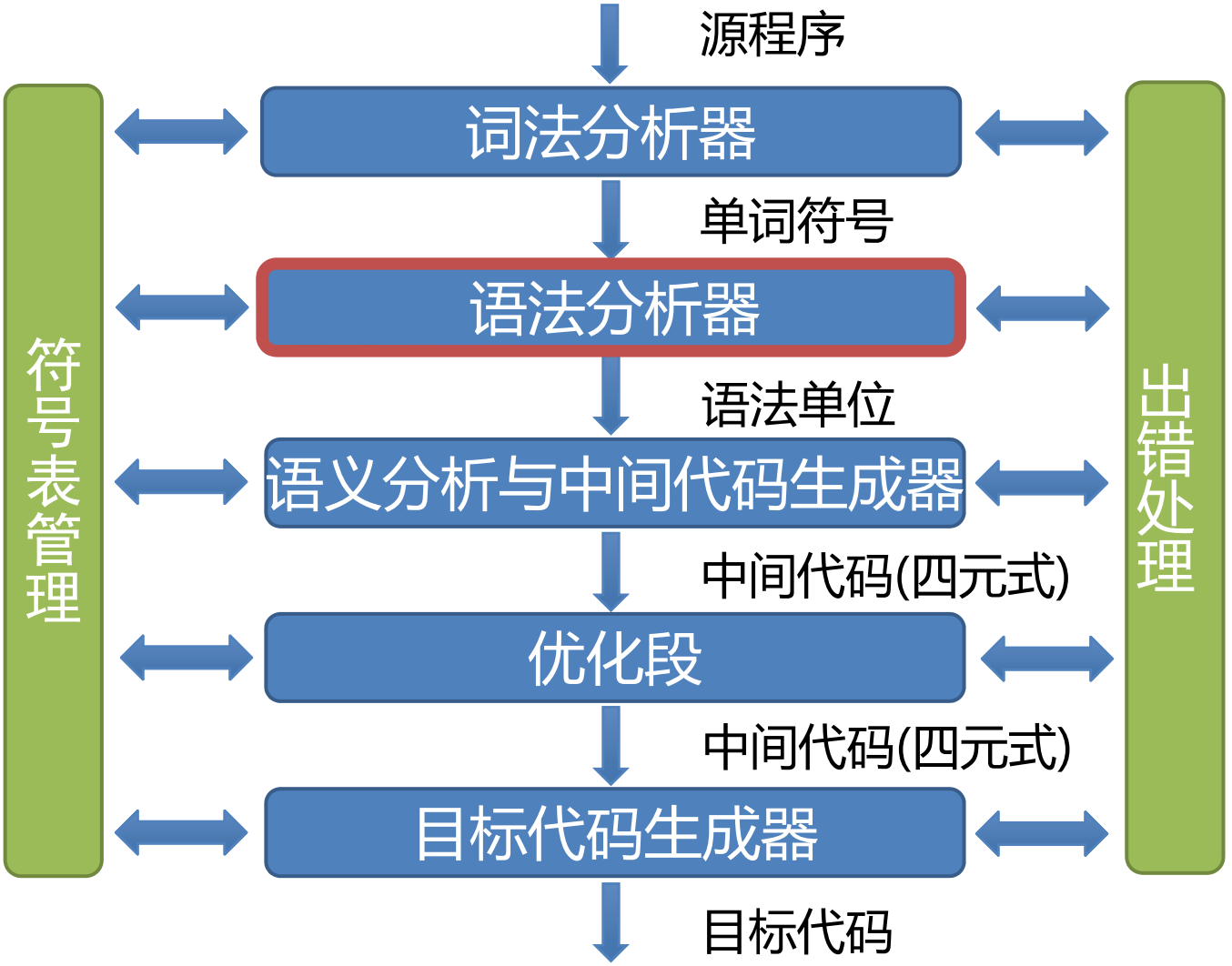
编译原理

LR(0)分析表的构造

编译原理

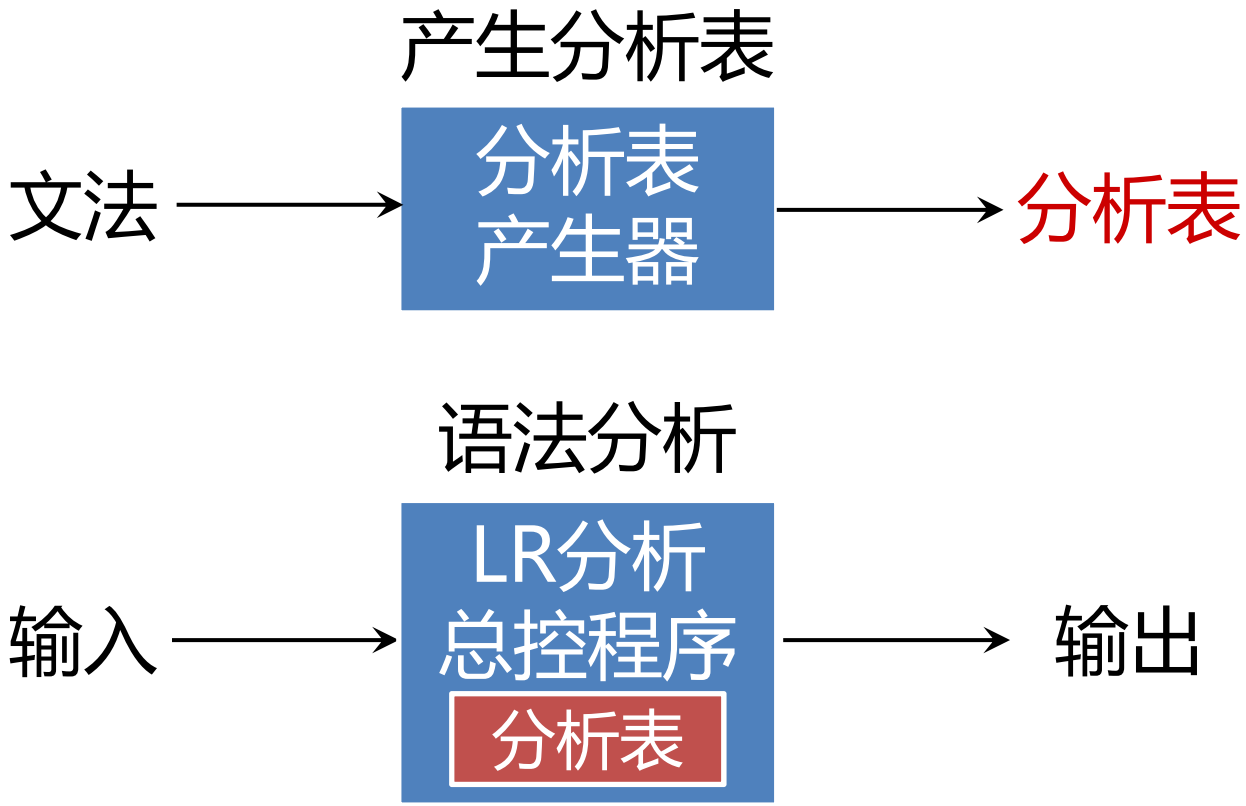
LR分析法回顾

编译程序总框


源程序
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
中间代码(四元式)
优化段
中间代码(四元式)
目标代码生成器
目标代码
符号表管理
出错处理

