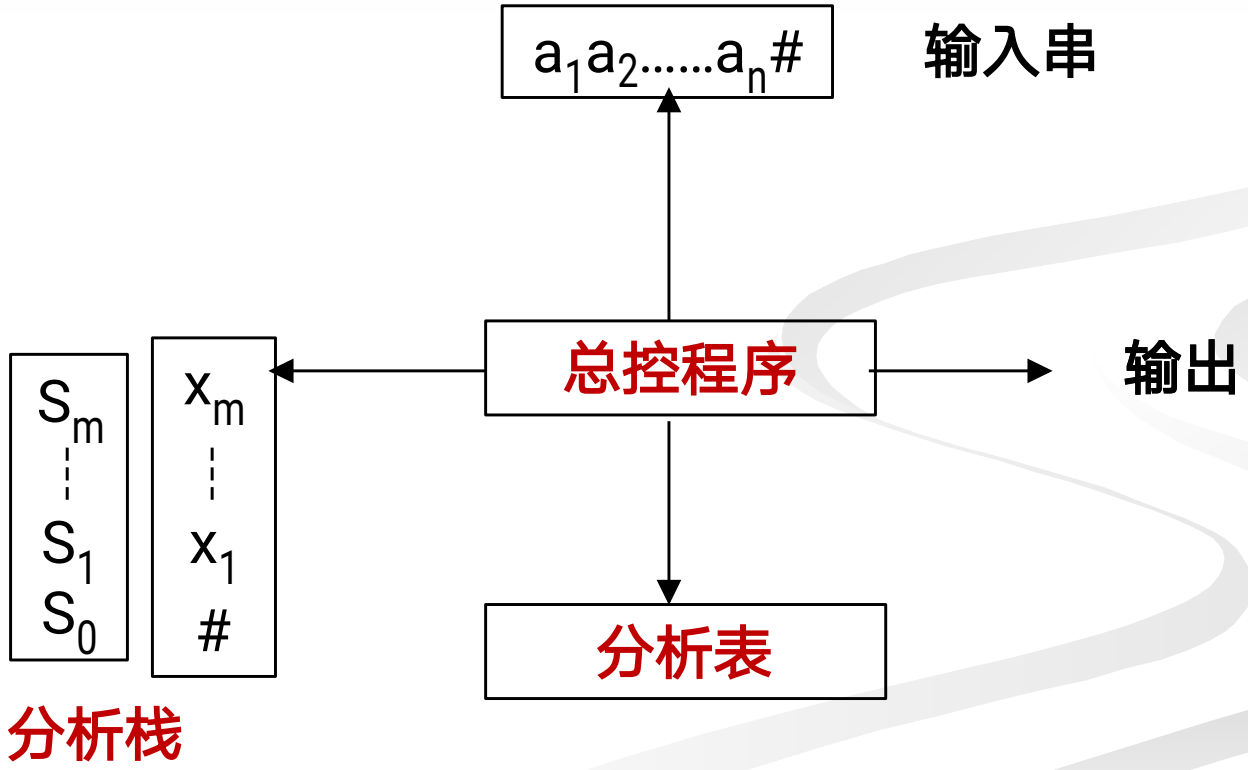
第6章　LR分析法

LR分析法是一类对上下文无关文法进行“自左向右的扫描和最左归约（即最右推导的逆过程）”分析的方法，是一种规范归约过程。LR(k)分析方法是1965年Knuth提出的，其中k表示向右查看输入符号串的个数。

6.1 LR分析器概述

LR分析器工作过程如下图所示。



（1）总控程序 所有LR分析器的总控程序都是相同的，共有四种动作：移进、归约、接受、出错。

（2）分析表 常见的有四种:

LR（0）分析表 适应文法范围小，是其它类型分析表构造的基础。

SLR（1）分析表 是LR（0）分析表的改进，适应文法范围强于LR（0）。

LR（1）分析表 分析能力强（指适应范围，查错速度），但状态太多。

LALR（1）分析表 是LR（1）分析表的改进，分析能力强于SLR而稍弱于LR（1），但状态少于LR（1）

利用四种不同分析表可得到四种不同的LR分析法。

例如 LR（0）分析表的形式如下(见P125 表7.1)

