见表

4. 若项目S'→S •属于I_k，则置ACTION[*k*, $]=acc。 见图 见表

5. 分析表中凡不能用规则1至4填入信息的元素均置为“出错标志”，为了分析表的清晰，仅用空白表示出错标志。 见表

4.5.2 LR(0)分析法

例2　考虑文法G[S]:S→(S) | a

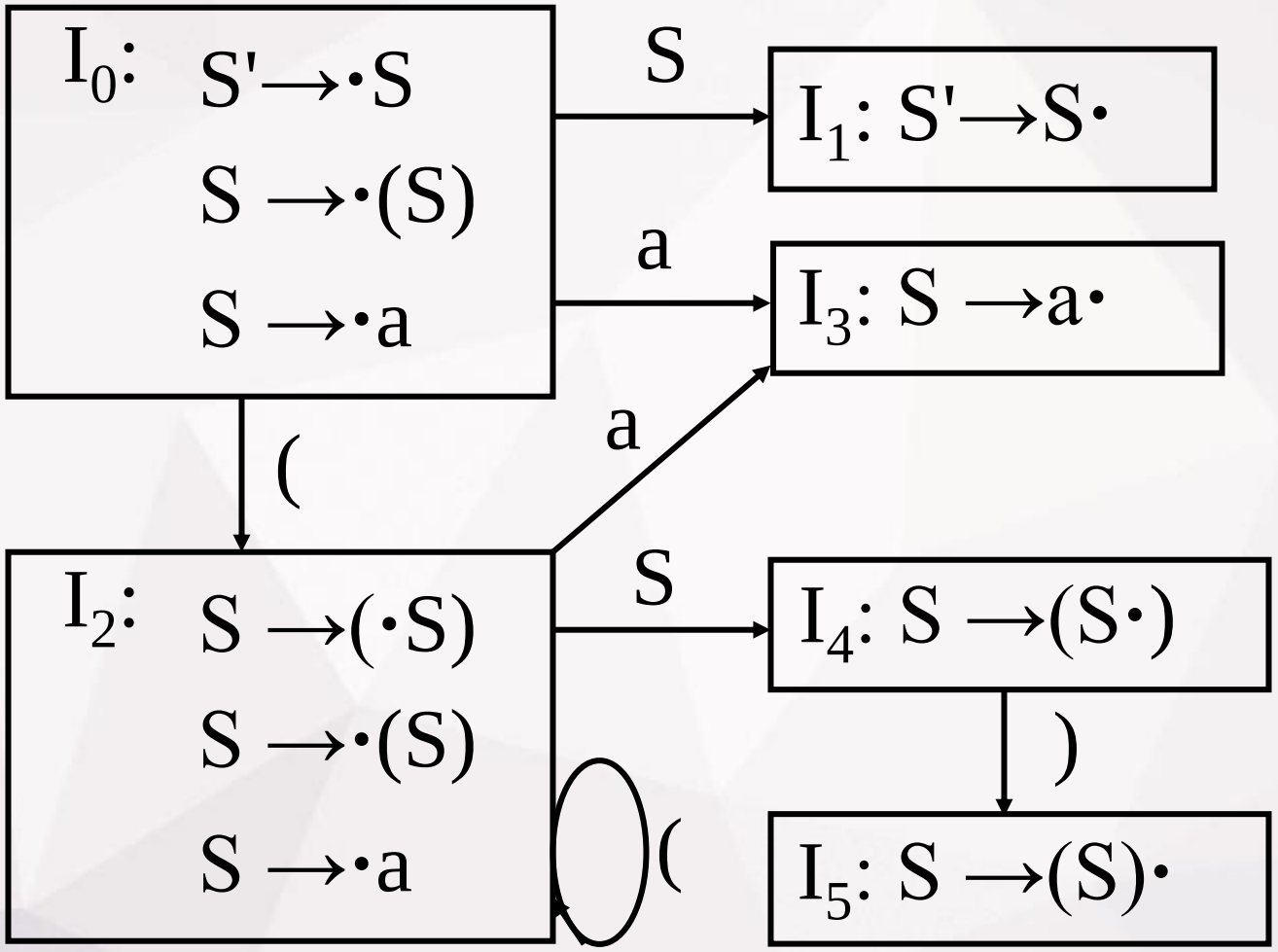
(1) 构造识别文法规范句型活前缀的DFA

(2) 判断该文法是否LR(0)文法，若是，请构造LR(0)分析表，若不是，请说明理由。

首先将文法拓广，并给每条规则编号

0. S’→S　1. S →(S)　2. S →a

4.5.2 LR(0) 分析法



0. S'→S
1. S →(S)
2. S →a

识别活前缀的DFA

4.5.2 LR(0) 分析法

文法G[S]的LR(0)分析表

状态	ACTION				GOTO
	a	(	)	$	S
0	s_3	s_2			1
1				acc	
2	s_3	s_2			4
3	r_2	r_2	r_2	r_2	
4			s_5		
5	r_1	r_1	r_1	r_1	

0. S'→S

1. S →(S)

2. S →a

4.5.2 LR(0)分析法

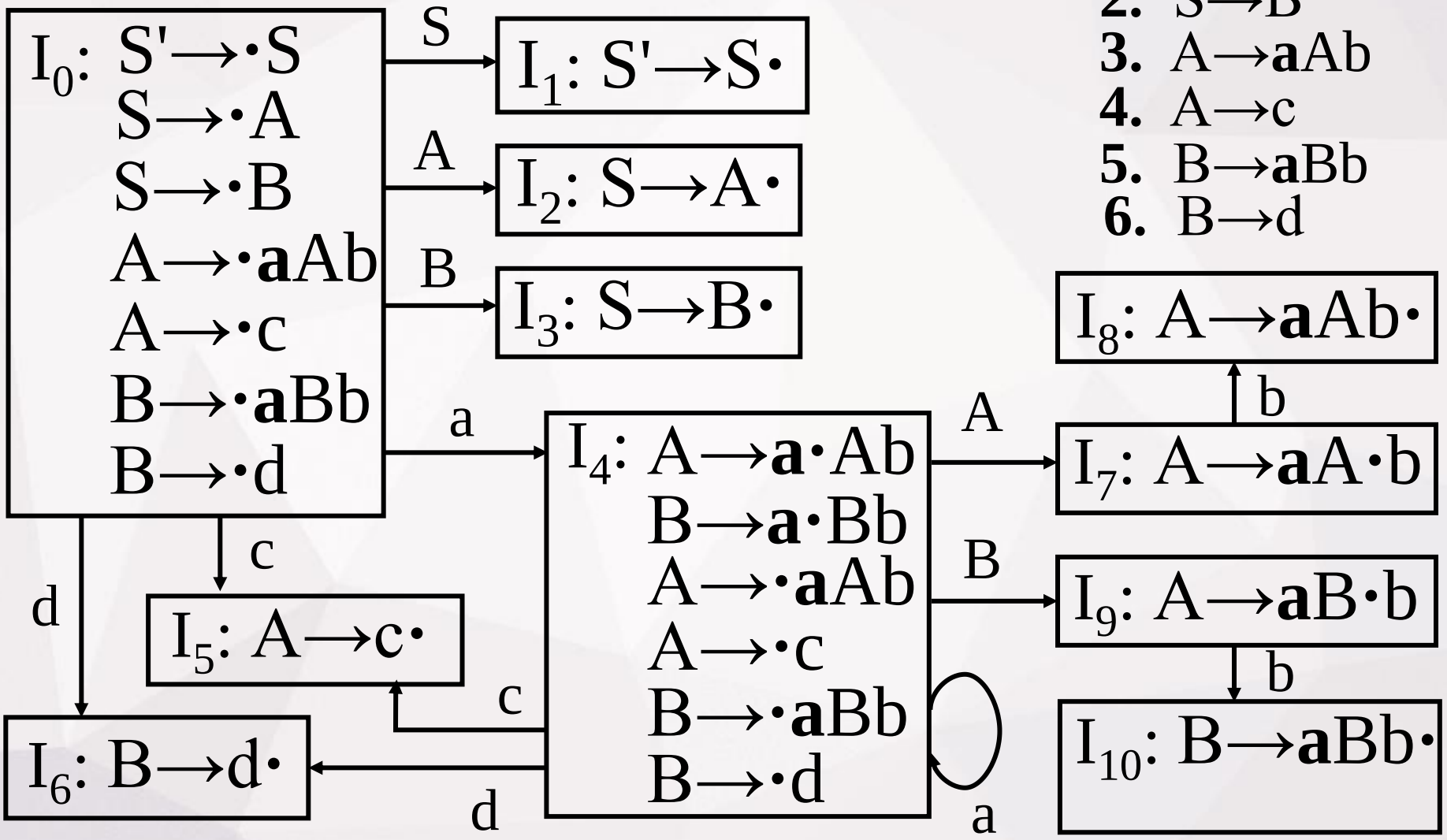
由上述构造过程可以看出，LR(0)分析器的特点是不向前查看输入符号就归约，即当栈顶状态包括归约项目，不管下一个输入符号是什么，都认为形成句柄立即进行归约。

