重庆大学
CHONGQING UNIVERSITY

第四讲 语法分析（3）

重庆大学 计算机学院 张敏

4.7.3 LR(1)分析法

例:文法G:

1.S->L=R

2.S->R

3.L->*R

4.L->i

5.R->L

LR(0)项目集 ⟶

I0:S'->•S
S->•L=R
S->•R
L->•*R
L->•i
R->•L

I1:S'->S•

I2:S->L•=R
R->L•

I3:S->R•

I4:L->*•R
R->•L
L->•*R
L->•i

I5:L->i•

I6:S->L=•R
R->•L
L->•*R
L->•i

I7:L->*R•

I8:R->L•

I9:S->L=R•

思考1:FOLLOW(R)={#,=},当在I2状态输入符号为”=“时,会出现什么情况?

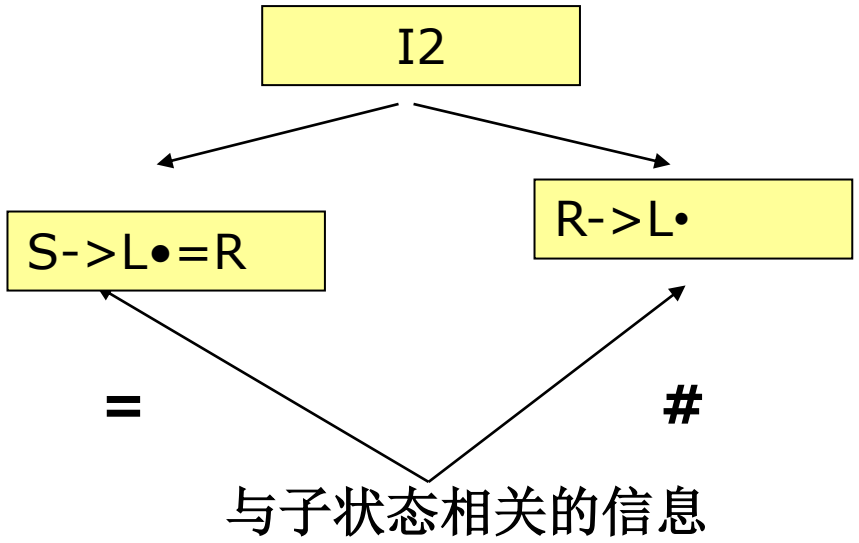
移进-归约冲突

思考2:能否用SLR(1)方法解决? ⟶ {=}∩FOLLOW(R)≠¢

课堂讨论

- 思考：冲突的原因是什么？如何解决？

----该状态中存在互相矛盾的子状态



4.7.3 LR(1)分析法

LR(k)项目

解决项目集内部冲突的办法:把该状态进行分裂,并把与这些状态的展望信息附加在相应的项目上.

在每个项目后面附加若干个展望信息—K个终结符号.

$[A \rightarrow \alpha \ . \beta, \ a_1 a_2 \ldots\ldots a_K]$

我们把这样的项目称为LR(k)项目, $a_1 a_2 \ldots\ldots a_K$ **称为它的向前搜索符号串.**

4.7.3 LR(1)分析法

LR(k)项目

解决项目集内部冲突的办法:把该状态进行分裂,并把与这些状态的展望信息附加在相应的项目上.

在每个项目后面附加若干个展望信息—K个终结符号.

[A →α . β， $a_1 a_2 \ldots\ldots a_k$]，这样的项目称为LR(k)项目

LR(k)项目：[A→α·β，$a_1a_2...a_k$]，其中A→α·β是一个LR(0)项目，a_i(i=1,2,...,k)是终结符号，$a_1a_2...a_k$称为该项目的向前搜索符号串

向前搜索符号串仅对归约项目[A→α·，$a_1a_2\cdots a_k$]起作用，对任何移进或待约项目[A→α·β，$a_1a_2\cdots a_k$]，(β≠ε) 是没有意义的。

归约项目[A→α·，$a_1a_2\cdots a_k$]意味着，当它所属项目集对应的状态在栈顶，且后续的k个输入符号为 $a_1a_2\cdots a_k$时，才允许把栈顶的文法符号串α归约为A

4.7.3 LR(1)分析法

LR(1)项目的一般形式为 [A→α·β，a]，其中，A→αβ是文法中的一个产生式，a是一个终结符或#；当β=ε时，若当前面临的输入符号为a，则用A→α进行规约；否则，a无任何作用。

LR(1)项目的有效性

一个LR(1)项目[A→α·β，a] 对活前缀γ=δα有效，是指存在规范推导

$S \overset{*}{\Rightarrow} \delta Aw \Rightarrow \delta\alpha\beta w$

其中$A \in V_N$，$w \in V_T^*$，$\alpha,\beta \in (V_N \cup V_T)^*$，$a \in (V_T \cup \{\#\})$且满足以下条件：

① if w=ε，then a为#

② else，then **a∈FIRST(βw)**

4.7.3 LR(1)分析法

LR(1)有效项目集和LR(1)项目集规范族

LR(1)有效项目集:

文法G的某个活前缀γ的所有LR(1)有效项目组成的集合称为γ的LR(1)有效项目集。

文法G的所有LR(1)有效项目集组成的集合称为G的LR(1)项目集规范族。

4.7.3 LR(1)分析法

LR(1)项目集族构造

LR(1)项目集族的构造基本方法：

针对初始项目**S’ →•S,#**求闭包后再用转换函数逐步求出整个文法的LR(1)项目集族。

4.7.3 LR(1)分析法

LR(1)项目集闭包（closure）的定义：

设I是文法G的一个LR(1)项目集，closure(I)是从I出发，用下面的方法构造的项目集

(1) I中的每一个项目都属于closure(I)；

(2) 若项目[A→α·Bβ，a]属于closure(I)，且B→η是G的一个产生式，则对任何终结符号b∈FIRST(βa)，若项目[B→·η，b]不属于集合closure(I)，则将它加入closure(I)；

(3) 重复规则(2)，直到closure(I)不再增大为止。

4.7.3 LR(1)分析法

算法： closure(I)的构造过程

输入：项目集合I。

输出：集合J=closure(I)。

方法：

```
J=I;
do {
J_new=J;
for ( J_new中的每一个项目[A→α·Bβ，a] 和
      文法G的每个产生式B→η )
    for (FIRST(βa)中的每一个终结符号b)
        if ([B→·η，b]∉J) 把[B→·η，b]加入J;
} while (J_new!=J);
```

4.7.3 LR(1)分析法

转移函数go

若I是文法G的一个LR(1)项目集，X是一个文法符号，定义：

$$go(I,X)=closure(J)$$

其中：$J=\{ [A\rightarrow\alpha X\cdot\beta，a] \mid$ 当$[A\rightarrow\alpha\cdot X\beta，a]$属于I时 $\}$，项目$[A\rightarrow\alpha X\cdot\beta，a]$称为$[A\rightarrow\alpha\cdot X\beta，a]$的后继。