状态	ACTION	GOTO
	列出文法的终结符和#，	列出文法的非终结符，
0	表示当前状态下所面临输	表示当前状态面临
1	入符应做的动作是 移进、	文法符号时应转向的
2	归约、接受、出错 四种之一	下一个状态。
3		
⋮		

（3）分析栈 包括文法符号栈和相应的状态栈。

6.2 LR（0）分析

1、可归前缀

规范句型中，包括句柄及句柄以左的部分，称为可归前缀。

如：文法G[S]：在各产生式尾部加上编号，但编号不是文法符号）

S→aAcBe[1]

A→b[2]

A→Ab[3]

B→d[4]

句子abbcde的规范归约过程为:

S⇒aAcBe[1]

⇒aAcd[4]e[1]

⇒aAb[3]cd[4]e[1]

⇒ab[2]b[3]cd[4]e[1]

其逆过程为最左归约（规范归约）

其归约得规范句型序列如下：

归约为： ab[2]b[3]cd[4]e[1]
aAb[3]cd[4]e[1]
aAcd[4]e[1]
aAcBe[1]
S

其中下划线部分为可归前缀。有些可归前缀的前缀是相同的，不仅仅属于某一个规范句型。我们把可归前缀的前缀称为活前缀。假设某文法G的全部可归前缀为：

α1[P1]
α2[P2]
⋮
αn[Pn]

进行语法分析时，只要将待分析符号串的当前部分符号与αi[Pi]进行比较，便可知是否归约，以及应按哪条产生式归约。

为了得到所有可归前缀，可以对文法G构造一个有穷自动机，该有穷自动机能识别文法G的所有可归前缀。

2、构造识别可归前缀的有穷自动机

（1）项目

文法的识别可归前缀的有穷自动机以文法的“项目”作为它的状态，所谓文法的项目，是在文法的每一条规则的右部添加一个圆点而形成。

如产生式U→XYZ对应四个项目:

U→·XYZ　U→X·YZ　U→XY·Z　U→XYZ·

之所以这样构造项目，是受可归前缀的启发。圆点表示在识别可归前缀的过程中，对句柄（即某产生的右部）已识别过的部分。

项目分四类：

（a）圆点在最右端的项目，形如A→α·，表示已从输入串看到能由一条产生式右部推导出的符号串，即已达一可归前缀末端，已识别出句柄可以归约，这种项目称为归约项目，相应状态称为归约状态。

（b）对形如S′→S· 的项目，其中S是文法开始符号，称为接受项目，相应状态称为接受状态，表明可由S推导的输入串已全部识别完，整个分析过程成功结束。

（c）对于形如A→α·aβ 的项目，表明尚未识别一可归前缀，需将a移进符号栈，故称移进项目，相应状态为移进状态。

（d）对于形如A→α·Bβ的项目，表明期待分析由B所推出的串归约到B，从而识别B。故称为待约项目，相应状态为待约状态。

（2）构造识别可归前缀的有穷自动机

方法一：首先构造识别可归前缀的NFA

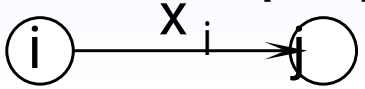
初态　S′→•S

为此，需先将文法拓广，加S′→S产生式。

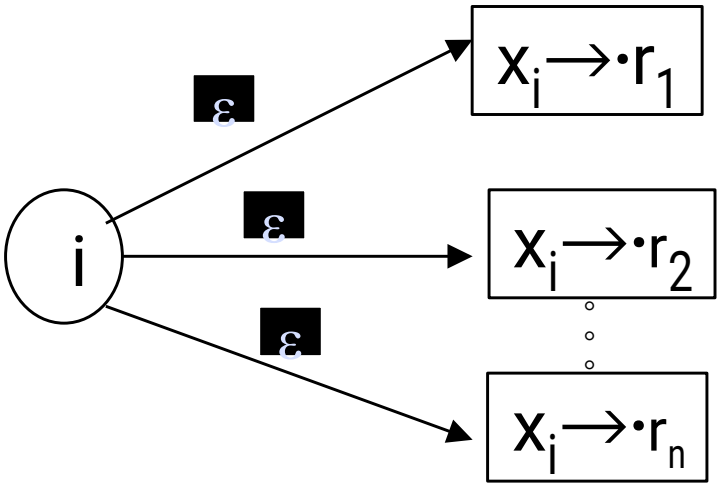
作图：如果状态 i 为：$X \rightarrow x_1 \ldots x_{i-1} \cdot x_i \ldots x_n$