若分析串不是文法的句子？

下一页

识别文法G活前缀的DFA

0. S'→S
1. S→A
2. S→B
3. A→**a**Ab
4. A→c
5. B→**a**Bb
6. B→d



返回

文法G[S]的LR(0)分析表

状态	ACTION					GOTO		
	a	b	c	d	$	S	A	B
0	s_4		s_5	s_6		1	2	3
1					acc			
2	r_1	r_1	r_1	r_1	r_1			
3	r_2	r_2	r_2	r_2	r_2			
4	s_4		s_5	s_6			7	9
5	r_4	r_4	r_4	r_4	r_4			

0. S'→S　**1.** S→A　**2.** S→B　**3.** A→**a**Ab

4. A→c　**5.** B→**a**Bb　**6.** B→d

返回

文法G[S]的LR(0)分析表

状态	ACTION					GOTO		
	a	b	c	d	$	S	A	B
0	s_4		s_5	s_6		1	2	3
1					acc			
2	r_1	r_1	r_1	r_1	r_1			
3	r_2	r_2	r_2	r_2	r_1			
4	s_4		s_5	s_6			7	9
5	r_4	r_4	r_4	r_4	r_4			
6	r_6	r_6	r_6	r_6	r_6			
7		s_8						
8	r_3	r_3	r_3	r_3	r_3			
9		s_{10}						
10	r_5	r_5	r_5	r_5	r_5			

返回

4.5.3 SLR(1)分析法

例 算术表达式的文法

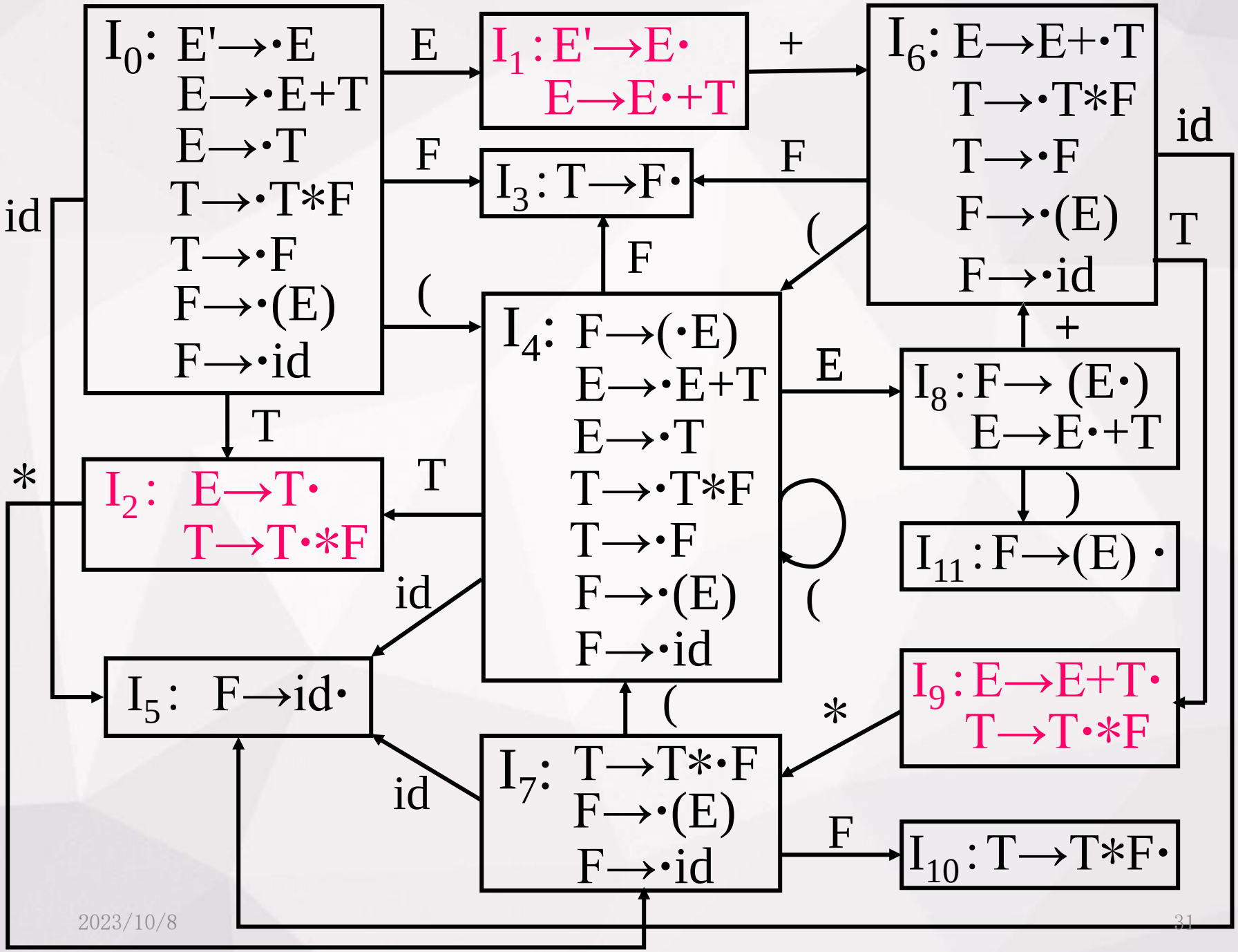
G[E]:E→E+T | T， T→T*F | F，F→(E) | id

将文法拓广并对规则进行编号

0. E’→E　　1. E→E+T　　2. E→T

3. T→T * F　　4. T→F　　5. F→(E)

6. F→id


I0: E'→·E
E→·E+T
E→·T
T→·T*F
T→·F
F→·(E)
F→·id
I1: E'→E·
E→E·+T
I2: E→T·
T→T·*F
I3: T→F·
I4: F→(·E)
E→·E+T
E→·T
T→·T*F
T→·F
F→·(E)
F→·id
I5: F→id·
I6: E→E+·T
T→·T*F
T→·F
F→·(E)
F→·id
I7: T→T*·F
F→·(E)
F→·id
I8: F→(E·)
E→E·+T
I9: E→E+T·
T→T·*F
I10: T→T*F·
I11: F→(E)·
E
+
F
(
T
id
*
)
识别表达式文法活前缀的DFA

表达式文法的LR(0)分析表

状态	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	s_5			s_4			1	2	3
1		s_6				acc			
2	r_2	r_2	s_7r_2	r_2	r_2	r_2			
3	r_4	r_4	r_4	r_4	r_4	r_4			
4	s_5			s_4			8	2	3
5	r_6	r_6	r_6	r_6	r_6	r_6			
6	s_5			s_4				9	3
7	s_5			s_4					10
8		s_6			s_{11}				
9	r_1	r_1	s_7r_1	r_1	r_1	r_1			
10	r_3	r_3	r_3	r_3	r_3	r_3			
11	r_5	r_5	r_5	r_5	r_5	r_5			

4.5.3 SLR(1)分析法

I_K={ X→δ·bB, A→α·, B→γ· }

若满足：FOLLOW(A)∩FOLLOW(B)= ∅

FOLLOW(A)∩{b}= ∅ FOLLOW(B)∩{b}= ∅

那么，当状态K面临输入符号a时：

(1) 若a=b则移进

(2) 若a∈FOLLOW(A)，则用规则A→α进行归约。

(3) 若a∈FOLLOW(B)，则用规则B→γ进行归约。

(4) 此外报错。

4.5.3 SLR(1)分析法

一般而言，若一个LR(0)项目集I中有m个移进项目和n个归约项目时：

$$I:\{A_1\to\alpha_1\cdot a_1\beta_1, A_2\to\alpha_2\cdot a_2\beta_2, \ldots, A_m\to\alpha_m\cdot a_m\beta_m,$$
$$B_1\to\gamma_1\cdot, B_2\to\gamma_2\cdot, \ldots, B_n\to\gamma_n\cdot\}$$

若移进符号集$\{a_1,a_2,\ldots,a_m\}$和$FOLLOW(B_1)$，$FOLLOW(B_2)$，…，$FOLLOW(B_n)$两两相交为空时，则项目集I中的冲突仍可用下述规则解决。

设a为当前输入符号：

(1)若$a\in\{a_1,a_2,\ldots,a_m\}$则移进。

(2) 若$a\in FOLLOW(B_i)$，$i=1,\ldots,n$，则用$B_i\to\gamma_i$进行归约。

(3) 此外报错。

这种用来解决分析动作冲突的方法称为**SLR(1)方法**。

4.5.3 SLR(1) 分析法

如果对于一个文法的某些LR(0)项目集或LR(0)分析表中所含有的动作冲突都能用SLR(1)方法解决，则称这个文法是SLR(1)文法。

现在分别考察上例中的项目集I_2，I_9

I_2={ E→T·, T→T·*F }

I_9={ E→E+T·, T→T·*F }

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

FOLLOW(E) ∩ {*}= ø

4.5.3 SLR(1)分析法

SLR(1)分析表的构造与LR(0)分析表的构造基本相同。仅对LR(0)分析表构造算法中的规则2进行如下修改：

若归约项目$A\rightarrow\alpha\bullet$属于$I_k$，则对任何终结符 $a \in FOLLOW(A)$置 $ACTION[k, a]=r_j$，其中$A\rightarrow\alpha$为文法的第j条规则。