LR分析法

- 工作框架



规范归约

- 定义：假定α是文法G的一个句子，我们称序列α_n，α_{n-1}，…，α_0 是α的一个规范归约，如果此序列满足：

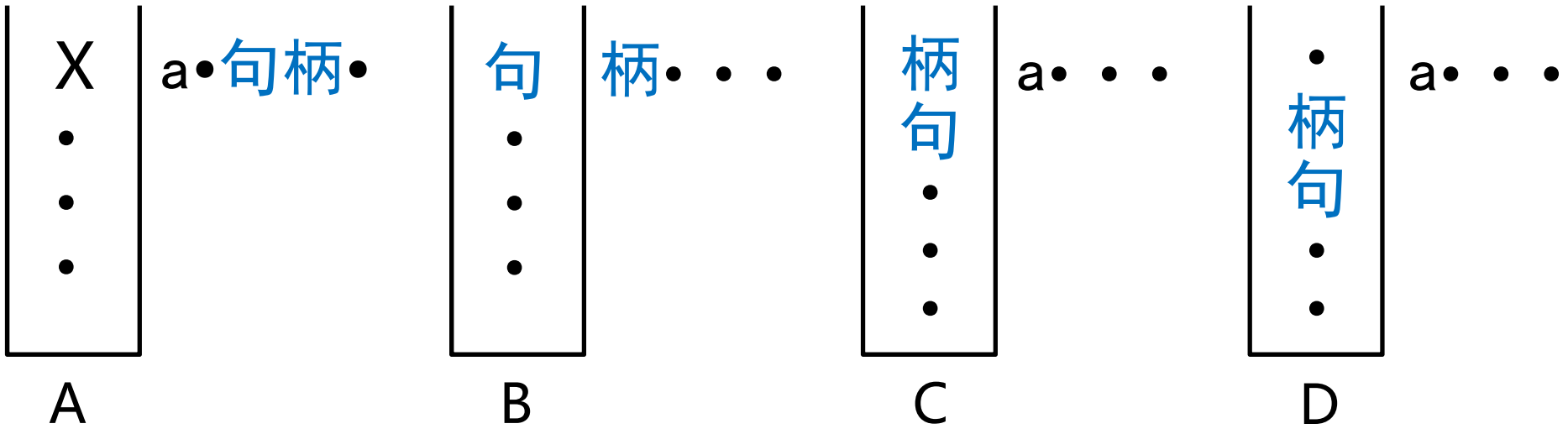
1. $\alpha_n = \alpha$
2. α_0为文法的开始符号，即$\alpha_0 = S$
3. 对任何i，$0 \le i \le n$，α_{i-1}是从α_i经把句柄替换成为相应产生式左部符号而得到的