状态 j 为：$X \rightarrow x_1 \ldots x_{i-1} x_i \cdot x_{i+1} \ldots x_n$

则作图：



若 x_i 是非终结符，且 $x_i \rightarrow r_1 | r_2 | \ldots | r_n$　　则再作图：



终态： 形如 $X \rightarrow x_1 x_2 \ldots x_n \cdot$ 即为可归前缀识别态。

例：文法G′为：

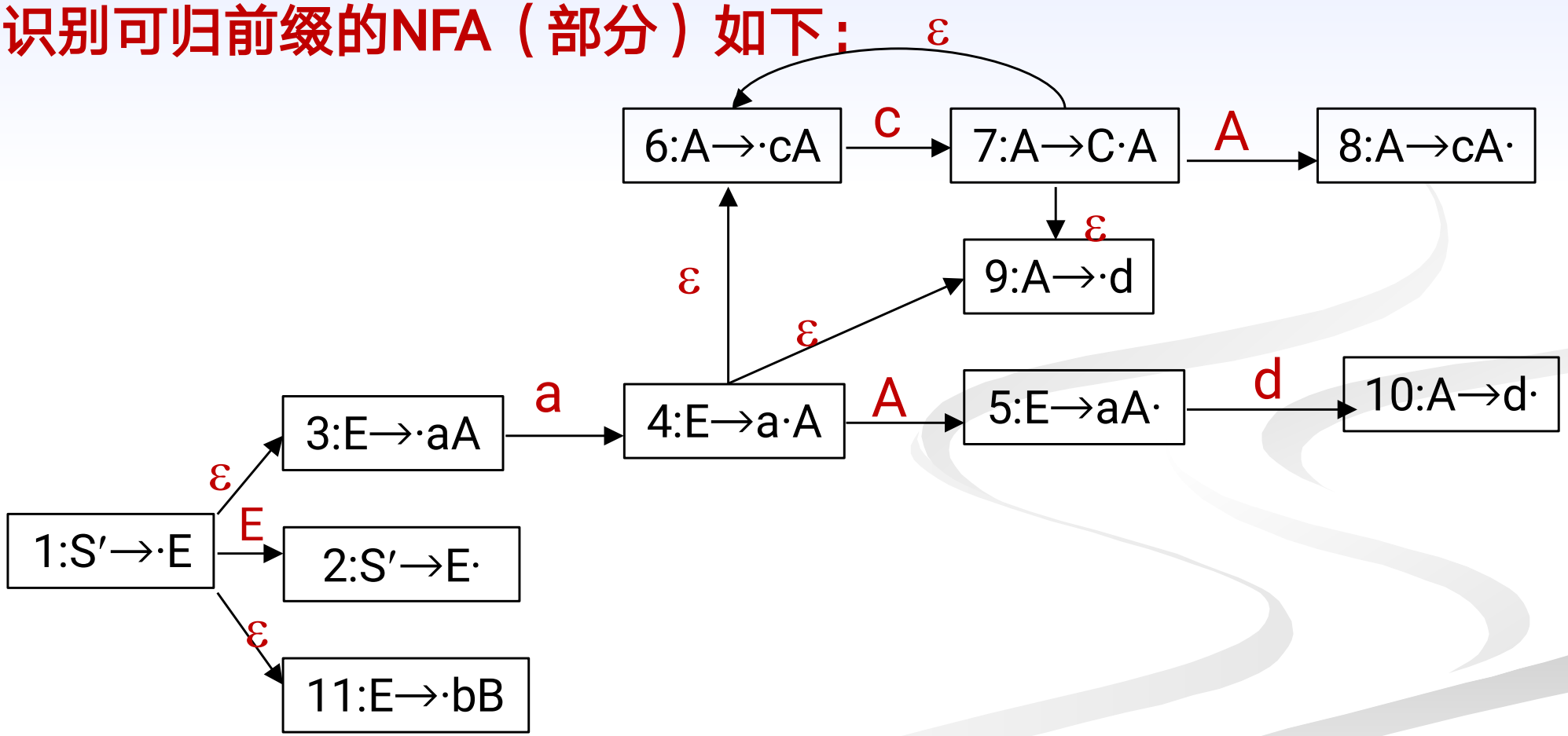