直观含义：

若I是某个活前缀 γ 的有效项目集，

则 go(I,X) 便是活前缀 γX 的有效项目集。

4.7.3 LR(1)分析法

转移函数go

若I是文法G的一个LR(1)项目集，X是一个文法符号，定义：

$$go(I,X)=closure(J)$$

其中：$J=\{ [A\rightarrow\alpha X\cdot\beta，a] \mid$ 当$[A\rightarrow\alpha\cdot X\beta，a]$属于I时 $\}$，项目$[A\rightarrow\alpha X\cdot\beta，a]$称为$[A\rightarrow\alpha\cdot X\beta，a]$的后继。

直观含义：

若I是某个活前缀 γ 的有效项目集，

则 go(I,X) 便是活前缀 γX 的有效项目集。

4.7.3 LR(1)分析法

算法：计算GO (I, X)

```
procedure GO (I, X)
    begin
     J={任何形如[A→αX·β，a]的项目 | [ A→α·Xβ，a∈I]};
     return CLOSURE (J);
    end;
```

4.7.3 LR(1)分析法

算法： LR(1)有效项目集族C的构造算法

输入: G的拓广文法G′
输出: G的LR(1)的项目集族 C
begin
  C:={CLOSURE ({[S′→·S, #]})};
  repeat
    for C 中每个项目集I和G′中每个文法符号X
    do if GO(I, X)非空且不属于C then将GO(I, X)加到C
  until C不再增大;
end

4.7.3 LR(1)分析法

示例：构造如下文法的LR(1)项目集规范族

(1) S→CC　(2) C→cC　(3) C→d　　（文法4.8）

- 拓广文法：

  (0) S′→S　(1) S→CC　(2) C→cC　(3) C→d

4.7.3 LR(1)分析法

示例：构造如下文法的LR(1)项目集规范族

(1) S→CC　(2) C→cC　(3) C→d　　(文法4.8)