LR分析法

- LR分析器的性质
  - 栈内的符号串和扫描剩下的输入符号串构成了一个规范句型
  - 一旦栈的顶部出现可归约串(句柄)，则进行归约

测试：规范归约过程中栈内符号串

- 对于句子，在规范归约过程中，栈内的符号串和扫描剩下的输入符号串构成了一个规范句型，下面哪种格局不会出现：

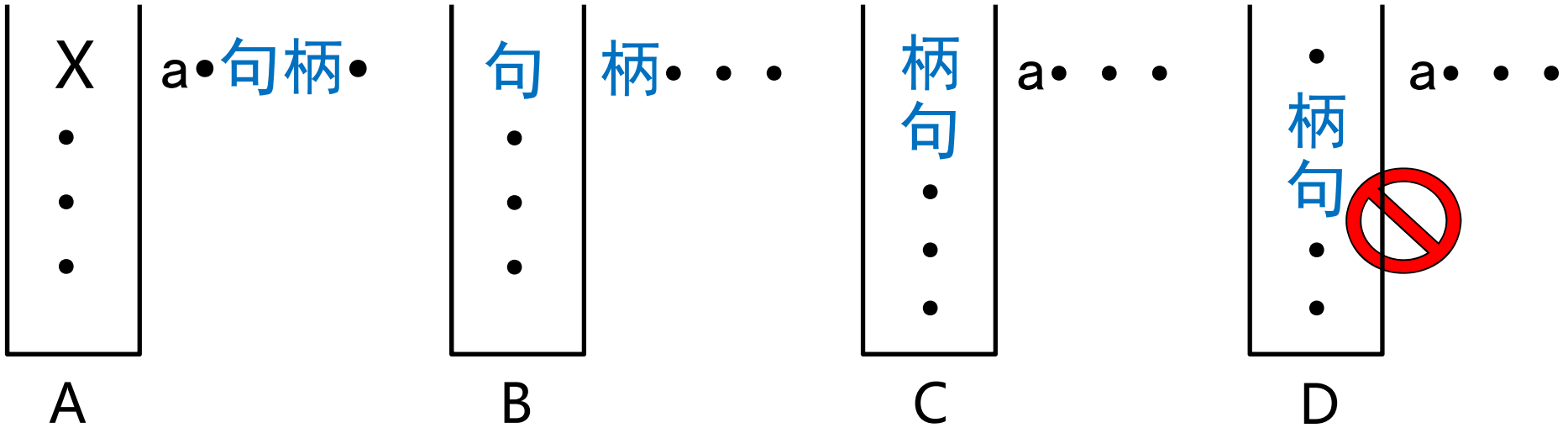


编译原理

活前缀

规范归约过程中栈内符号串

- 规范归约过程中
  - 栈内的符号串和扫描剩下的输入符号串构成了一个规范句型
  - 栈内的如果出现句柄，句柄一定在栈的顶部
  - 栈内永远不会出现句柄之后的符号



字的前缀、活前缀

- 字的前缀：是指字的任意首部，如字*abc*的前缀有ε，*a*，*ab*，*abc*
- 活前缀：是指规范句型的一个前缀，这种前缀不含句柄之后的任何符号。即，对于规范句型αβδ，β为句柄，如果$\alpha\beta=u_1u_2...u_r$，则符号串$u_1u_2...u_i(1\le i\le r)$是αβδ的活前缀。(δ必为终结符串)
- 规范归约过程中，保证分析栈中总是活前缀，就说明分析采取的移进/归约动作是正确的

识别活前缀

- 能否判断一个符号串是不是活前缀?
- 对于一个文法G, 可以构造一个DFA,它能识别G的所有活前缀。

编译原理

构造识别活前缀的DFA

文法的拓广

- 将文法G(S)拓广为G′(S′)
  - 构造文法G′，它包含了整个G，并引进不出现在G中的非终结符S′、以及产生式S′→S，S′是G′的开始符号
  - 称G′是G的拓广文法

LR(0)项目

- LR(0)项目
  - 在每个产生式的右部添加一个圆点
  - 表示我们在分析过程中看到了产生式多大部分
- A→XYZ有四个项目
  - A→ •XYZ　A→X•YZ　A→XY•Z　A→XYZ•
- A→α• 称为"归约项目"
- 归约项目 S′→α • 称为"接受项目"
- A→α•aβ (a∈V_T) 称为"移进项目"
- A→α•Bβ (B∈V_N) 称为"待约项目"

示例：LR(0)项目

- 文法G(S′)
  S′→E
  E→aA|bB
  A→cA|d
  B→cB|d
- 该文法的项目有：

1. S′→•E　2. S′→E•
3. E→•aA　4. E→a•A　5. E→aA•
6. A→•cA　7. A→c•A　8. A→cA•　9. A→•d　10. A→d•
11. E→•bB　12. E→b•B　13. E→bB•
14. B→•cB　15. B→c•B　16. B→cB•　17. B→•d　18. B→d•

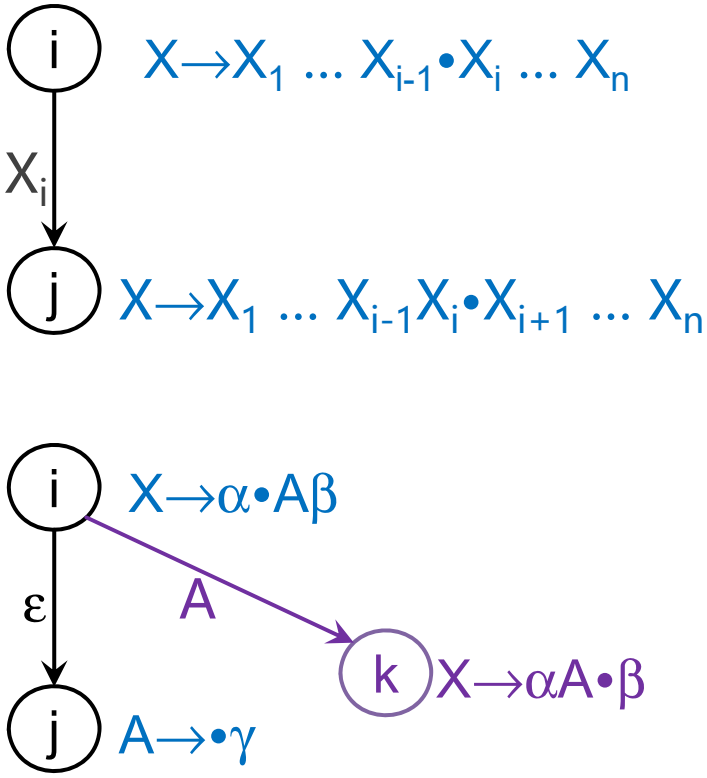
构造识别文法所有活前缀的DFA

- 构造识别文法所有活前缀的NFA
  - 若状态i为$X\rightarrow X_1 \dots X_{i-1}\bullet X_i \dots X_n$，