S′→E

E→aA|bB

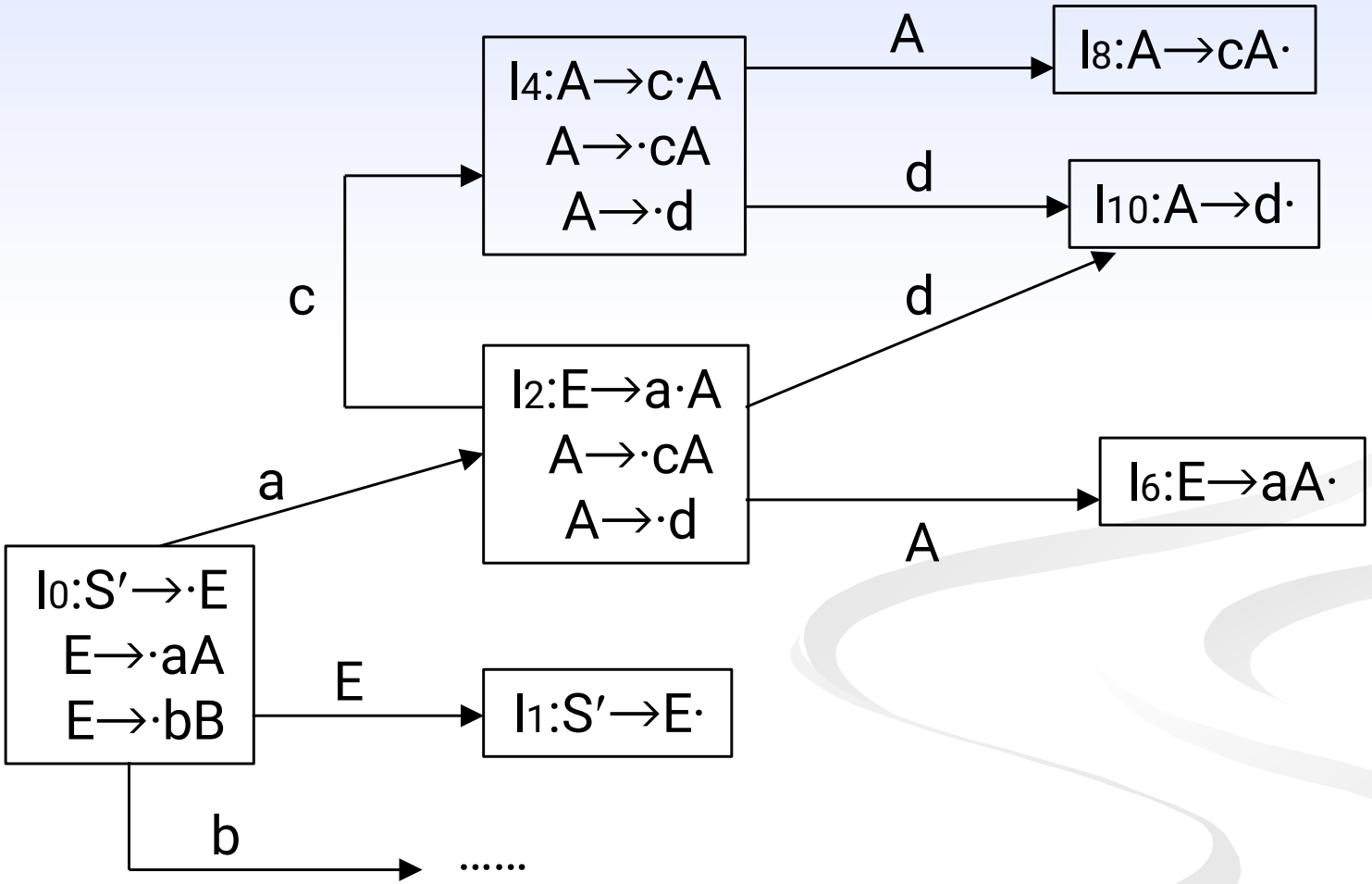
A→cA|d

B→cB|d

识别可归前缀的NFA（部分）如下：



然后将NFA确定化，得DFA（部分）如下：



方法二：根据文法直接构造识别文法可归前缀的DFA。

①拓广文法

对文法G加一条产生式 S′→S 得 G′

目的是使开始状态唯一，接受状态易于识别。

②定义项目集I的闭包CLOSURE(I)