续：

根据算法4.6构造其LR(1)项目集规范族。

I_0=closure({[S′→·S,$]})={[S′→·S,$] [S→·CC,$] [C→·cC,c/d] [C→·d,c/d]}

I_1=go(I_0,S)=closure({[S′→S·,$]})={[S′→S·,$]}

I_2=go(I_0,C)=closure({[S→C·C,$]})={[S→C·C,$] [C→·cC,$] [C→·d,$]}

I_3=go(I_0,c)=closure({[C→c·C,c/d]})={[C→c·C,c/d] [C→·cC,c/d] [C→·d,c/d]}

I_4=go(I_0,d)=closure({[C→d·,c/d]})={[C→d·,c/d]}

I_5=go(I2,C)=closure({[S→CC·,$]})={[S→CC·,$]}

I_6=go(I_2,c)=closure({[C→c·C,$]})={[C→c·C,$] [C→·cC,$] [C→·d,$]}

I_7=go(I_2,d)=closure({[C→d·,$]})={[C→d·,$]}

I_8=go(I_3,C)=closure({[C→cC·,c/d]})={[C→cC·,c/d]}

go(I_3,c)=closure({[C→c·C,c/d]})=I_3

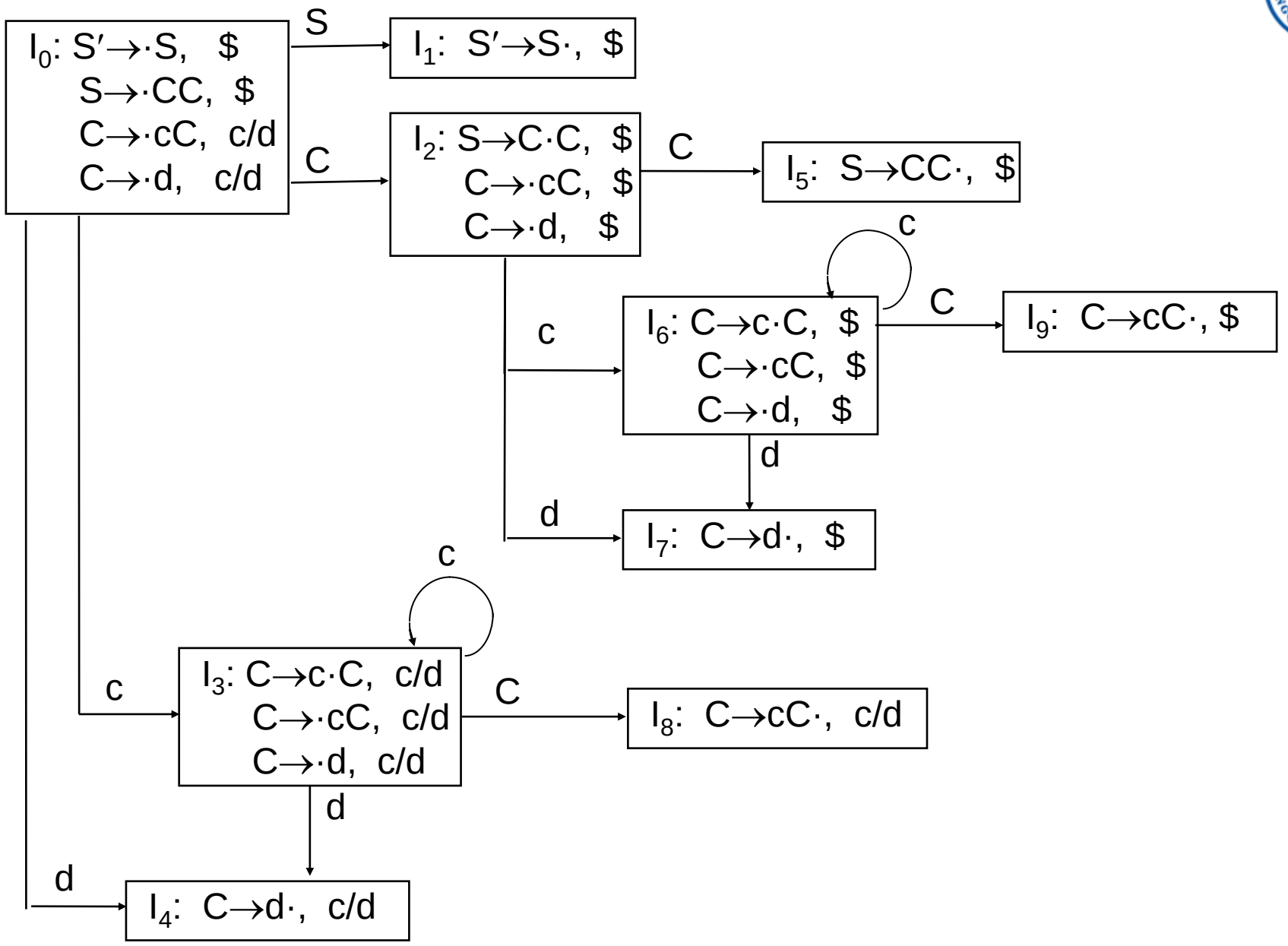
go(I_3,d)=closure({[C→d·,c/d]})=I_4

I_9=go(I_6,C)=closure({[C→cC·,$]})={[C→cC·,$]}

go(I_6,c)=closure({[C→c·C,$]})=$I_6$

go(I_6,d)=closure({[C→d·,$]})=$I_7$

示例(续)：识别所有活前缀的DFA



(0) S′→S
(1) S→CC
(2) C→cC
(3) C→d

4.7.3 LR(1)分析法

算法：构造LR(1)分析表

输入：拓广文法G′

输出：文法G′的分析表

1.构造文法G′的LR(1)项目集规范族C={$I_0,I_1,\ldots,I_n$}

2.对于状态i（代表项目集I_i），分析动作如下：

a) 若$[A\rightarrow\alpha\cdot a\beta,b]\in I_i$，a为终结符且$go(I_i,a)=I_j$，则置action[i,a]=sj

b) 若$[A\rightarrow\alpha\cdot,a]\in I_i$，且$A\neq S'$，则置action[i,a]=rj

c) 若$[S'\rightarrow S\cdot,\$]\in I_i$，则置action[i,$]=acc

3.若对非终结符号A，有$go(I_i,A)=I_j$，则置goto[i,A]=j

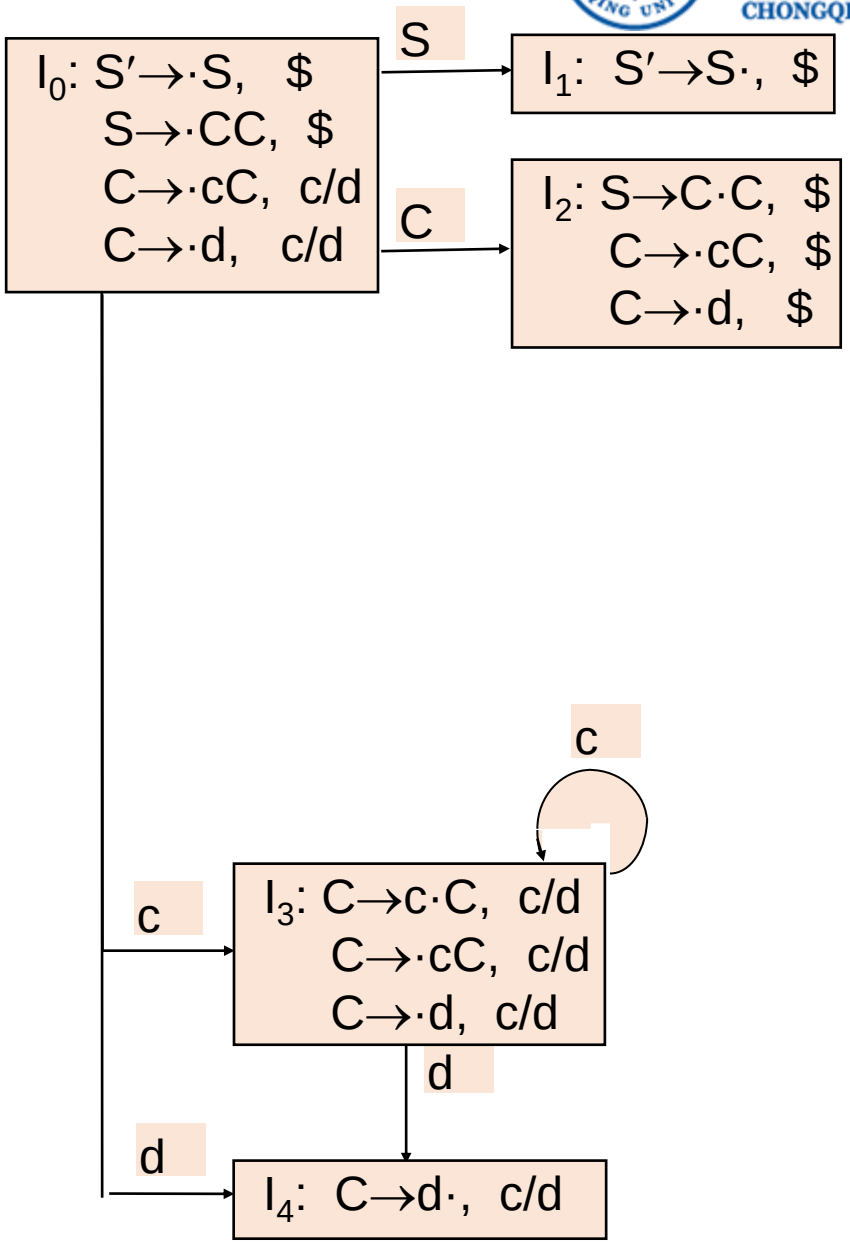
4.凡是不能用上述规则填入信息的空白表项，均置上出错标志error。

5.分析程序的初态是包括$[S'\rightarrow\cdot s,\$]$的有效项目集所对应的状态。

4.7.3 LR(1)分析法

LR(1)分析表

- 考察I_0:
  - [S′→·S,$] 且 go($I_0$,S)=$I_1$， 故：goto[0,S]=1
  - [S→·CC,$] go($I_0$,C)=$I_2$ goto[0,C]=2
  - [C→·cC,c/d] go(I_0,c)=I_3 action[0,c]=s3
  - [C→·d,c/d] go(I_0,d)=I_4 action[0,d]=s4
- 考察I_1：由于[S′→S·,$] 故action[1,$]=acc
- 考察I_2:
  - [S→C·C,$] 且 go($I_2$,C)=$I_5$ 故：goto[2,C]=5
  - [C→·cC,$] go($I_2$,c)=$I_6$ action[2,c]=s6
  - [C→·d,$] go($I_2$,d)=$I_7$ action[2,d]=s7
- 考察I_4:
  - [C→d·,c/d] 故 action[4,c]=action[4,d]=r3