若文法的SLR(1)分析表不含多重定义元素，则称文法G为SLR(1)文法。

4.5.3 SLR(1) 分析法

对算术表达式文法构造SLR(1)分析表：

0 E’→E

1 E→E+T

2 E→T

3 T→T * F

4 T→F

5 F→(E)

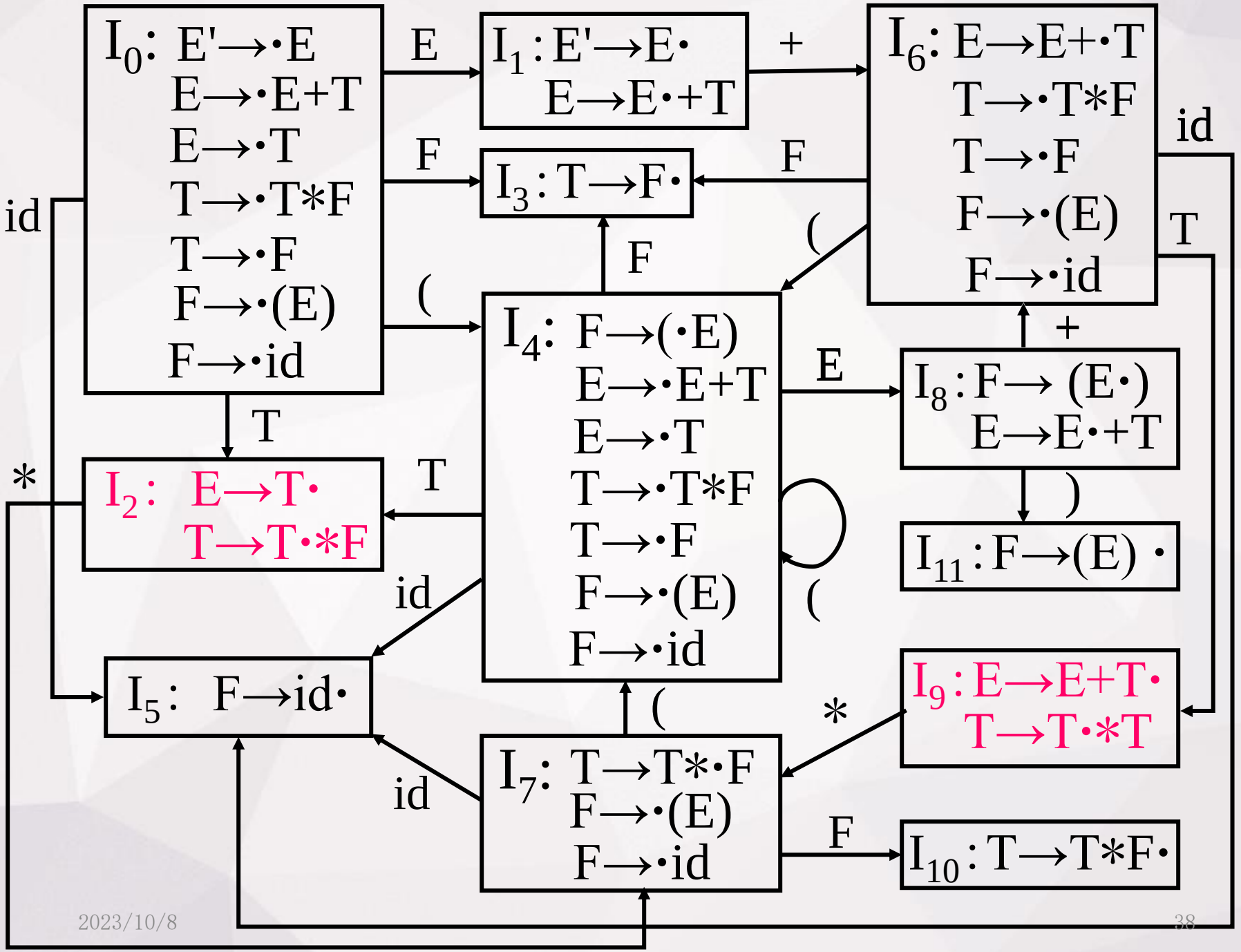
6 F→id

FOLLOW(E')={$}

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

FOLLOW(T)=FOLLOW(F)

={+,), *, $}


I0: E'→·E
E→·E+T
E→·T
T→·T*F
T→·F
F→·(E)
F→·id
I1: E'→E·
E→E·+T
I2: E→T·
T→T·*F
I3: T→F·
I4: F→(·E)
E→·E+T
E→·T
T→·T*F
T→·F
F→·(E)
F→·id
I5: F→id·
I6: E→E+·T
T→·T*F
T→·F
F→·(E)
F→·id
I7: T→T*·F
F→·(E)
F→·id
I8: F→(E·)
E→E·+T
I9: E→E+T·
T→T·*T
I10: T→T*F·
I11: F→(E)·
E
F
+
id
T
(
)
*
识别表达式文法活前缀的DFA

表达式文法的SLR(1)分析表

状态	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	s_5			s_4			1	2	3
1		s_6				acc			
2		r_2	s_7		r_2	r_2			
3		r_4	r_4		r_4	r_4			
4	s_5			s_4			8	2	3
5		r_6	r_6		r_6	r_6			
6	s_5			s_4				9	3
7	s_5			s_4					10
8		s_6			s_{11}				
9		r_1	s_7		r_1	r_1			
10		r_3	r_3		r_3	r_3			
11		r_5	r_5		r_5	r_5			

4.5.3 SLR(1)分析法

例1 设有拓广文法G[S′]