a) I的项目均属于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。

③定义状态转换函数GO（I，X）

其中I是项目集，X是文法符号。
GO（I，X）=CLOSURE(J)
其中J={任何形如A→αx•β的项目|A→α•xβ∈I}
以上可以避免出现 ε 弧，避免从同状态射出相同标记弧。

④构造DFA

a) DFA的初态集：CLOSURE({S′→•S})

b) 对初态集或其它所构造的项目集应用转换函数

GO (I, X) =CLOSURE（J）求出新的项目集。

c) 重复b)直到不出现新的项目集为止。

DFA中所有状态组成的集合也称为该文法的LR（0）项目集规范族。

例：文法G：

（1）S→aAc

（2）A→Abb

（3）A→b

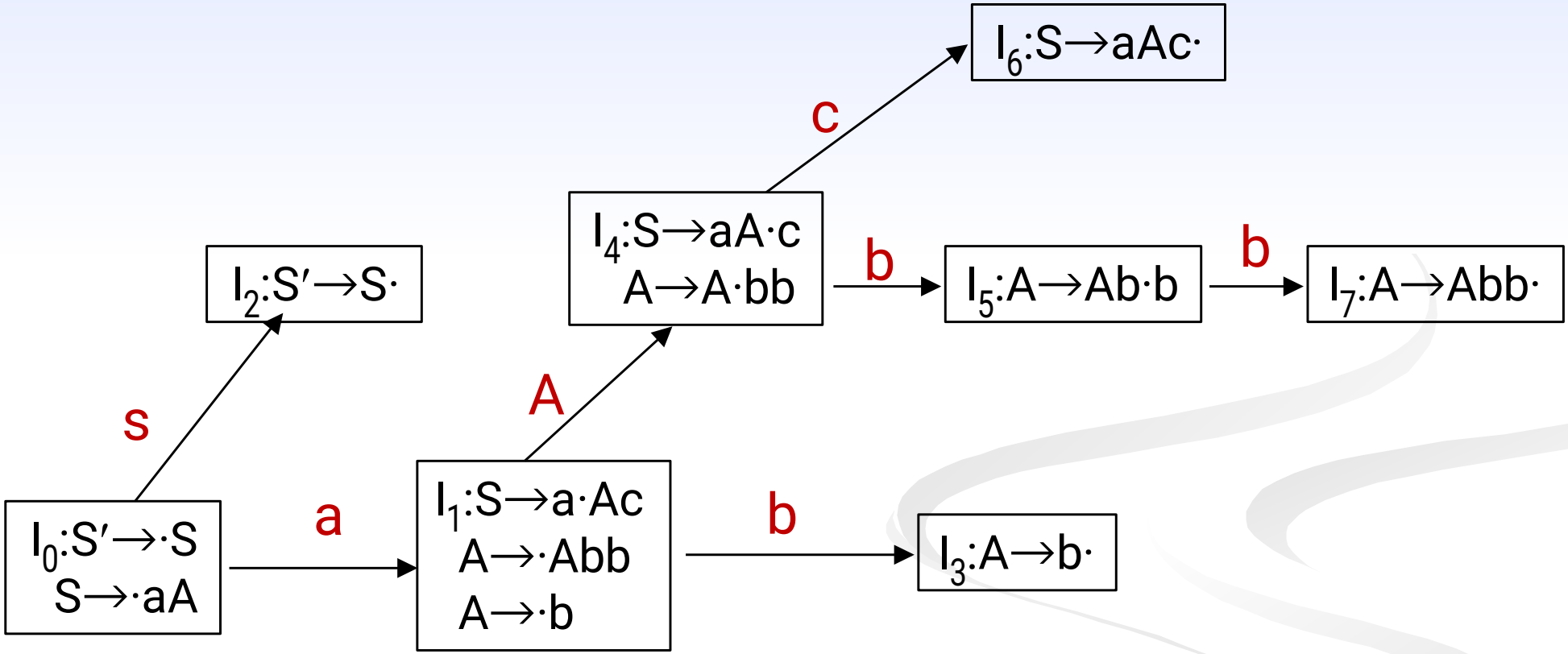
构造识别G可归前缀的DFA（也叫识别G活前缀的DFA）。

解：构造出DFA的状态转换矩阵形式如下：

I	Go(I, a)	Go (I, b)	Go (I, c)	Go (I, s)	Go (I, A)
I_0:{S′→·S S→·aAc}	I_1:{S→a·Ac A→·Abb A→·b }			I_2:{S′→S·}	
I_1:{S→a·Ac A→·Abb A→·b }		I_3{A→b·}			I_4:{S→aA·c A→A·bb}
I_2:{S′→S·}					
I_3:{A→b·}					
I_4:{S→aA·c A→A·bb }		I_5:{A→Ab·b}	I_6:{S→aAc·}		
I_5:{A→Ab·b}		I_7:{A→Abb·}			
I_6:{S→aAc·}					
I_7:{A→Abb·}					

也可画出状态转换图：

看LR（0）分析表



3、构造LR（0）分析表

LR（0）分析表的形式如下(见P125)

状态	ACTION	GOTO
	列出文法的终结符和#，	列出文法的非终结符，
0	表示当前状态下所面临输	表示当前状态面临
1	入符应做的动作是 移进、	文法符号时应转向的
2	归约、接受、出错 四种之一	下一个状态。
3		
⋮		

首先构造文法的识别可归前缀（或话前缀）的DFA；利用DFA的状态转换矩阵，构造LR（0）分析表较为方便。

LR（0）分析表构造算法