4.7.3 LR(1)分析法

LR(1)分析表

状态	action			goto	
	c	d	$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

4.7.4 LALR(1)分析法

LR(1)分析表与LR(0)分析表相比较又什么优劣？可以怎样改进？

4.7.4 LALR(1)分析法

描述LR(1)项目集特征的两个概念

同心集:
如果两个LR(1)项目集去掉搜索符号之后是相同的，则称这两个项目集具有相同的心（core），即这两个项目集是同心集。

核:
除去初态项目集外，一个项目集的核（kernel）是由该项目集中那些圆点不在最左边的项目组成。

LR(1)初态项目集的核中有且只有项目
[S′→·S,$]。

4.7.4 LALR(1)分析法

构造LALR(1)分析表的基本思想：

- 合并LR(1)项目集规范族中的同心集，以减少分析表的状态数。
- 用核代替项目集，以减少项目集所需的存储空间。
- go(I,X)仅仅依赖于I的心，因此LR(1)项目集合并后的转移函数可以通过go(I,X)自身的合并得到。
- 同心集的合并，可能导致归约-归约的冲突，但不会产生新的移进-归约冲突？

为什么？

4.7.4 LALR(1)分析法

同心集的合并不会引进新的移进-归约冲突

如果合并后的项目集中存在移进-归约冲突，则意味着：

项目[A→α·,a]和[B→β·aγ,b] 处于合并后的同一项目集中，合并前必存在某个c，使得[A→α·,a]和[B→β·aγ,c]同处于某个项目集中。说明，原来的LR(1)项目集中已经存在移进-归约冲突。而这与合并前为LR(1)项目所冲突。

4.7.4 LALR(1)分析法

同心集的合并可能导致归约-归约冲突

例：有文法G：

S→aAd|bBd|aBe|bAe

A→c

B→c　　(文法4.9)

- 拓广文法G′

(0) S′→S

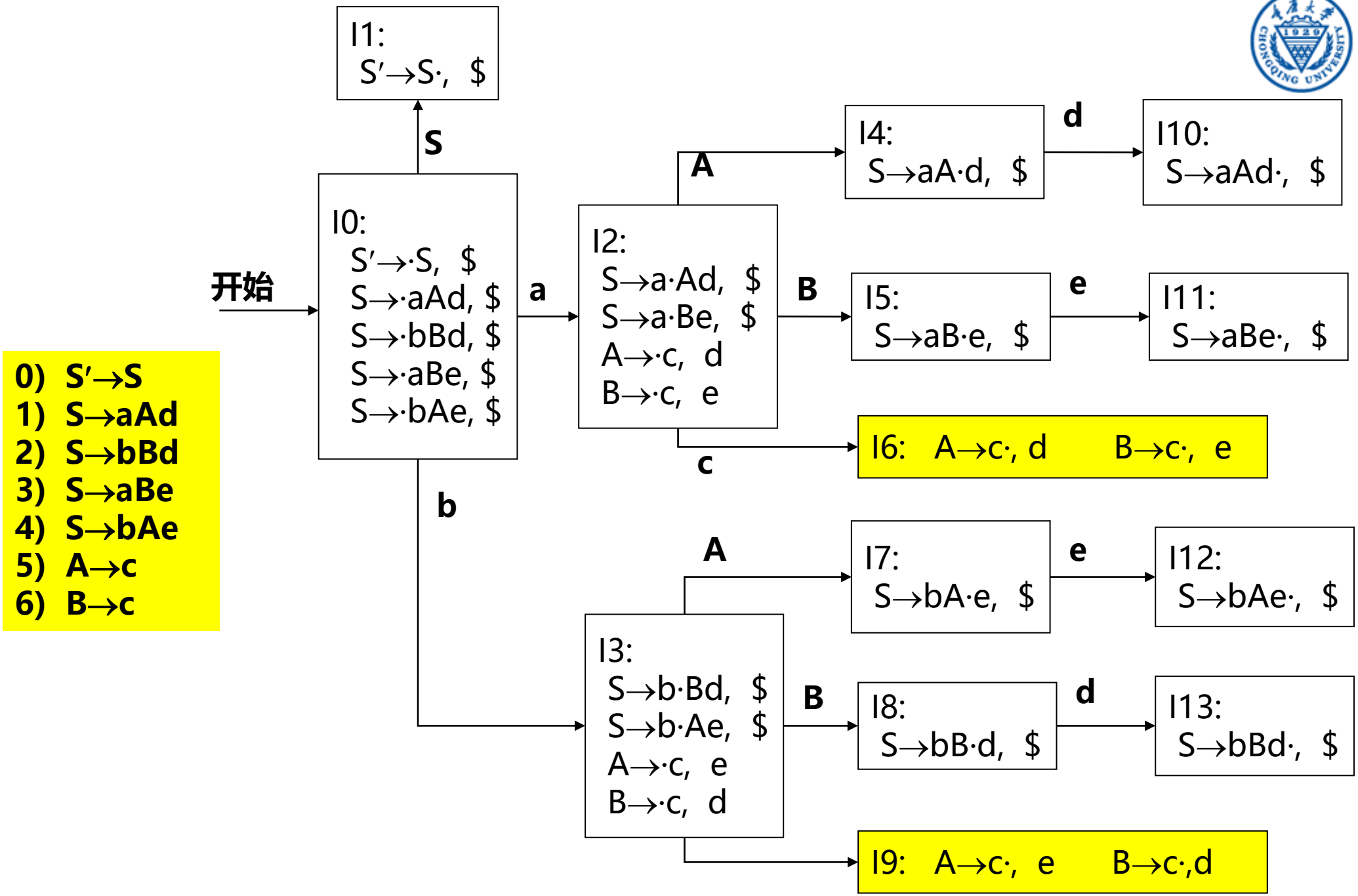
(1) S→aAd (2) S→bBd (3) S→aBe (4) S→bAe

(5) A→c (6) B→c

- 构造识别文法G′的所有活前缀的DFA

I_0={[S′→·S,$]

[S→·aAd,$] [S→·bBd,$] [S→·aBe,$] [S→·bAe,$]}


0) S′→S
1) S→aAd
2) S→bBd
3) S→aBe
4) S→bAe
5) A→c
6) B→c
I1:
S′→S·, $
S
I0:
S′→·S, $
S→·aAd, $
S→·bBd, $
S→·aBe, $
S→·bAe, $
开始
a
b
I2:
S→a·Ad, $
S→a·Be, $
A→·c, d
B→·c, e
A
B
c
I4:
S→aA·d, $
d
I10:
S→aAd·, $
I5:
S→aB·e, $
e
I11:
S→aBe·, $
I6: A→c·, d B→c·, e
I3:
S→b·Bd, $
S→b·Ae, $
A→·c, e
B→·c, d
A
B
I7:
S→bA·e, $
e
I12:
S→bAe·, $
I8:
S→bB·d, $
d
I13:
S→bBd·, $
I9: A→c·, e B→c·,d

4.7.4 LALR(1)分析法

状态	action						goto		
	a	b	c	d	e	$	S	A	B
0	s2	s3					1		
1						acc			
2			s6					4	5
3			s9					7	8
4				s10					
5					s11				
6				r5	r6				
7					s12				
8				s13					
9				r6	r5				
10						r1			
11						r3			
12						r4			
13						r2			