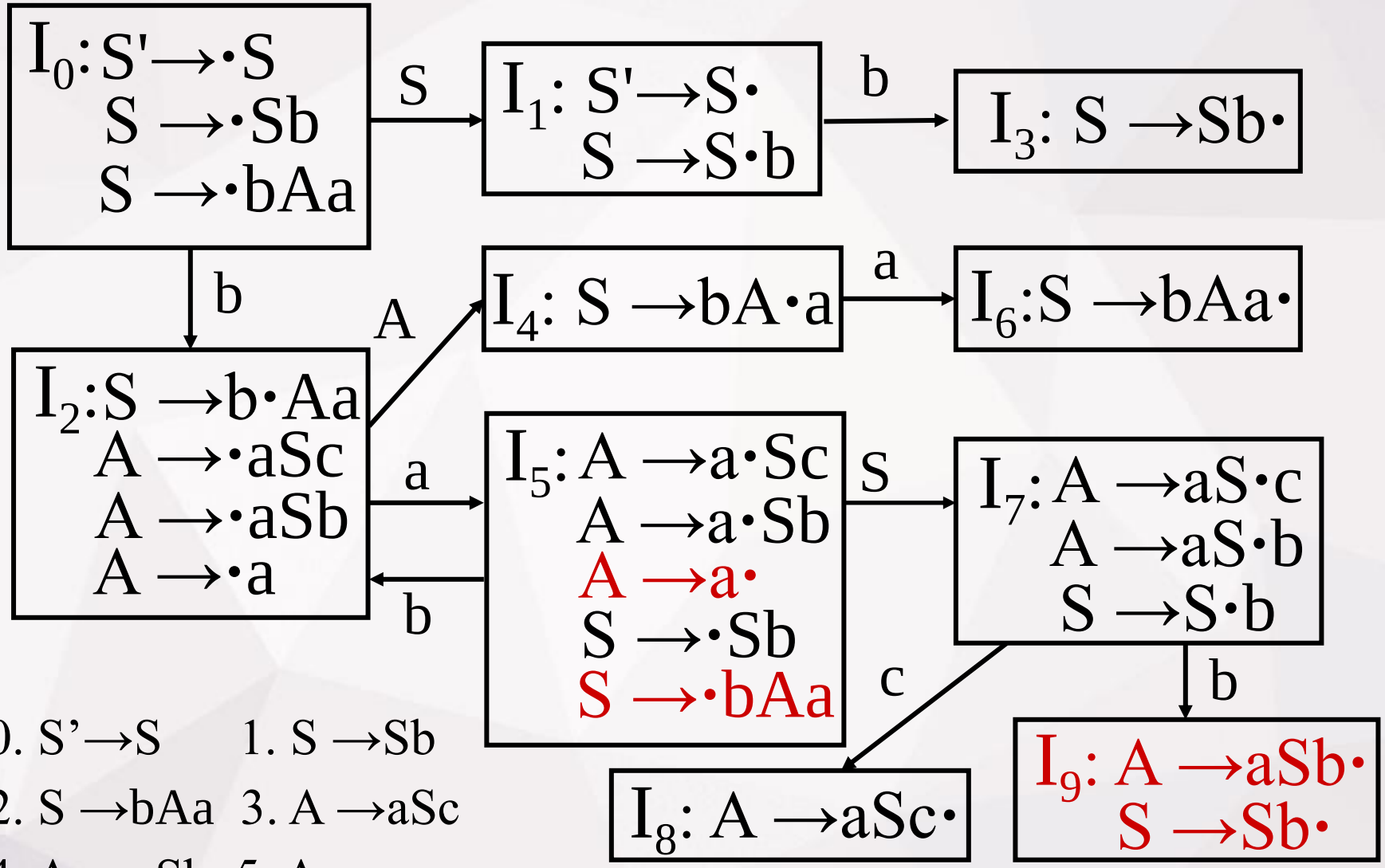
0. S’→S　　1. S →Sb　　2. S →bAa

3. A →aSc　　4. A →aSb　　5. A →a

(1)构造识别文法规范句型活前缀的DFA。

(2) 判断该文法是否SLR(1)文法，若是，构造SLR(1)分析表，若不是，请说明理由。

文法G[S']LR(0)项目集及转换函数



0. S'→S　1. S →Sb

2. S →bAa　3. A →aSc

4. A →aSb　5. A →a

4.5.3 SLR(1)分析法

分析所有这些项目集，可知在项目集I_5中存在移进—归约冲突，I_9中存在归约—归约冲突，因此该文法不是LR(0)文法。

考[illegible]项[illegible]I[illegible]决。

FOLLOW(S)={b,c,$}

FOLLOW(S')={$}

FOLLOW(A)={a}

S'→S

1. S →Sb
2. S →bAa
3. A →aSc
4. A →aSb
5. A →a

4.5.3 SLR(1) 分析法

I_5={ A →a·, S →·bAa }

FOLLOW(A) ∩ {b}= {a} ∩ {b}= ø

I_9={ A →aSb·, S →Sb· }

FOLLOW(A) ∩ FOLLOW(S)= {a} ∩ {b, c, $}= ø

4.5.3 SLR(1) 分析法

状态	ACTION				GOTO	
	a	b	c	$	S	A
0		s_2			1	
1		s_3		acc		
2	s_5					4
3		r_1	r_1	r_1		
4	s_6					
5	r_5	s_2			7	
6		r_2	r_2	r_2		
7		s_9	s_8			
8	r_3					
9	r_4	r_1	r_1	r_1		