    状态j为$X\rightarrow X_1 \dots X_{i-1}X_i\bullet X_{i+1} \dots X_n$，

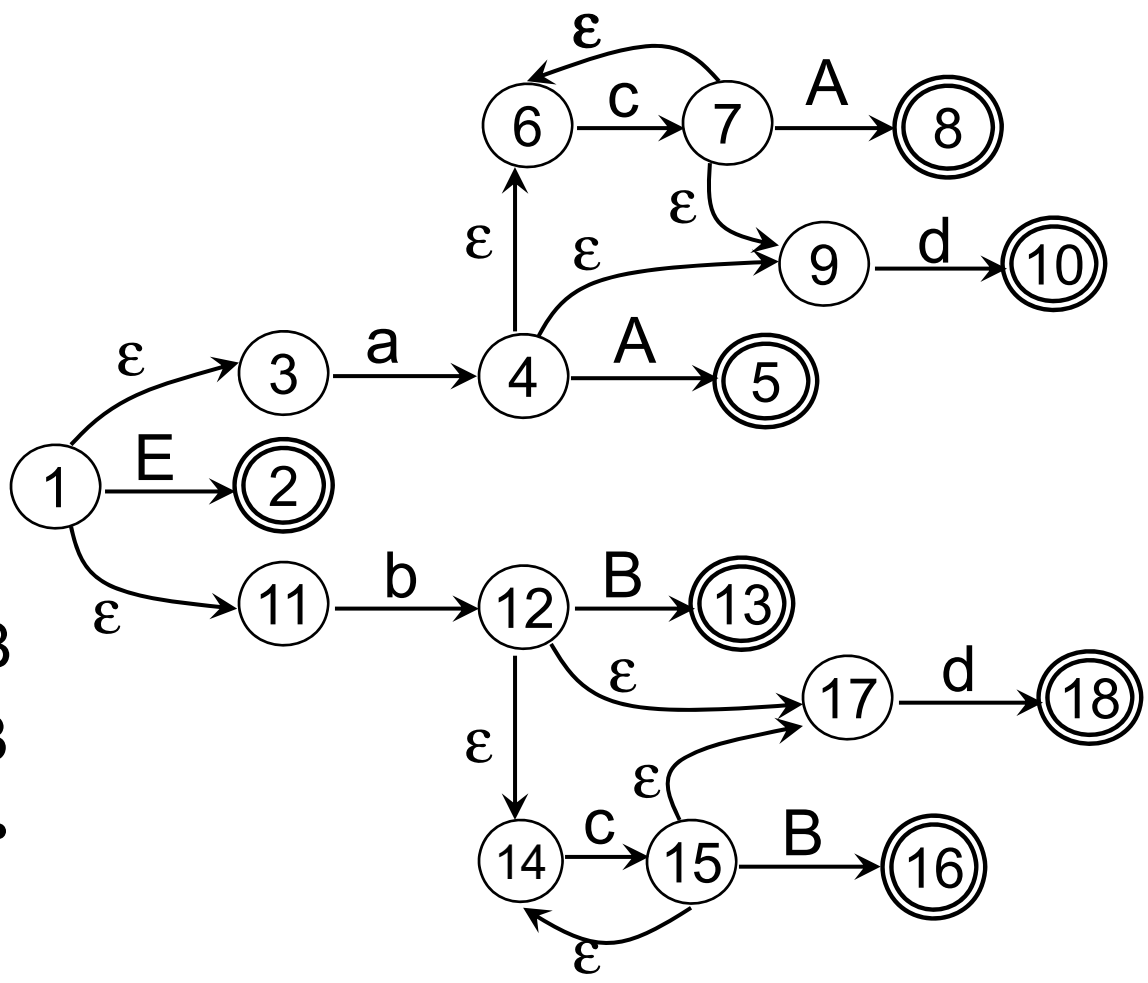
    则从状态i画一条标志为X_i的有向边到状态j；
  - 若状态i为$X\rightarrow\alpha\bullet A\beta$，A为非终结符，

    则从状态i画一条ε边到所有状态$A\rightarrow\bullet\gamma$
- 把识别文法所有活前缀的NFA确定化



识别活前缀的NFA

1. S′→•E　　2. S′→E•
3. E→•aA　　4. E→a•A
5. E→aA•　　6. A→•cA
7. A→c•A　　8. A→cA•
9. A→•d　　10. A→d•
11. E→•bB　　12. E→b•B
13. E→bB•　　14. B→•cB
15. B→c•B　　16. B→cB•
17. B→•d　　18. B→d•