4.7.4 LALR(1)分析法

合并同心集

- LR(1)项目集规范族中
  - 活前缀ac的有效项目集是I_6
  - 活前缀bc的有效项目集是I_9
  - 这两个项目集都不含冲突项目，且是同心集。

- 它们合并后得到的集合为：
  {[A→c·, d/e] [B→c·, d/e]}

- 含有归约-归约冲突。

4.7.4 LALR(1)分析法

LALR(1)分析表的构造

- 基本思想是：
  - 首先构造LR(1)项目集规范族；
  - 如果它不存在冲突，就把同心集合并在一起；
  - 若合并后的项目集规范族不存在归约-归约冲突，就按这个项目集规范族构造分析表。

4.7.4 LALR(1)分析法

算法 构造LALR(1)分析表

输入：一个拓广文法G′

输出：文法G′的LALR(1)分析表

方法：

1.构造文法G′的LR(1)项目集规范族C={I_0，I_1，…，I_n}。

2.合并C中的同心集，得到一个新的项目集规范族C′={J_0,J_1,…,J_m}，其中含有项目[S′→·S,$]的$J_k$为分析表的初态。

3.从C′出发，构造action子表

a) 若[A→α·aβ,b]∈J_i，且go(J_i,a)=J_j，则置action[i,a]=sj

b) 若[A→α·,a]∈J_i，则置action[i,a]=r A→α

c) 若[S′→S·,$]∈$J_i$，则置action[i,$]=acc

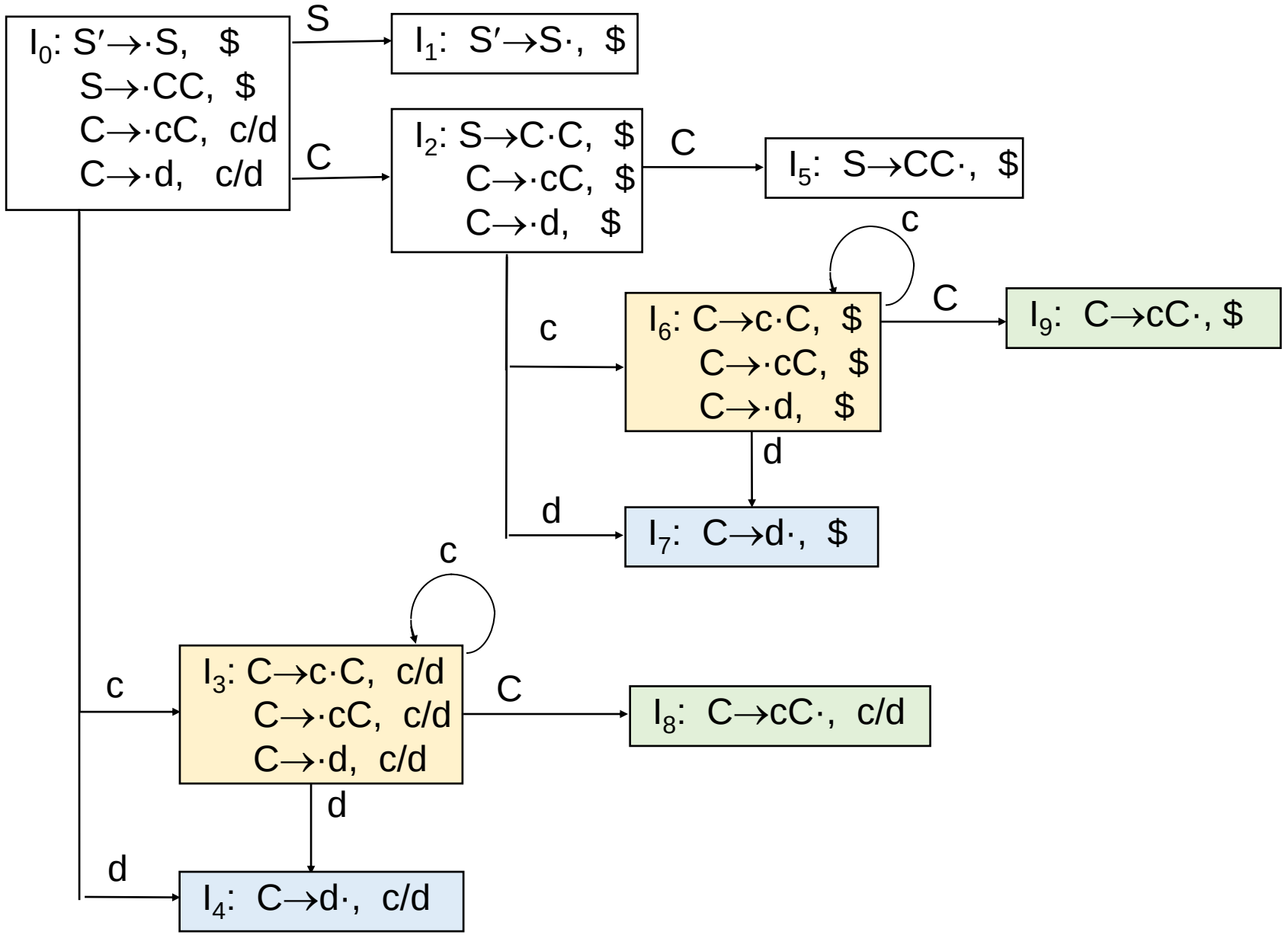
4.构造goto子表

设Jk={Ii1，Ii2，…，Iit},由于这些Ii是同心集，因此 go(Ii1,X)、go(Ii2,X)、…、go(Iit,X)也是同心集

把所有这些项目集合并后得到的集合记作Ji，则有：go(Jk,X)=Ji,于是，若go(Jk,A)=Ji，则置goto[k,A]=I

5.分析表中凡不能用上述规则填入信息的空表项，均置上出错标志。

示例: 构造文法4.8的LALR(1)分析表



(0) S′→S
(1) S→CC
(2) C→cC
(3) C→d

4.7.4 LALR(1)分析法

- 有三对同心集可以合并，即
  - I_3和I_6合并，得到项目集：
    I_{36}={[C→c·C,c/d/$] [C→·cC,c/d/$] [C→·d,c/d/$]}
  - I_4和I_7合并，得到项目集：
    I_{47}={[C→d·, c/d/$]}
  - I_8和I_9合并，得到项目集：
    I_{89}={[C→cC·, c/d/$]}
- 同心集合并后得到的新的项目集规范族为：
  C'={I_0, I_1, I_2, I_{36}, I_{47}, I_5, I_{89}}
- 利用算法4.8，可以为该文法构造LALR(1)分析表。

4.7.4 LALR(1)分析法

文法4.8的LALR(1)分析表

状态	action			goto	
	c	d	$	S	C
0	s36	s47		1	2
1			acc		
2	s36	s47			5
36	s36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