用项目集I_k的下标K表示分析表的状态。

其中I_k为项目集的名字，k为状态名，令包含S′→·S项目的集合I_k的下标k为分析器的初始状态。那么分析表的ACTION表和GOTO表构造步骤为：

a) 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，其动作含意为将终结符a移进符号栈，状态j进入状态栈，（即当状态k遇a时转向状态j）。

b) 若项目A→α·属于I_k，则对a为任何终结符或‘#’号，置ACTION[k,a]为“r_j”，j为在文法G′中某产生式A→α的序号。r_j 动作的含义是把当前文法符号栈顶的符号串α归约为A，且将栈指针从栈顶向下移动|α|的长度（或符号栈中弹出|α|个符号），非终结符A变为当前面临的符号。

c) 若GO（Ik, A）= Ij，则置GOTO[k, A]为“j”，其中A为非终结符，表示当前状态为“k”时，遇文法符号A时状态应转向j，因此A 移入文法符号栈，j 移入状态栈。

d) 若项目S′→S·属于Ik，则置ACTION[k, #]为“acc”，表示接受。

e) 凡不能用上述方法填入的分析表的元素，均应填上“报错标志”。为了表的清晰我们令在表中用空白表示错误标志。

根据这种方法构造的分析表不含多重定义时，称这样的分析表为LR（0）分析表，能用LR（0）分析表的分析器称为LR（0）分析器，能构造LR（0）分析表的文法称为LR（0）文法。

例：对文法G′

（0）S′→S

（1）S→aAc

（2）A→Abb

（3）A→b

由识别可归前缀的DFA（状态转换矩阵）构造得LR（0）分析表如下：

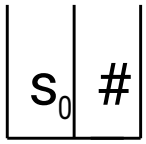
看状态转换矩阵

状态	ACTION				GoTo	
	a	b	c	#	S	A
⇒0	S_1				2	
1		S_3				4
2				acc		
3	r_3	r_3	r_3	r_3		
4		S_5	S_6			
5		S_7				
6	r_1	r_1	r_1	r_1		
7	r_2	r_2	r_2	r_2		

4、LR（0）分析器的工作过程

在分析的每一步，通用的总控程序按照状态栈顶状态 q 和当前输入符号 a 查阅LR（0）分析表，并执行其中 ACTION [q, a] 和 GOTO 部分规定的操作。