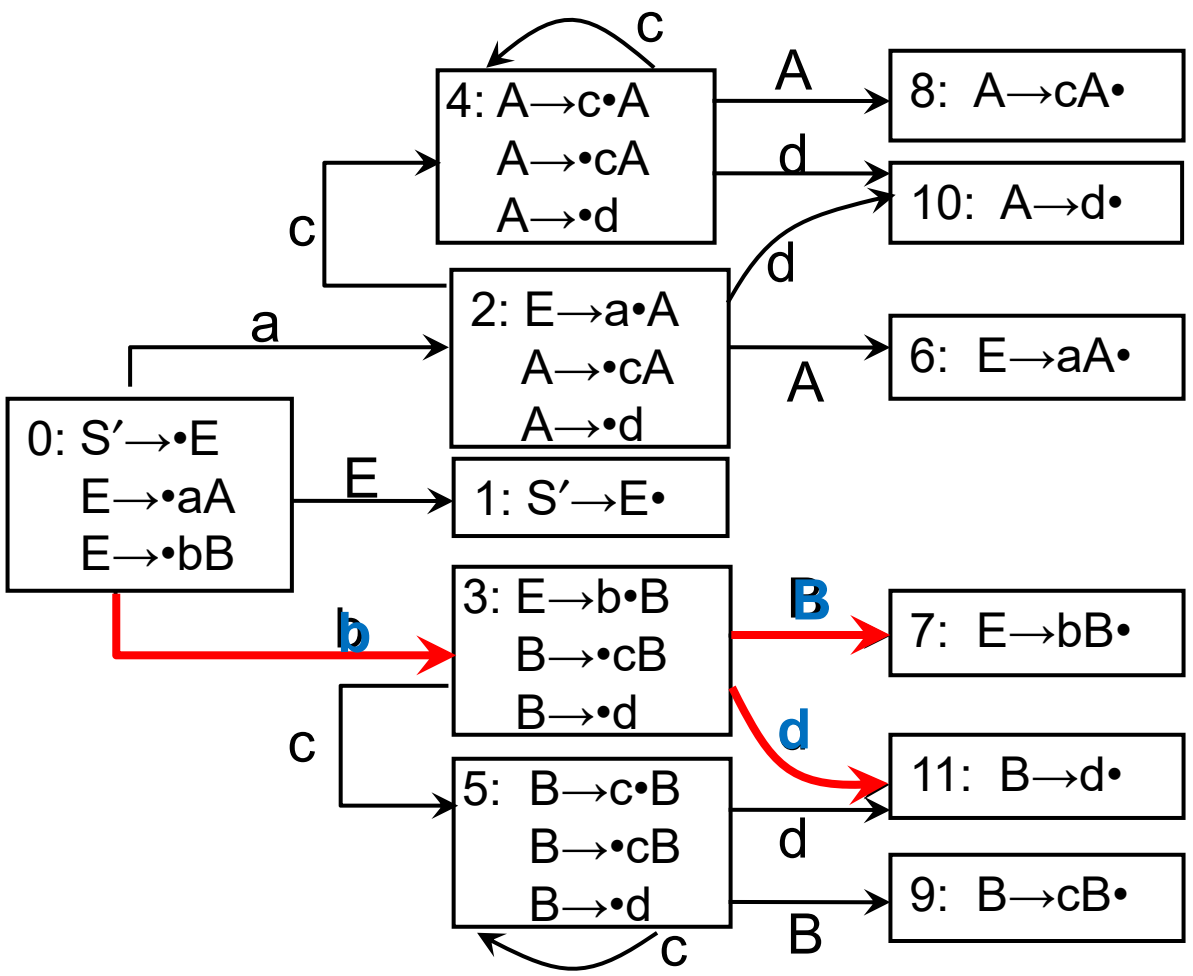


识别活前缀的DFA

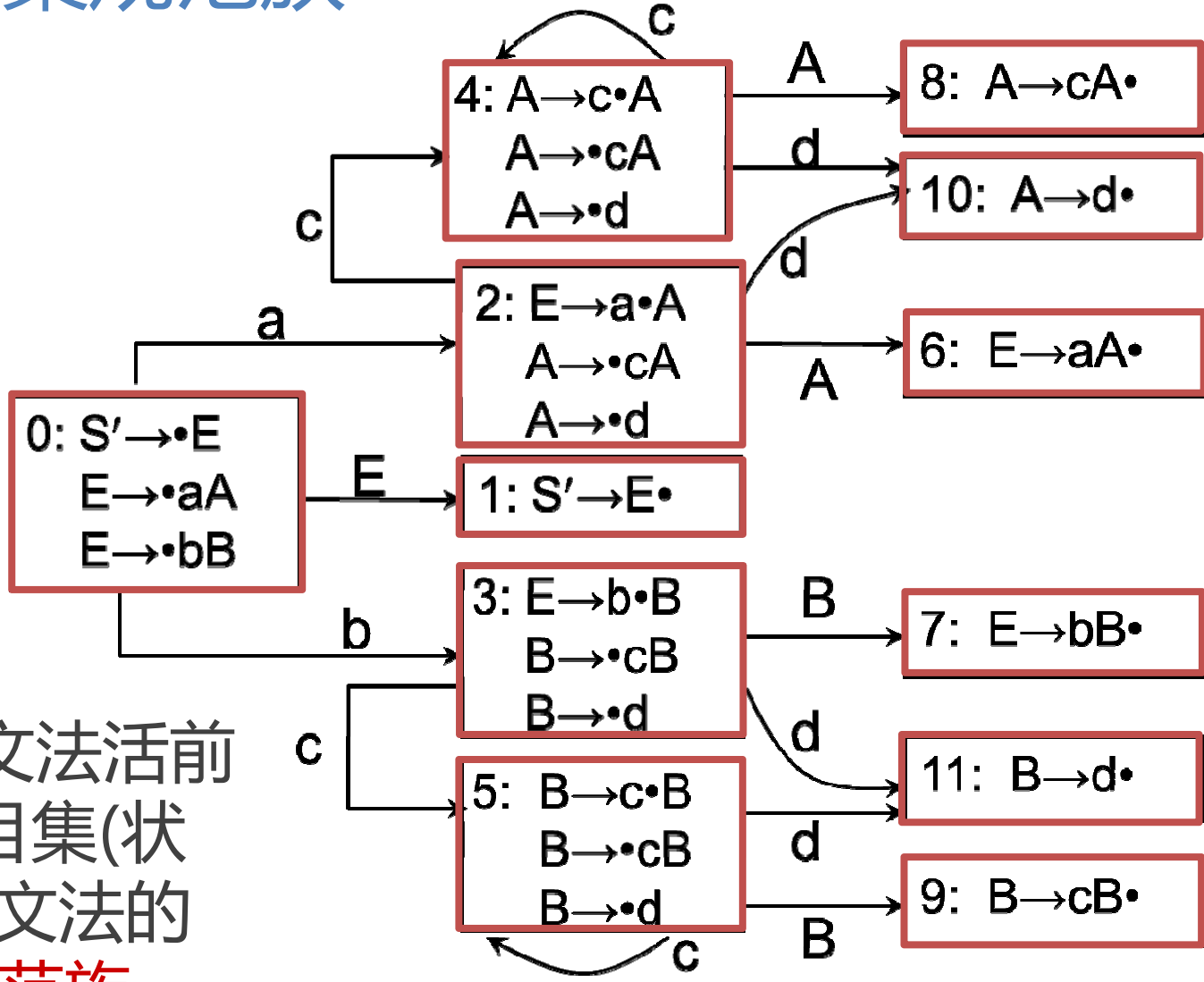
7	B
3	b
0	#

状态　符号
分析栈

b d #
输入串



LR(0)项目集规范族



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

编译原理

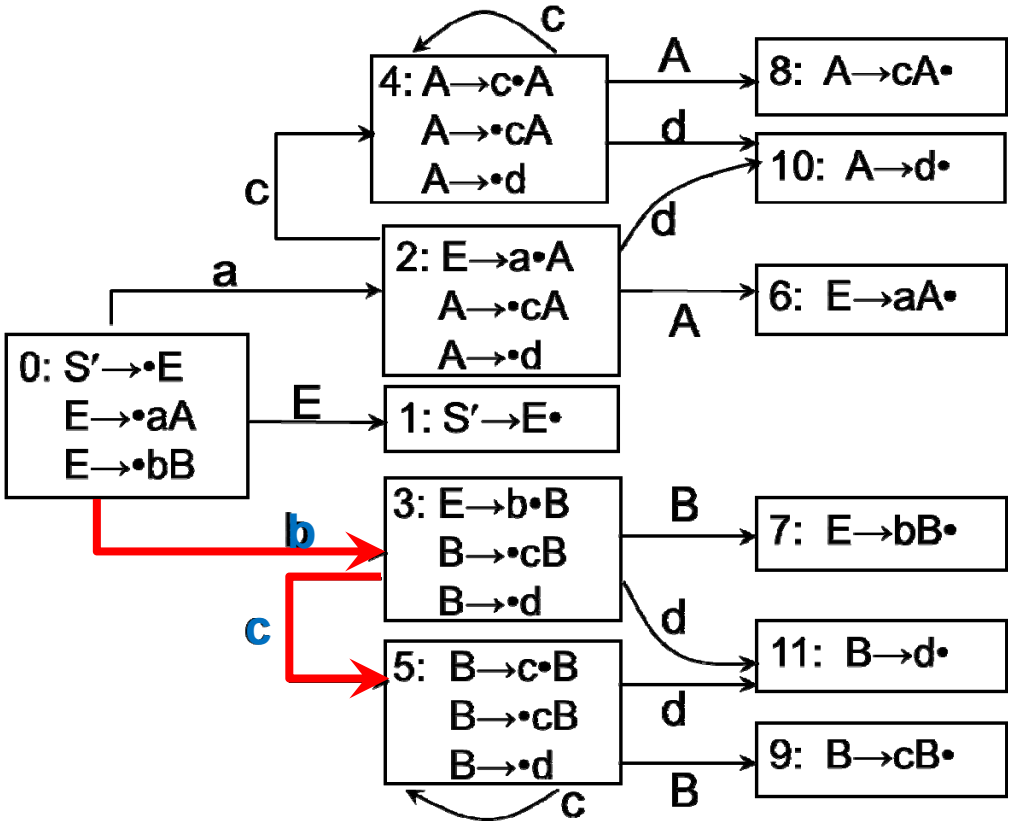
通过计算项目集规范族构造识别活前缀的DFA

有效项目

- 项目A→ $\beta_1 \bullet \beta_2$对活前缀$\alpha\beta_1$是有效的，其条件是存在规范推导

$$S' \overset{*}{\Rightarrow}_R \alpha A \omega \Rightarrow_R \alpha\beta_1\beta_2\omega$$

- 在任何时候，分析栈中的活前缀$X_1X_2 \ldots X_m$的有效项目集正是从识别活前缀的DFA的初态出发，读出$X_1X_2 \ldots X_m$后到达的那个项目集(状态)。



有效项目的性质

项目A→ β_1•β_2对活前缀$\alpha\beta_1$是有效的，其条件是存在规范推导：

$$S' \overset{*}{\Rightarrow}_R \alpha A\omega \Rightarrow_R \alpha\beta_1\beta_2\omega$$