0. S’→S
1. S →Sb
2. S →bAa
3. A →aSc
4. A →aSb
5. A →a

G[S]的SLR(1)分析表

4.5.3 SLR(1)分析法

SLR(1)分析法是一种简单而实用的方法,构造分析表的算法简单,状态数目少,且大多数程序设计语言都可以用SLR(1)文法来定义。

但是仍存在一些文法，其项目集中的移进-归约冲突或归约-归约冲突不能使用SLR(1)方法解决。

4.5.3 SLR(1)分析法

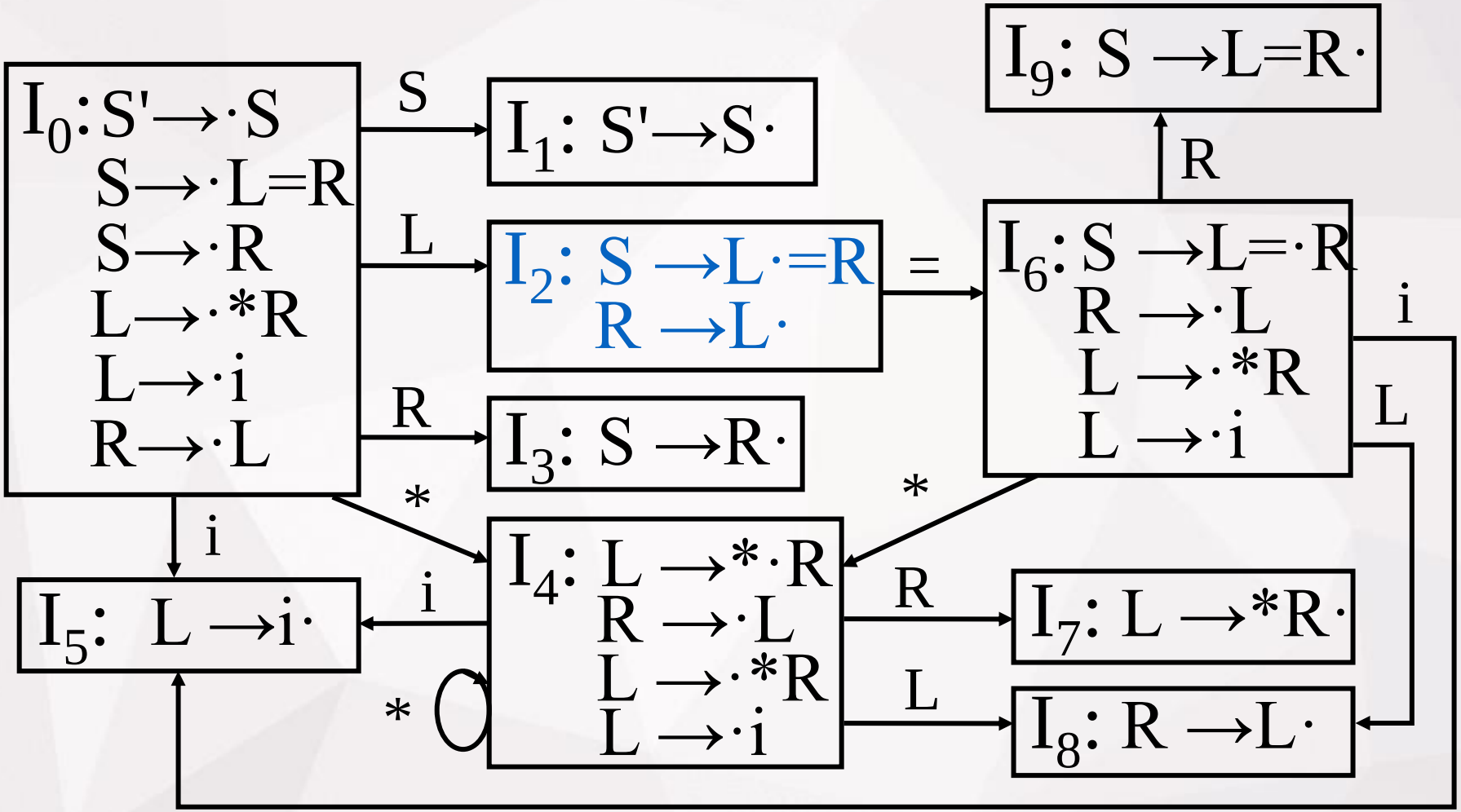
例2 下列拓广文法G′为

0. S’→S　　1. S →L=R　　2. S →R

3. L →*R　　4. L →i　　5. R →L

我们首先用S'→•S作为初态集的项目，然后用闭包函数和转换函数构造识别文法G'活前缀的DFA如下图所示。

LR(0)识别G'活前缀的DFA



0. S'→S　　1. S →L=R　　2. S →R

3. L →*R　　4. L →i　　5. R →L

4.5.3 SLR(1) 分析法

可以发现项目集 I_2 中存在移进和归约冲突：

$$I_2 = \begin{Bmatrix} S \to L \cdot = R \\ R \to L \cdot \end{Bmatrix}$$

$S \Rightarrow L=R \Rightarrow *R=R$

$S \Rightarrow R$

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

$= \{=,\$\} \cap \{=\} \neq \varnothing$

0. S’→S
1. S →L=R
2. S →R
3. L →*R
4. L →i
5. R →L

1. SLR(1)方法为什么不能解决 I_2 中的移进和归约冲突?
2. 怎样解决 I_2 中的移进和归约冲突?

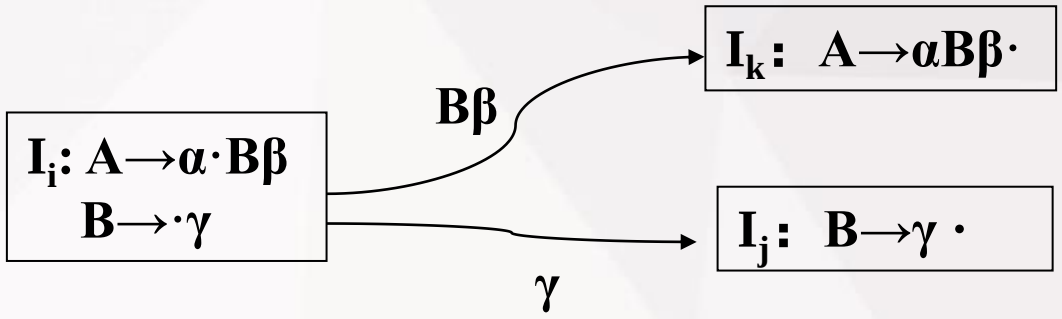
4.5.3 SLR(1)分析法

一般情况：

若：$A\rightarrow\alpha\cdot B\beta\in I_i$

则：$B\rightarrow\cdot\gamma\in I_i$

对应DFA：

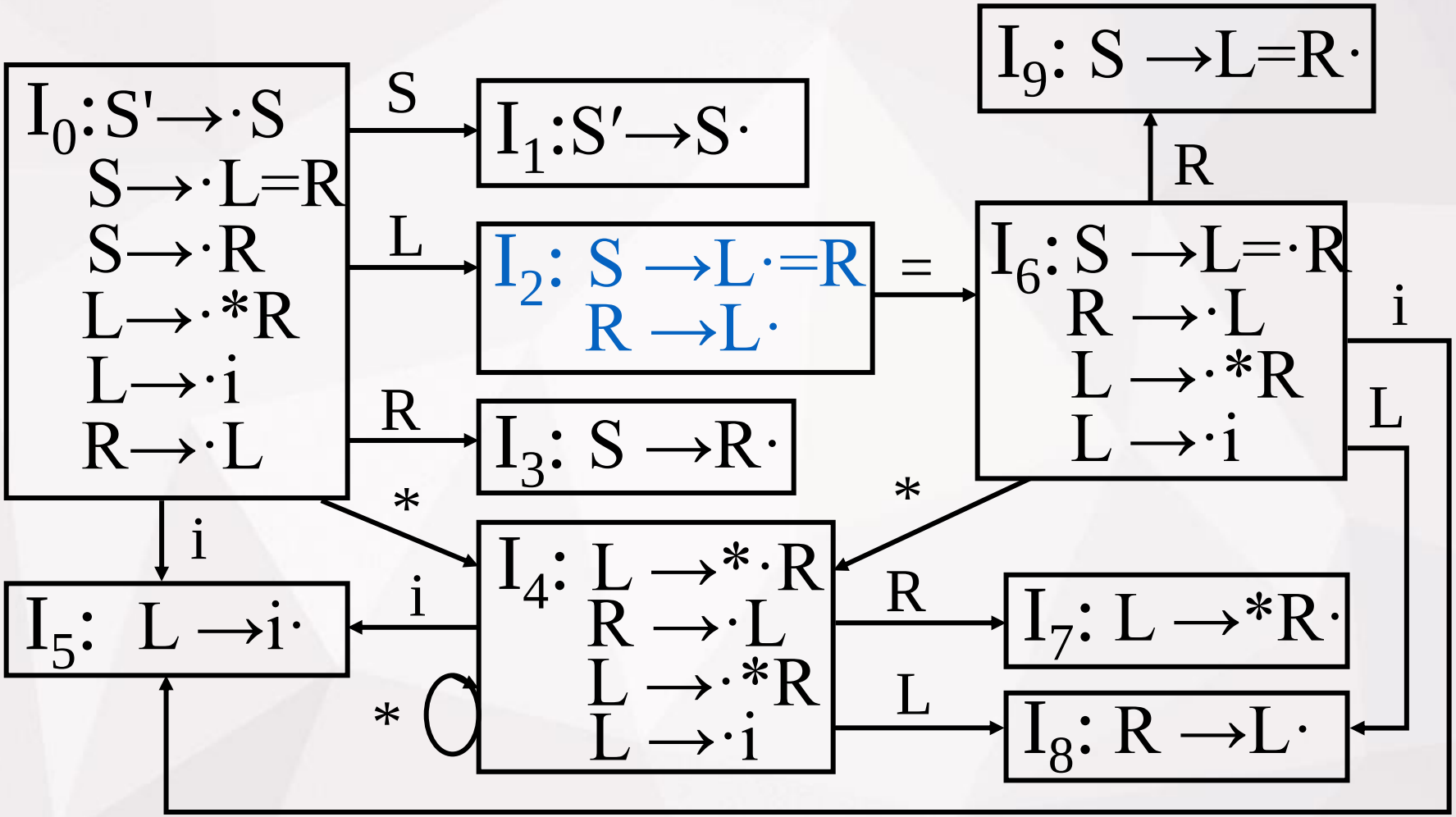


状态I_j和状态I_k的关系，有归约序列：

$a_1 \ldots\ldots a_n \overset{*}{\Rightarrow} \delta\ \alpha\ \gamma\ a_p \ldots\ldots a_n \overset{*}{\Rightarrow} \delta\ \alpha\ B\ a_p \ldots\ldots a_n \overset{*}{\Rightarrow} \delta\ \alpha\ B\ \beta\ a_q \ldots\ldots a_n$

在状态I_j处SLR（1）直接按搜索符是否属于B的FOLLOW（B）集合来确定是否归约，而I_i的项目$A\rightarrow\alpha\cdot B\beta$表示要接收Bβ到达状态$I_k$，这样当输入栈当前符$a_p$是属于FOLLOW（B）但不属于FIRST（β）时，就会在状态I_j进行一次错误的归约。

LR(0)识别G'活前缀的DFA



0. S'→S　　1. S →L=R　　2. S →R

3. L →*R　　4. L →i　　5. R →L

4.5.4 LR（1）分析法

LR(1)项目 [A→α•β，a]

其中A→α•β是一个LR(0)项目，每个a是终结符，称为展望符或搜索符。

当β≠ε时，搜索符是无意义的。

当β = ε时，搜索符a明确指出当[A→α•β，a]是栈顶状态的一个LR(1)项目时，仅在输入符号是a时才能用A→α归约。

4.5.4 LR（1）分析法

构造LR(1)项目集族的方法和构造LR(0)项目集规范族的方法基本相同。

(1) 构造LR(1)项目集 I 的闭包函数

(a) I的任何LR(1)项目都属于CLOSURE(I)

(b) 若项目 $[A\rightarrow\alpha\cdot B\beta, a]$ 属于CLOSURE(I), $B\rightarrow\gamma$ 是文法中的一条规则， **$b\in FIRST(\beta a)$，则$[B\rightarrow\cdot\gamma, b]$也属于CLOSURE(I)**

(c) 重复(b),直到CLOSURE(I)不再增大为止

4.5.4 LR（1）分析法

对例2中的文法G', 令 I=[S'→•S，$]为初态集的初始项目集，对其求闭包：

0. S'→S　1. S→L=R　2. S→R

3. L→*R　4. L→i　5. R→L

CLOSURE(I)

= CLOSURE({[S'→•S，$]})

= {[S'→•S，$] [S→•L=R，$] [S→•R，$]

[L→•*R，=/$] [L→•i，=/$]

[R→•L，$]}

= I_0

FIRST(βa)=FIRST($)={$}

FIRST(βa)=FIRST(=R$)={=}

FIRST(βa)=FIRST($)={$}

FIRST(βa)=FIRST($)={$}

4.5.4 LR（1）分析法

(2) 构造转换函数

令I是一个LR(1)项目集，X是一个文法符号，函数 GO(I，X)=CLOSURE(J)。

$$J=\{[A\rightarrow\alpha X\cdot\beta,a]\mid[A\rightarrow\alpha\cdot X\beta,a]\in I\}$$

4.5.4 LR（1）分析法

I_0 = {[S'→•S，$] [S→•L=R，$]

[S→•R，$] [L→•*R，=/$]