初始：分析栈　　输入串



α #

写成三元式形式：

状态栈	符号栈	输入串
S_0	#	α #

设分析到某一步，三元式是如下形式：

状态栈	符号栈	输入串
q0 q1 … qi	# x1 x2 …xi	ak … #

下一步的操作，根据当前栈顶状态 qi 和当前输入符号ak 查阅LR（0）分析表，执行其中 ACTION [qi, ak] 所规定的动作。

退出

（1）若ACTION [q_i, a_k] = S_j ，则将状态 S_j, a_k 进栈，三元式变化过程:

$q_0 q_1 \ldots q_i$	$\# x_1 x_2 \ldots x_i$	$a_k a_{k+1} \ldots a_n \#$	原来状况
q0 q1 … q i qj	# x1 x2 … x i ak	a k+1 … a n #	移进后

（2）若ACTION [q_i, a_k] = r_j ，且第 j 条产生式为 A → β ，|β| = r ，则按 第 j 条产生式归约。

设第 j 条产生式为 A → β ，|β|=m。

则三元式变化过程如下：

状态栈	符号栈	输入串	说明
$q_0 q_1 \ldots q_{i-m} q_{i-m+1} \ldots q_i$	$\# x_1 x_2 \ldots x_{i-m} x_{i-m+1} \ldots x_i$	$a_k a_{k+1} \ldots a_n \#$	原来的状况
$q_0 q_1 \ldots q_{i-m}$	$\# x_1 x_2 \ldots x_{i-m} A$	$a_k a_{k+1} \ldots a_n \#$	从栈中顶出m项A进栈
$q_0 q_1 \ldots q_{i-m} q_t$	$\# x_1 x_2 \ldots x_{i-m} A$	$a_k a_{k+1} \ldots a_n \#$	查GoTo[q_{i-m}, A]=q_t 将q_t进状态栈

（3）若ACTION[qi, ak]=acc 则结束分析，输入串被接受。

（4）若ACTION[qi, ak]=ERROR 或表中为空白，表示出错，进行相应出错处理。

退出

例：已知文法G′为：

（0）S′→S

（1）S→aAc

（2）A→Abb

（3）A→b

该文法的LR（0）分析表为：

状态	ACTION				GoTo	
	a	b	c	#	S	A
⇒0	S_1				2	
1		S_3				4
2				acc		
3	r_3	r_3	r_3	r_3		
4		S_5	S_6			
5		S_7				
6	r_1	r_1	r_1	r_1		
7	r_2	r_2	r_2	r_2		

要求分析输入串abbbc

解： 分析步骤：

状态栈	符号栈	输入串	ACTION	GoTo
0	#	abbbc#	S_1	
01	#a	bbbc#	S_3	
013	#ab	bbc#	r_3	4
014	#aA	bbc#	S_5	
0145	#aAb	bc#	S_7	
01457	#aAbb	c#	r_2	1
014	#aA	c#	S_6	
0146	#aAc	#	r1	2
02	#S	#	***acc***	

（3）A→b 归约

（2）A→Abb 归约

（1）S→aAc 归约

5、非LR（0）文法的判断

判断方法一：

考察识别文法可归前缀的DFA，若某个状态（即项目集）中既含移进项目又含归约项目；

或含不只一个归约项目，则会发生分析动作的冲突，可知该文法不是LR（0）的。

判断方法二：

若文法的LR（0）分析表中含多重定义，即表中某项动作不唯一，则该文法不是LR（0）文法。

6.3 SLR（1）分析

1、SLR（1）方法的引进

LR（0）方法实际上隐含了这样一个要求：构造出的识别可归前缀的有穷自动机的各个状态中不能有冲突项目，否则分析表将含有动作冲突。

设有一个状态 I： $x \rightarrow \alpha \cdot b\beta$

$A \rightarrow r \cdot$

$B \rightarrow \delta \cdot$ 含有冲突项目

在LR（0）分析表中，对任何终结符a（包括#）ACTION[I, a]的动作均为归约。

这样就造成移进与其它归约之间的冲突。

1、SLR（1）方法的引进

若对于A→r·改为只对FOLLOW（A）中的元素（设为a ），ACTION[I, a]为归约; 对于B→δ· 只对FOLLOW（B）中元素才归约, 如此处理后，

如果 FOLLOW（A）∩FOLLOW（B）=ϕ

FOLLOW（A）∩{b}=ϕ

FOLLOW（B）∩{b}=ϕ

则当状态 I 面临输入符号为b时，则分析动作可唯一确定。（移进）

用SLR（1）方法，对于当前状态中的归约项目，如A→α·，必须当前输入符号属于FOLLOW（A）时，才可做归约。有望解决LR（0）方法中的分析动作冲突问题。

退出

2、SLR（1）分析表的构造

将LR（0）分析表构造算法中的 b）改为：

若项目A→α·属于I_k，则对a为任何终结符或‘#’号，且满足a∈FOLLOW(A)时，置ACTION[k, a]为“rj”，j为在文法G′中某产生式A→α的序号。