- 此文法是LALR(1)文法
- 转移函数go的计算只依赖于项目集的心

4.7.4 LALR(1)分析法

LALR(1)分析程序对符号串cdcd的分析过程

步骤	栈	输入	分析动作
0	0	cdcd$	shift
1	0c36	dcd$	shift
2	0c36d47	cd$	reduce by C→d
3	0c36C89	cd$	reduce by C→cC
4	0C2	cd$	shift
5	0C2c36	d$	shift
6	0C2c36d47	$	reduce by C→d
7	0C2c36C89	$	reduce by C→cC
8	0C2C5	$	reduce by S→CC
9	0S1	$	accept

4.7.4 LALR(1)分析法

LR(1)分析程序对符号串cdcd的分析过程

步骤	栈	输入	分析动作
0	0	cdcd$	shift
1	0c3	dcd$	shift
2	0c3d4	cd$	reduce by C→d
3	0c3C8	cd$	reduce by C→cC
4	0C2	cd$	shift
5	0C2c6	d$	shift
6	0C2c6d7	$	reduce by C→d
7	0C2c6C9	$	reduce by C→cC
8	0C2C5	$	reduce by S→CC
9	0S1	$	accept

4.7.4 LALR(1)分析法

LR(1)分析程序对符号串ccd的分析过程

步骤	栈	输入	分析动作
0	0	ccd$	shift
1	0c3	cd$	shift
2	0c3c3	d$	shift
3	0c3c3d4	$	error

状态	action			goto	
	c	d	$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

4.7.4 LALR(1)分析法

LALR(1)分析程序对符号串ccd的分析过程

步骤	栈	输入	分析动作
0	0	ccd$	shift
1	0c36	cd$	shift
2	0c36c36	d$	shift
3	0c36c36d47	$	reduce by C→d
4	0c36c36C89	$	reduce by C→cC
5	0c36C89	$	reduce by C→cC
6	0C2	$	error

状态	action			goto	
	c	d	$	S	C
0	s36	s47		1	2
1			acc		
2	s36	s47			5
36	s36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

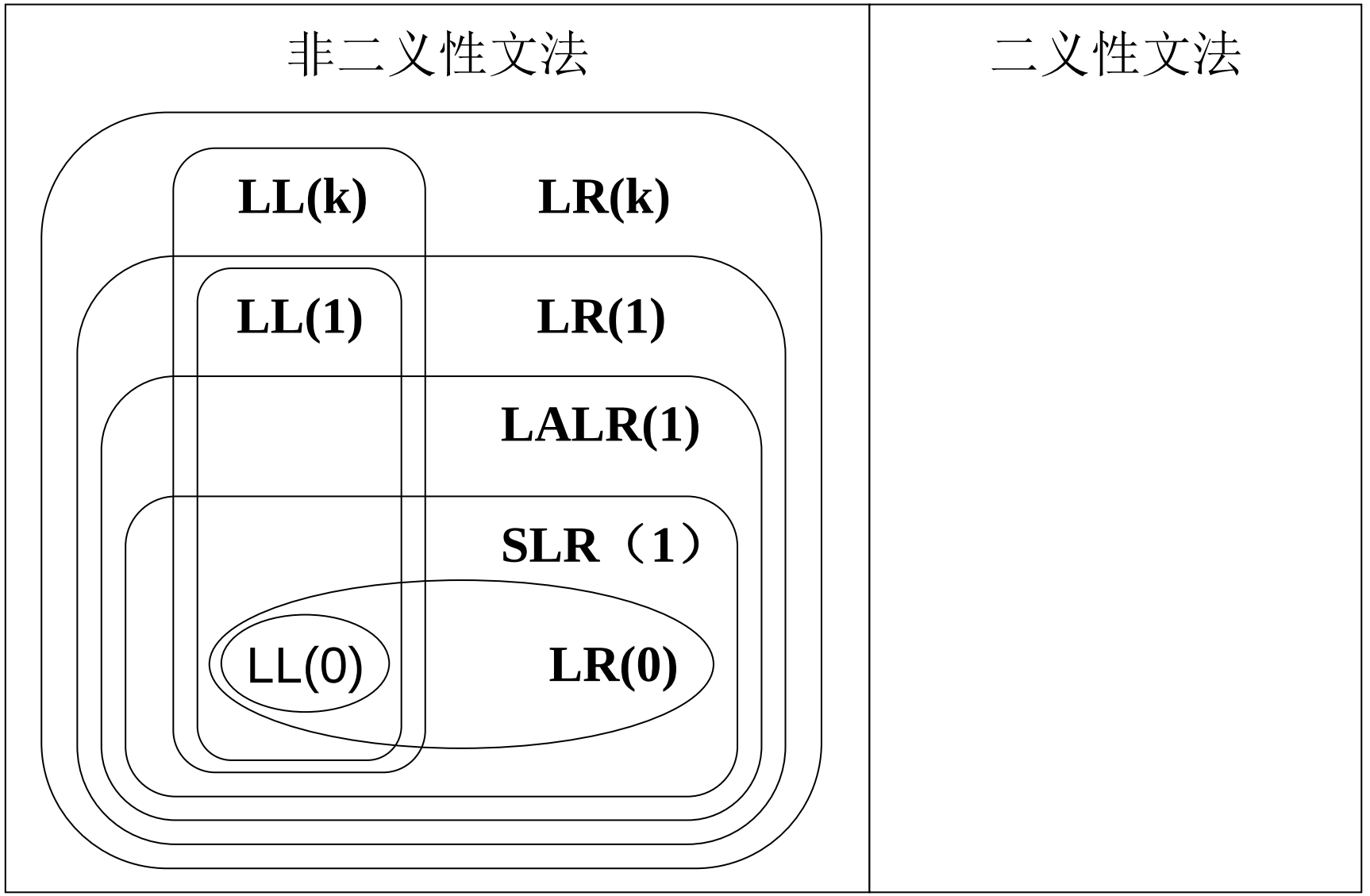
4.7.4 LALR(1)分析法

定理：任何二义性文法决不是LR文法，因而也不是SLR或LALR文法。

程序设计语言的某些结构用二义性文法描述比较直观，使用方便。例如关于算术表达式的文法和if语句的文法。

在所有情况下，都说明了消除二义性的一些规则（即这类结构的使用限制）。

4.7.4 LALR(1)分析法



4.7.4 LALR(1)分析法

利用优先级和结合规则解决表达式冲突

- 描述算术表达式集合的二义性文法：
  E→E+E|E*E|(E)|id　　(文法4.10)
- 无二义性的文法：
  E→E+T|T
  T→T*F|F
  F→(E)|id
- 前者具有两个明显的优点：
  - 改变运算符号的优先级或结合规则时，文法本身无需改变，只需改变限制条件。
  - LR分析表所包含的状态数比后者少得多