[L→•i，=/$] [R→•L，$] }

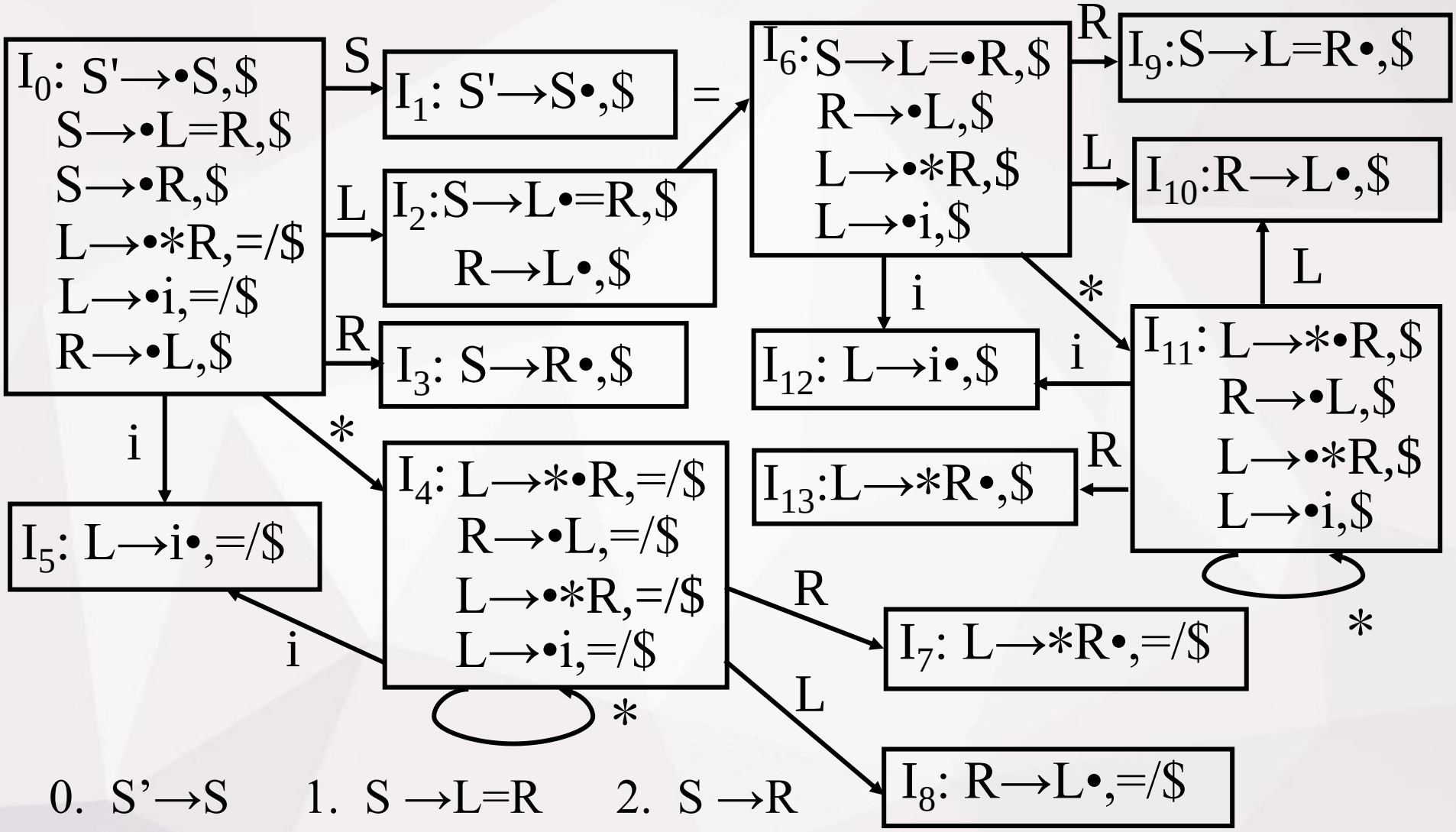
GO(I_0 ,S) = { [S'→S·，$] }= I_1

GO(I_0 ,*) = { [L→*·R，=/$] [R→·L，=/$]

[L→·*R，=/$] [L→·i，=/$] }

= I_4

LR(1)项目集族及转换函数



4.5.4 LR（1）分析法

构造LR(1)分析表的方法与构造LR(0)分析表的方法基本相同，仅对归约项目作如下修改：

若归约项目[A→α·,a]属于I_k，则对搜索符a置ACTION[K，a]＝r_j。其中A→α为文法的第j条规则。

状态	ACTION				GOTO		
	i	*	=	$	S	L	R
0	s_5	s_4			1	2	3
1				acc			
2			s_6	r_5			
3				r_2			
4	s_5	s_4				8	7
5			r_4	r_4			
6	s_{12}	s_{11}				10	9
7			r_3	r_3			
8			r_5	r_5			
9				r_1			
10				r_5			
11	s_{12}	s_{11}				10	13
12				r_4			
13				r_3			

0. S’→S
1. S →L=R
2. S →R
3. L →*R
4. L →i
5. R →L

G[S]的LR(1)分析表

4.5.4 LR（1）分析法

如果一个文法的LR(1)分析表不含多重入口时，或者任何一个LR(1)项目集中没有移进一归约冲突或归约一归约冲突，则称该文法为LR(1)文法。

例2中的文法是一个LR(1)文法，却不是SLR(1)文法。然而，当一个文法是LR(0)文法，则一定也是一个SLR(1)文法，也是LR(1)文法。反之则不一定成立。

4.5.4 LR（1）分析法

例3 考虑拓广文法:

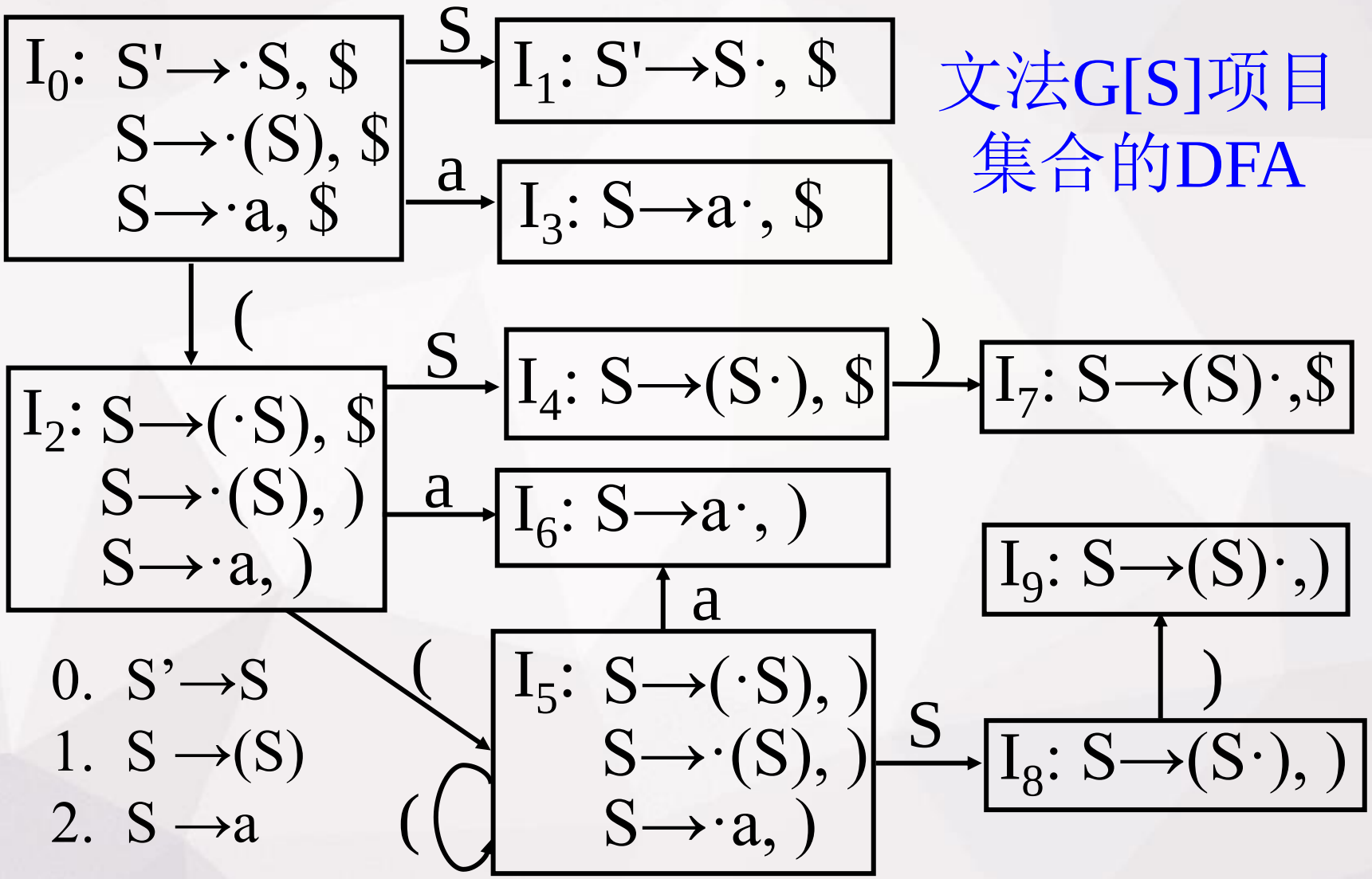
0. S’→S

1. S →(S)

2. S →a

试构造它的LR(1)项目集合的DFA和LR(1)分析表。

4.5.4 LR(1) 分析法



文法G[S]项目集合的DFA

4.5.4 LR（1）分析法

状态	ACTION			GOTO
	a	（	） $	S
0	s_3	s_2		1
1			acc	
2	s_6	s_5		4
3			r_2	
4		s_7		
5	s_6	s_5		8
6		r_2		
7			r_1	
8		s_9		
9		r_1		