- 若项目A→α•Bβ对活前缀η=δα是有效的且B→γ 是一个产生式，则项目B → •γ 对η=δα 也是有效的。

证明：

若项目A→α•Bβ对活前缀η=δα是有效的，则有

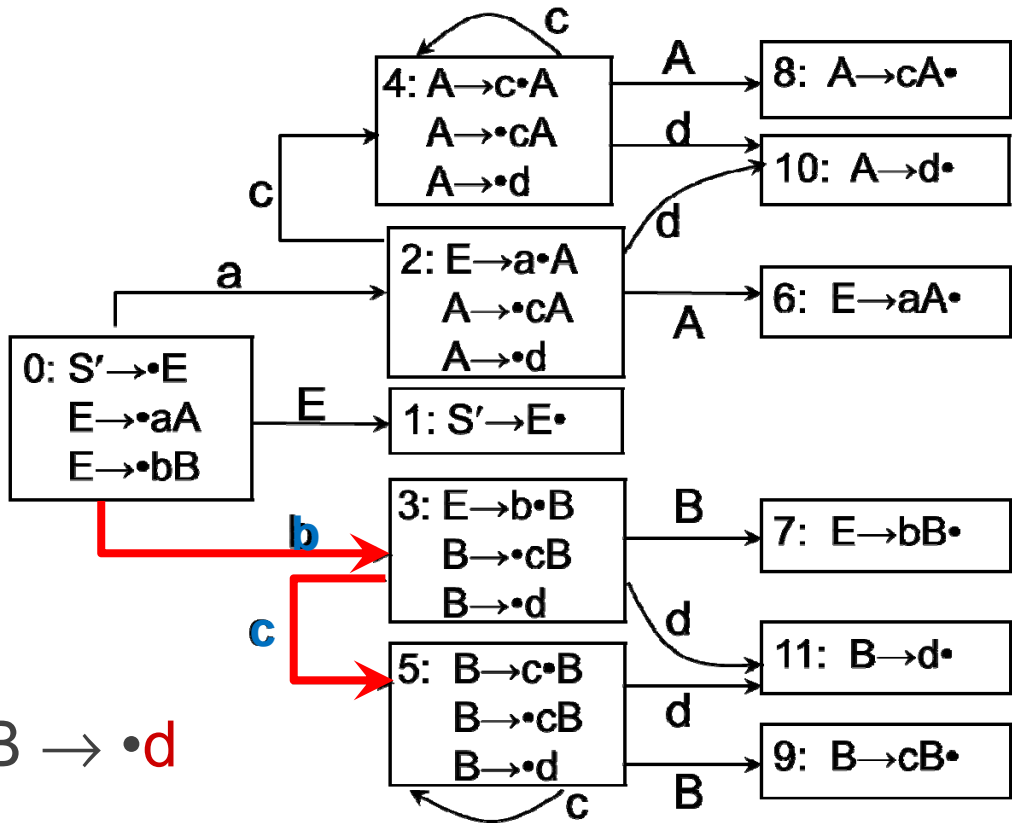
$$S' \overset{*}{\Rightarrow}_R \delta A\omega \Rightarrow_R \delta\alpha B\beta\omega$$

设$\beta\omega \overset{*}{\Rightarrow}_R \varphi\omega$, 那么

$$S' \overset{*}{\Rightarrow}_R \delta A\omega \Rightarrow_R \delta\alpha B\beta\omega \overset{*}{\Rightarrow}_R \delta\alpha B\varphi\omega \Rightarrow_R \delta\alpha\gamma\varphi\omega$$

所以，B → •γ 对 η=δα 也是有效的

有效项目的性质



- 文法G(S′)

  S′→E

  E→aA|bB

  A→cA|d

  B→cB|d
- 项目B → c•B，B → •cB，B → •d
- 对于活前缀：bc 有效

  S’⇒ E ⇒ bB ⇒ bcB

  S’⇒ E ⇒ bB ⇒ bcB ⇒bccB

  S’⇒ E ⇒ bB ⇒ bcB ⇒ bcd

项目A→ β_1•β_2对活前缀$\alpha\beta_1$是有效的，其条件是存在规范推导：

$$S' \overset{*}{\Rightarrow}_R \alpha A\omega \Rightarrow_R \alpha\beta_1\beta_2\omega$$

LR(0)项目集规范族的构造

- 将文法G(S)拓广为G′(S′)
  - 构造文法G′，它包含了整个G，并引进不出现在G中的非终结符S′、以及产生式S′→S，S′是G′的开始符号
  - G′唯一的“接受”态：仅含项目S′→S•的状态

项目集的闭包CLOSURE

构造

- 构造
  - 若
    状
    则
  - 若
    则
- 把识

$X_{i-1} \bullet X_i \ldots X_n$

$X_{i-1} X_i \bullet X_{i+1} \ldots X_n$

DFA与NFA的等价性证明

- NFA确定化--子集法 (解决ε弧和转换关系)
- 设I是的状态集的一个子集，定义I的ε-闭包ε-closure(I)为:
  - 若s∈I，则s∈ε-closure(I)；
  - 若s∈I，则从s出发经过任意条ε弧而能到达的任何状态s’都属于ε-closure(I)