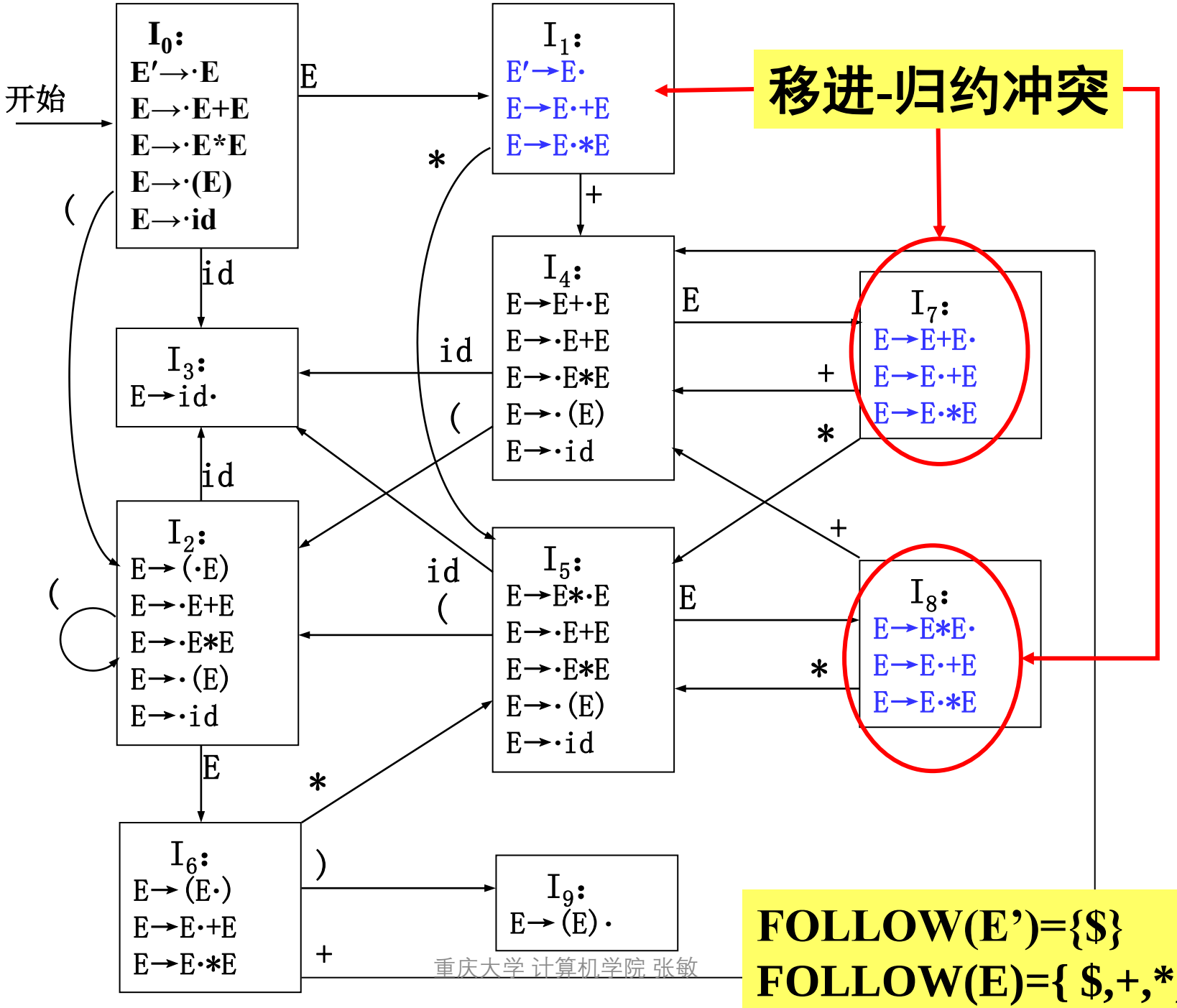
4.7.4 LALR(1)分析法

例：其拓广文法G′具有产生式：

(0) E′→E　　(1) E→E+E　(2) E→E*E

(3) E→(E)　　(4) E→id

构造文法G′的LR(0)项目集规范族及识别所有活前缀的DFA



移进-归约冲突

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

‘+’ 和 ‘*’ 的优先级及结合规则共有4种情况

(1) * 优先于 + ， 遵从左结合规则

(2) * 优先于 + ， 遵从右结合规则

(3) + 优先于 * ， 遵从左结合规则

(4) + 优先于 * ， 遵从右结合规则

I_7:
E→E+E·
E→E·+E
E→E·*E

I_8:
E→E*E·
E→E·+E
E→E·*E

条件	状态	action					goto
		id	+	*	(	)	$	E
(1)	7		r1	s5		r1	r1	
	8		r2	r2		r2	r2	
(2)	7		s4	s5		r1	r1	
	8		r2	s5		r2	r2	
(3)	7		r1	r1		r1	r1	
	8		s4	r2		r2	r2	
(4)	7		s4	r1		r1	r1	
	8		s4	s5		r2	r2	

4.7.4 LALR(1)分析法

状态	action						goto
	id	+	*	(	)	$	E
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	

4.7.4 LALR(1)分析法

利用最近最后匹配原则解决if语句冲突

- 映射程序设计语言中if-then-else结构的文法：

  S→ if E then S else S
  | if E then S
  | others　　(文法4.11)