0. S’→S
1. S →(S)
2. S →a

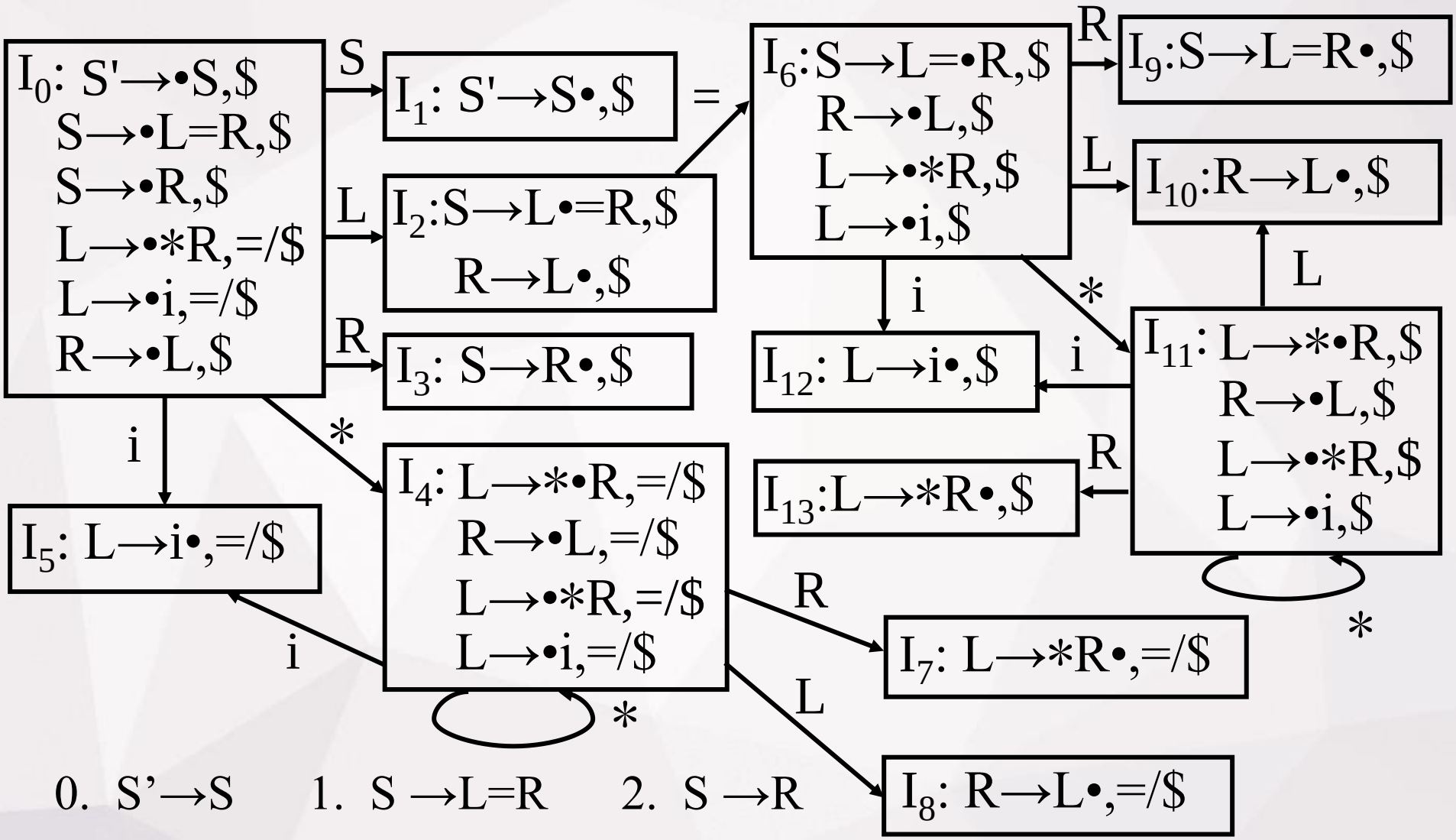
LR(1)分析表

4.5.4 LALR（1）分析法

LALR(1)分析法的基本思想是对LR(1)项目集规范族中将**所有同心的项目集合并为一**，以减少项目集个数。

所谓同心的LR(1)项目集是指在**两个LR(1)项目集中，除搜索符不同之外，核心部分是相同的。**

LR(1)项目集族及转换函数



4.5.5 LALR（1）分析法

I_4与I_{11}，I_5与I_{12}，I_7与I_{13}，I_8与I_{10}它们两两之间除了搜索符不同之外,“心”是相同的。将**同心集**合并为：

$I_{4,11}$: L→*•R, =/$
R→•L, =/$
L→•*R, =/$
L→•i, =/$

$I_{5,12}$: L→i•, =/$

$I_{7,13}$: L→*R•, =/$

$I_{8,10}$: R→L•, =/$

4.5.5 LALR（1）分析法

假定LR(1)文法的项目集I_k与I_j为同心集，其中

I_k={ [A→α•, a_1] ， [B→β•aγ, b_1] }

I_j={ [A→α•, a_2] ， [B→β•aγ, b_2] }

I_{kj}={ [A→α•, a_1/a_2] [B→β•aγ, b_1/b_2] }

I_k={ [A_1→α_1•, a_1] ， [A_2→α_2•, a_2] ， [B→β•aγ, b_1]}

I_j={ [A_1→α_1•, a_2] ， [A_2→α_2•, a_1] ， [B→β•aγ, b_2]}

I_{kj}={ [A_1→α_1•, a_1/a_2] ， [A_2→α_2•, a_1/a_2] ， [B→β•aγ, b_1/b_2] }

合并同心集后可能带来归约-归约冲突，但不可能是移进-归约冲突。

4.5.5 LALR（1）分析法

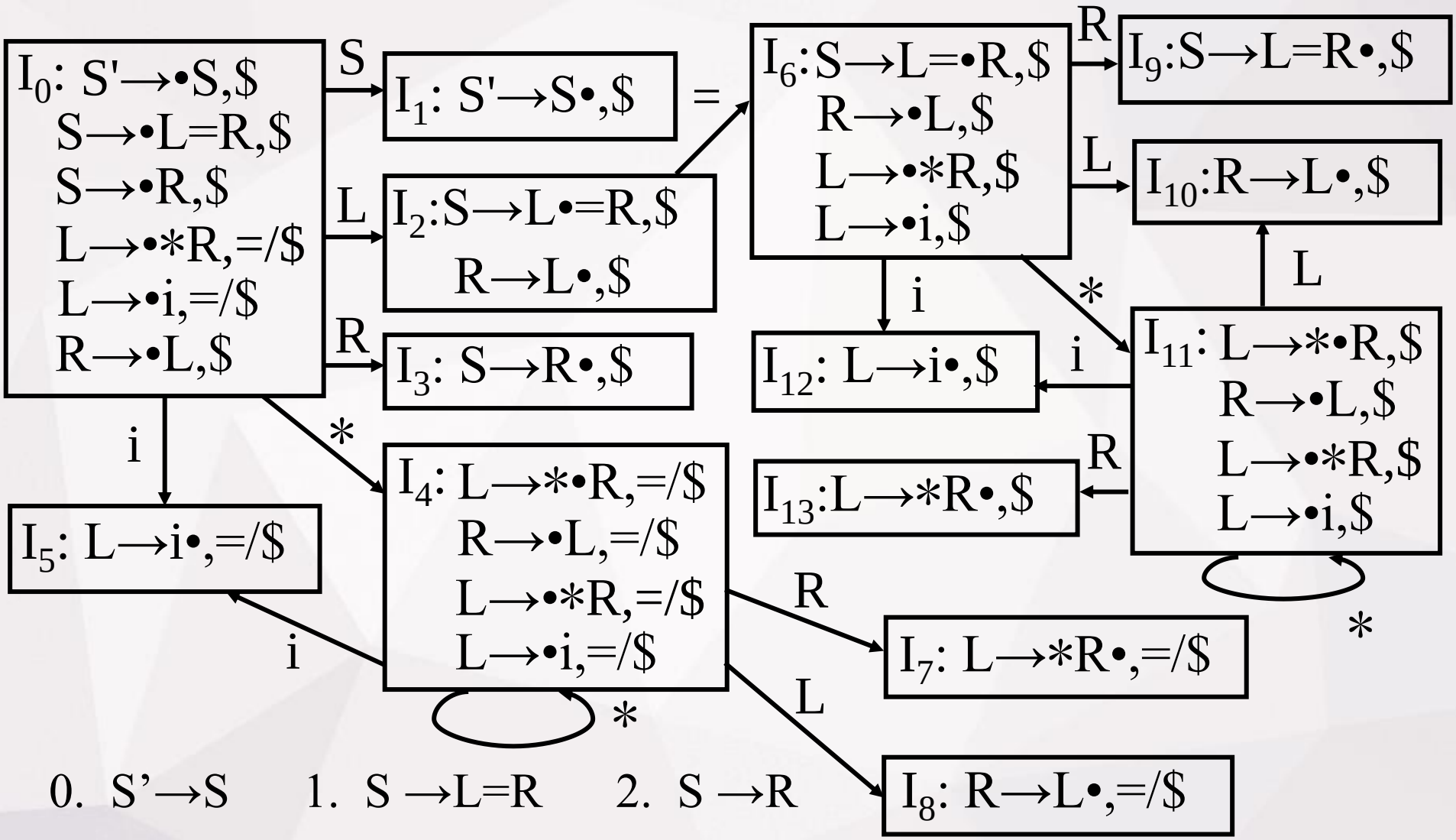
我们看到**合并同心集后的项目集其核心部分不变，仅搜索符合并**。对合并同心集后的项目集的**转换函数为GO(I,X)自身的合并**，这是因为相同的心之转换函数仍属同心集。