其余均同LR（0）分析表的构造SLR（1）分析。

总控程序使用SLR（1）分析表进行分析。

3、非SLR文法的判断

判断方法一：

对识别文法可归前缀DFA中任一状态下，

设形式为：$A_1 \to \alpha_1 \cdot$　　必须：

$a_1\beta_1$　　$\{a_1, \ldots, a_m\}$

⋮　　FOLLOW(B_1)

$A_m \to \alpha_m \cdot a_m\beta_m$　　…

$B_1 \to r_1 \cdot$　　FOLLOW(B_n)

⋮

$B_n \to r_n \cdot$

两两不相交，

否则，文法不是SLR的。

判断方法二：

若构造的SLR分析表含多重定义，则文法不是SLR的。

例: P 140

6.4 LR（1）分析

1、LR（1）方法的引进

对某些文法，用SLR（1）方法仍解决不了分析动作的冲突问题，可采取以下措施：若某归约项目$A\rightarrow\alpha\cdot\in I$，当 I 为当前状态，面临当前输入符号**a** 时，只有**a** 是在 I 状态下A 的后继符号时 才用 $A\rightarrow\alpha$ 产生式归约，而不是对A的所有后继符号都可以归约。从而有望解决冲突。

2、构造以LR（1）项目集为状态的识别可归前缀的DFA

为了得知在 I 状态下，归约项目

$A \rightarrow \alpha \cdot$

的 A 的后继符号是哪些，在LR（0）项目的后面加上向前搜索符，称为LR（1）项目。

如： $A \rightarrow \alpha \cdot$，b

构造文法的LR（1）项目集规范族和GO 函数（即DFA）

初态：CLOSURE（S′→S，#）

构造 CLOSURE（I）的方法:

（1）I 的任何项目均属于CLOSURE（I）；

（2）如果项目 A → α·Bβ，a 属于CLOSURE（I），

且 B → r 是文法中的产生式，b∈FIRST(βa)

则 B → ·r，b 也属于CLOSURE（I）；

（3）重复（2），直至CLOSURE（I）不再增大为止。

构造GO 函数方法与LR（0）的相似，向前搜索符无变化。

退出

将LR（0）分析表构造算法中的 b) 中：

若项目 $A \to \alpha \cdot$ 属于I_k，则对 a 为任何终结符或‘#’号，置ACTION[k, a]为 r_j

改为：若项目 $[A \to \alpha \cdot, a]$ 属于I_k，

则置 ACTION [k,a] = r_j

4、LR（k）分析表

如果用 LR（1）方法仍不能解决冲突，则可再向前多搜索几个符号，这时的项目为 $[A \to \alpha \cdot \beta, a_1 a_2 \ldots a_k]$ 称为 LR（k）项目，相应的分析表构造方法类似 LR（1）分析表的构造。

6.5 LALR（1）分析

在LR（1）项目中，有很多状态中的项目除了向前搜索符号不同外，产生式部分是完全相同的，称这样的状态是同心的，为了克服LR（1）分析中状态太多的问题，可以将这些同心集合并。如果合并后得到的新状态没有冲突出现。则按新状态构造分析表。

这就是LALR（1）分析法的基本思想。

- 注意：语法分析器的自动生成器YACC就是使用LALR（1）分析表进行分析的。

本章要点：

可归前缀
LR（0）项目
构造识别文法可归前缀的活前缀的DFA
（即求构造LR（0）项目集规范族和Go函数）
构造LR（0）分析表
非LR（0）文法的判断
LR（0）分析
构造SLR（1）分析表
SLR（1）分析
非SLR文法的判断
LR（1）分析