- 对该文法进行抽象
  - 用i表示“if E then”
  - 用e表示“else”
  - 用a表示“others”
- 得到文法：

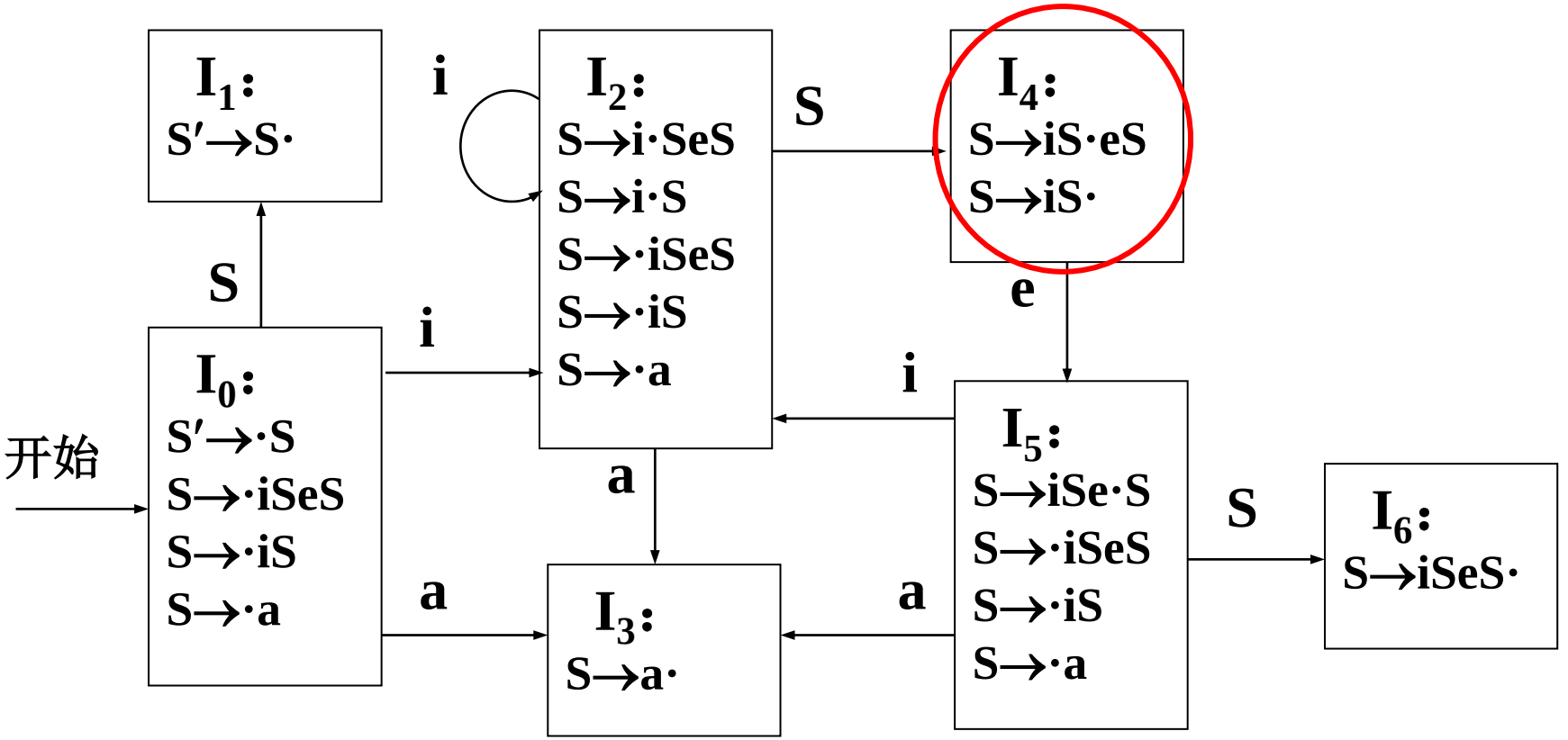
  S→iS | iSeS | a　　(文法4.12)

4.7.4 LALR(1)分析法

文法4.12 的LR(0)项目集规范族及识别其所有活前缀的DFA

其拓广文法G′为:

(0) S′→S (1) S→iSeS (2) S→iS (3) S→a



4.7.4 LALR(1)分析法

最近最后匹配原则：

else与离它最近的一个未匹配的then相匹配

文法4.14的LR分析表

状态	action				goto
	i	e	a	$	S
0	s2		s3		1
1				acc	
2	s2		s3		4
3		r3		r3	
4		s5		r2	
5	s2		s3		6
6		r1		r1	

例：分析输入符号串iiaea

步骤	栈	输入	分析动作
1	0 -	iiaea$	shift
2	0 2 - i	iaea$	shift
3	0 2 2 - i i	aea$	shift
4	0 2 2 3 - i i a	ea$	reduce by S→a
5	0 2 2 4 - i i S	ea$	shift
6	0 2 2 4 5 - i i S e	a$	shift
7	0 2 2 4 5 3 - i i S e a	$	reduce by S→a
8	0 2 2 4 5 6 0 i i S e S	$	reduce by S→iSeS
9	0 2 4 - i S	$	reduce by S→iS
10	0 1 - S	$	accept

4.7.5 LR分析的错误处理与恢复

- LR分析程序可采取以下恢复策略：
  - 首先，从栈顶开始退栈，可能弹出0个或若干个状态，直到出现状态S为止，根据S在goto子表中可以找到一个非终结符号A，即它有关于A的转移；
  - 然后，抛弃0个或若干个输入符号，直到找到符号a为止，$a \in FOLLOW(A)$，即a可以合法地跟在A的后面；
  - 然后，分析程序把状态goto[S,A]压入栈，继续进行语法分析。
- 在弹栈过程中出现的A可能不止一个，通常选择表示主要结构成分的非终结符号。
- 实际上是跳过包含错误的一部分终结符号串。

例：

符号

期待输入符号为运算符号或右括号，而遇到的却是运算对象（id或左括号）。

诊断信息：“缺少运算符号”

恢复策略：把运算符号‘+’压入栈，转移到状态4。

状态						goto
	id			(	)	$	E
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		
8							
9							

期待输入符号为运算符号或右括号，而遇到的却是输入串结束标志‘$’。

诊断信息：“缺少右括号”

恢复策略：把右括号压入栈，转移到状态9。

4.7.5 LR分析的错误处理与恢复

e1：在状态0、2、4、5，期待输入符号为运算对象的首字符，即id或(，而输入中出现的却是运算符号‘+’或‘*’，或是输入串结束标志‘$’。

- 策略：把一个假想的id压入栈，并将状态3推入栈顶。
- 诊断信息：“缺少运算对象”

e2：在状态0、1、2、4、5，期待输入符号为运算对象的首字符或运算符号，但却遇到右括号。

- 策略：删掉输入的右括号
- 诊断信息：“括号不匹配”

e3：在状态1、6，期待输入符号为运算符号或右括号，而遇到的却是运算对象（id或左括号）。

- 策略：把运算符号‘+’压入栈，转移到状态4。
- 诊断信息：“缺少运算符号”

e4：在状态6，期待输入符号为运算符号或右括号，而遇到的却是输入串结束标志‘$’。

- 策略：把右括号压入栈，转移到状态9。
- 诊断信息：“缺少右括号”

示例：分析符号串id+)

步骤	栈	输入	分析动作
(1)	0	id+)$	shift
	-		
(2)	0 3	+)$	reduce by E→id
	- id		
(3)	0 1	+)$	shift
	- E		
(4)	0 1 4	)$	CALL e2 “括号不匹配”，删掉‘）’
	- E +		
(5)	0 1 4	$	CALL e1 “缺少运算对象”，id压入栈
	- E +		
(6)	0 1 4 3	$	reduce by E→id
	- E + id		
(7)	0 1 4 7	$	reduce by E→E+E
	- E + E	$	reduce by E→E+E
(8)	0 1	$	accept
	- E		

课后作业 1

1、构造下列文法的LR（1）分析表

1.S->AcA

2.A->aA

3.A->d

课后作业 2

请对几种语法分析方法进行总结