例如 $GO(I_{4,11}, i)=GO(I_4, i)\cup GO(I_{11}, i)= I_{5,12}$

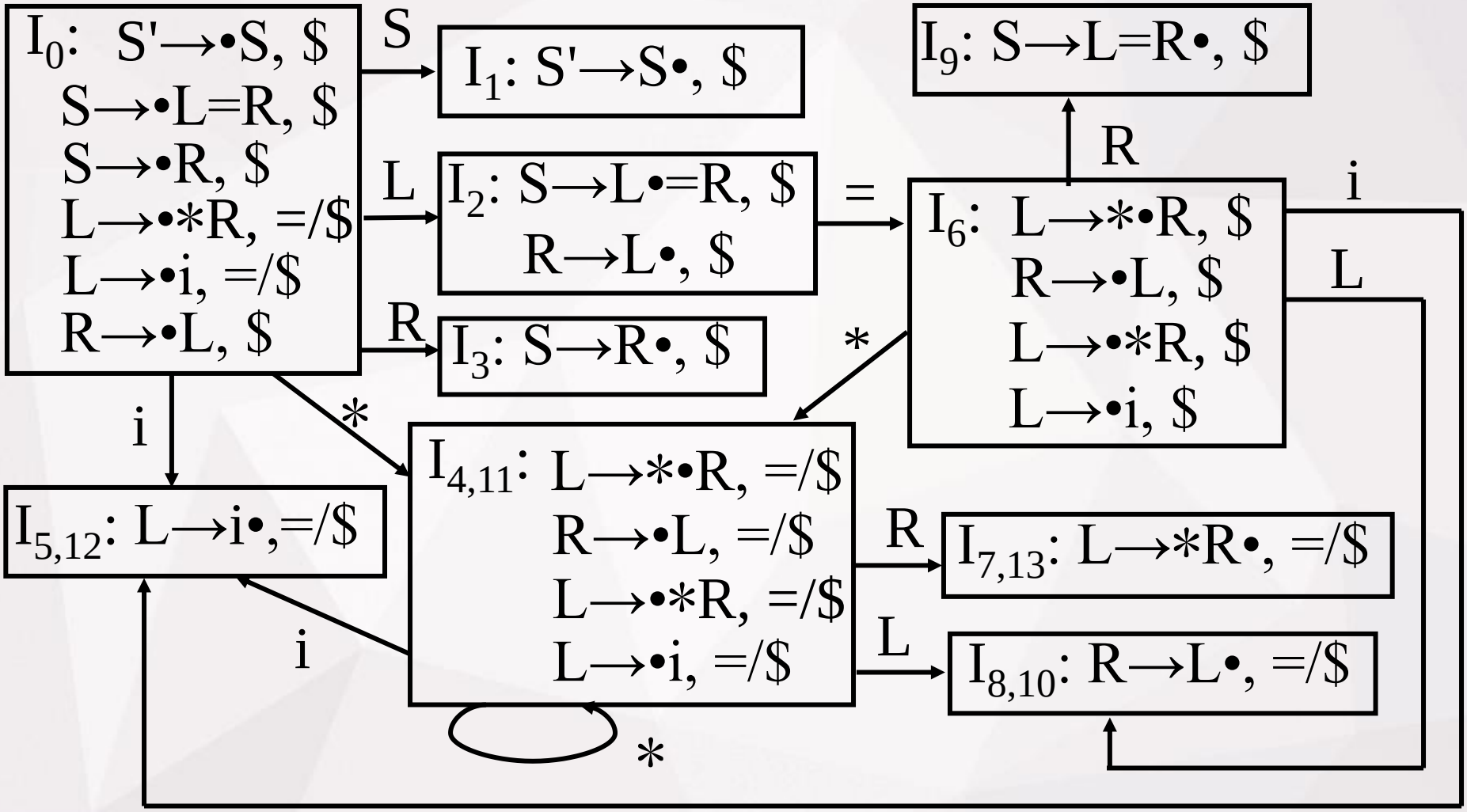
$GO(I_{4,11}, R)=GO(I_4, R)\cup GO(I_{11}, R)= I_{7,13}$

$GO(I_{4,11}, *)=GO(I_4, *)\cup GO(I_{11}, *)= I_{4,11}$

LR(1)项目集族及转换函数



LALR(1)项目集族及转换函数



0. S'→S　1. S →L=R　2. S →R

3. L →*R　4. L →i　5. R →L

4.5.5 LALR（1）分析法

根据合并同心集后的项目集族构造文法的LALR(1)分析表的方法：

1. 构造拓广文法G′的LR(1)项目集族。

2. 若LR(1)项目集族中不存在含冲突的项目集，则**合并所有同心集，构造出文法的LALR(1)项目集族。**

3. 若LALR(1)项目集族中不存在归约-归约冲突，则该文法是LALR(1)文法。

4. LALR(1)项目集族构造LALR(1)分析表的方法与LR(1)分析表的构造方法相同。

G[S]的LALR(1)分析表

状态	ACTION				GOTO		
	i	*	=	$	S	L	R
0	$S_{5,12}$	$S_{4,11}$			1	2	3
1				acc			
2			s_6	r_5			
3				r_2			
4,11	$S_{5,12}$	$S_{4,11}$				8,10	7,13
5,12			r_4	r_4			
6	$S_{5,12}$	$S_{4,11}$				8,10	9
7,13			r_3	r_3			
8,10			r_5	r_5			
9				r_1			

4.5.5 LALR（1）分析法

对一给定的文法G而言，其LALR(1)分析表比LR(1)分析表状态数要少(例中LALR(1)的状态数比LR(1)状态数减少了4个)，但在分析文法G[S]的某一个含有错误的符号串时，LALR(1)分析速度比LR(1)分析速度要慢，这是因为合并同心集后多做不必要的归约，从而推迟发现错误。

4.5.5 小结

1.分析能力：LR(0)分析、 SLR(1)分析、LALR(1)分析、LR(1)分析依次增强

2. LR(0)⊂SLR(1)⊂LALR(1)⊂ LR(1)

3.任何二义性文法都不是LR类文法

4.5.6 对二义性文法的应用

例如，考虑算术表达式的二义性文法

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

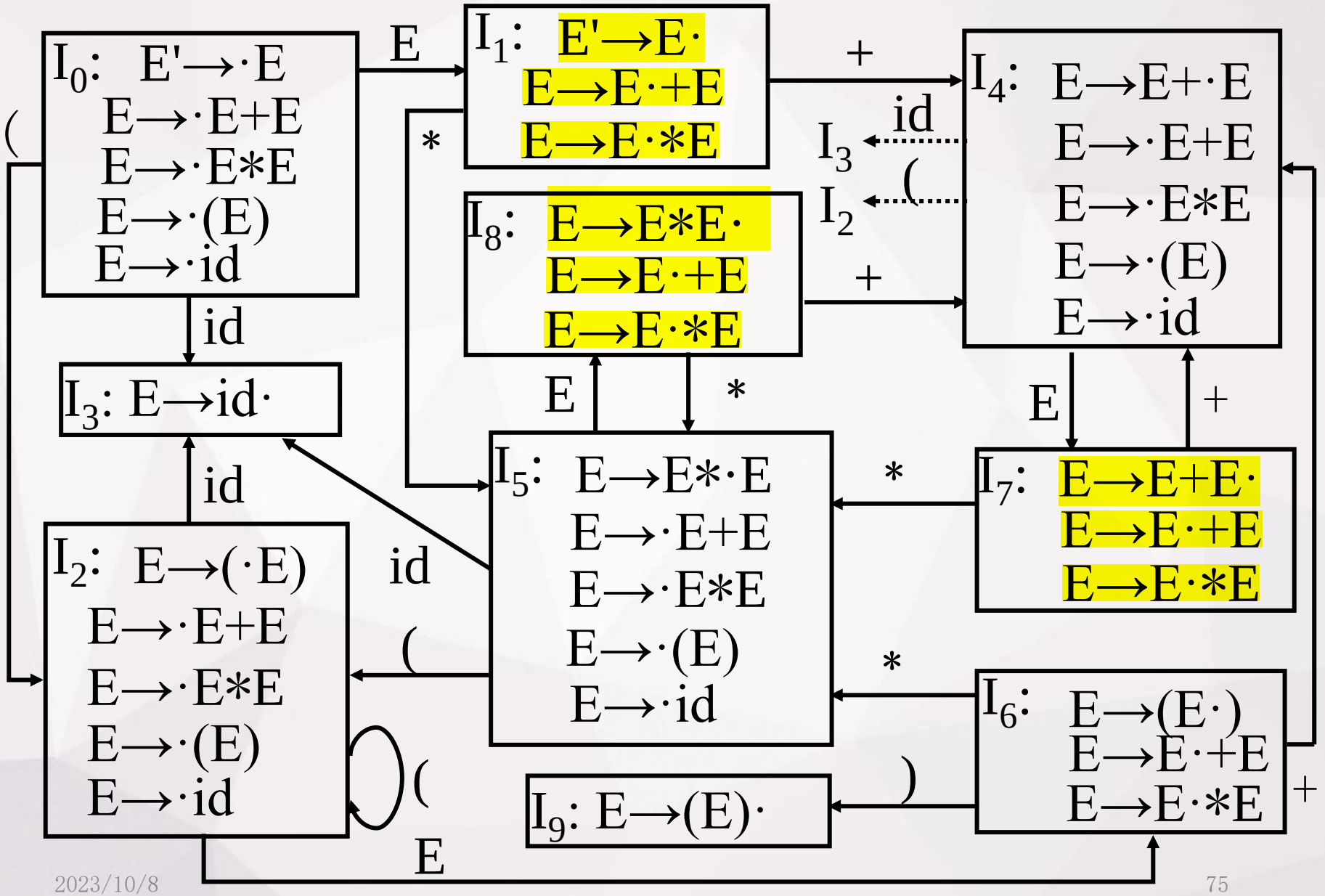
相应的非二义性文法为：

$$E \rightarrow E+T \mid T$$

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

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

算术表达式二义性文法的LR(0)项目集规范族和转移函数



4.5.6 对二义性文法的应用

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

二义性文法的LR分析表

4.5.6 对二义性文法的应用

其它的二义性文法也可用类似方法进行处理，可以构造出无多重定义的LR分析表。