即，

ε-closure(I)=I∪{s’ |从某个s∈I出发经过任意条ε弧能到达s’ }

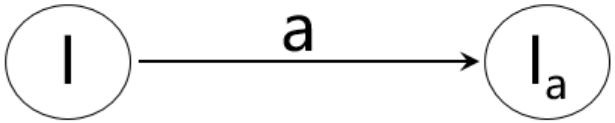
状态转换函数

DFA与NFA的等价性证明

- 设a是Σ中的一个字符，定义

$$I_a = \varepsilon\text{-closure}(J)$$

其中，J为I中的某个状态出发经过一条a弧而到达的状态集合。



态转换函数
项目集，X是
义为：
)

α•Xβ属于I}。

有效的项目集，

那么，GO(I，X)便是对X有效的项目集。

示例：项目集的转移函数计算

- 文法G(S′)

  S′→E

  E→aA|bB

  A→cA|d

  B→cB|d

- I_0={S′→•E, E→•aA, E→•bB}

GO(I_0, E)= closure(J)=closure({S’→E•})

= {S’→E•} = I_1

GO(I_0, a)= closure(J)=closure({E→a•A})

={ E→a•A, A→•cA, A→•d})=I_2

GO(I_0, b)= closure(J)=closure ({E →b•B})

={E→b•B, B→•cB, B→•d}= I_3

- 项目集I的闭包CLOSURE(I)：

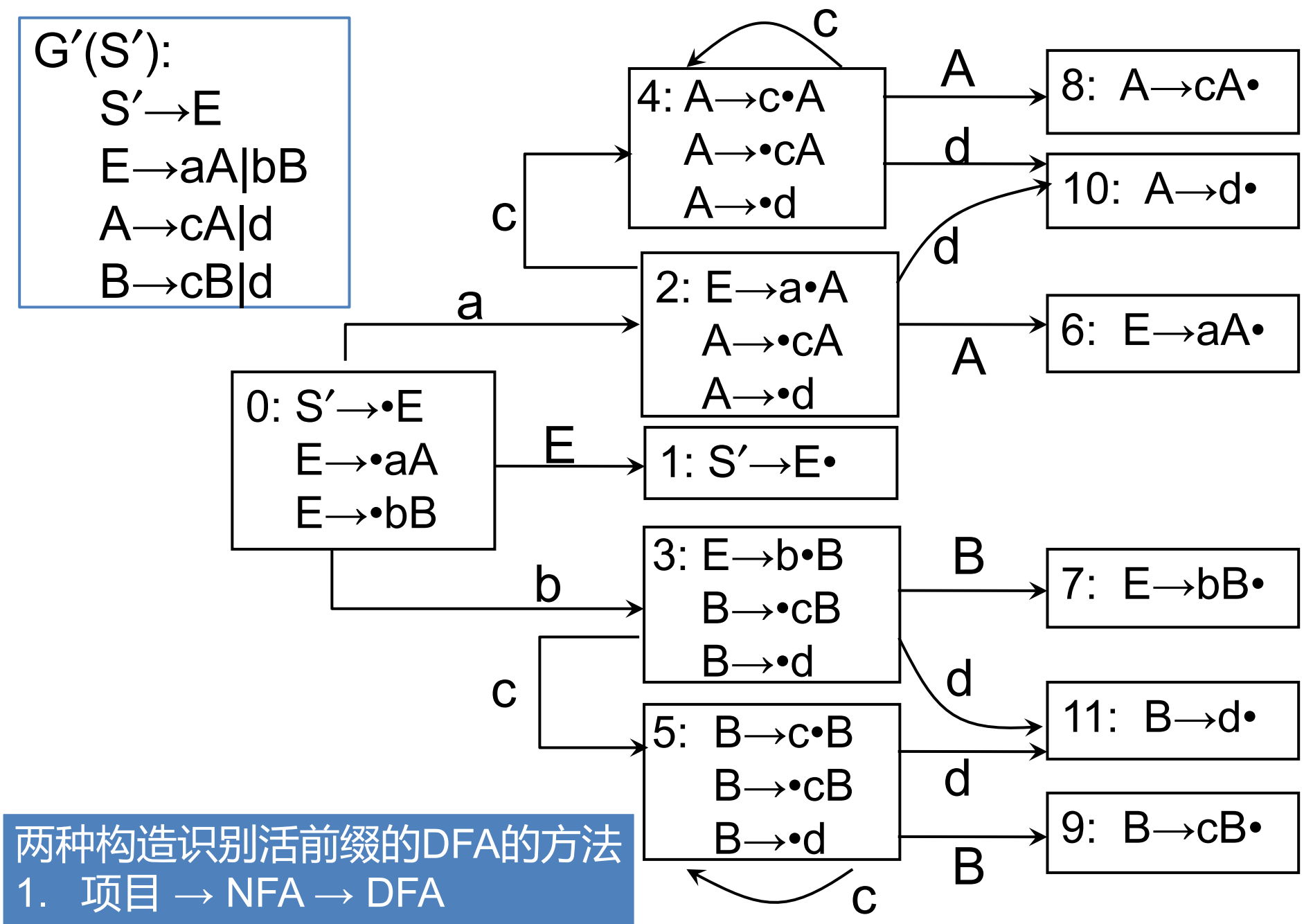
1. I的任何项目都属于CLOSURE(I)；
2. 若A→α•Bβ属于CLOSURE(I)，那么，对任何关于A的产生式B→γ，项目B→•γ也属于CLOSURE(I)；
3. 重复执行上述两步骤直至CLOSURE(I) 不再增大为止。

GO(I，X) = CLOSURE(J)

J = {任何形如A→αX•β的项目| A→α•Xβ属于I}。

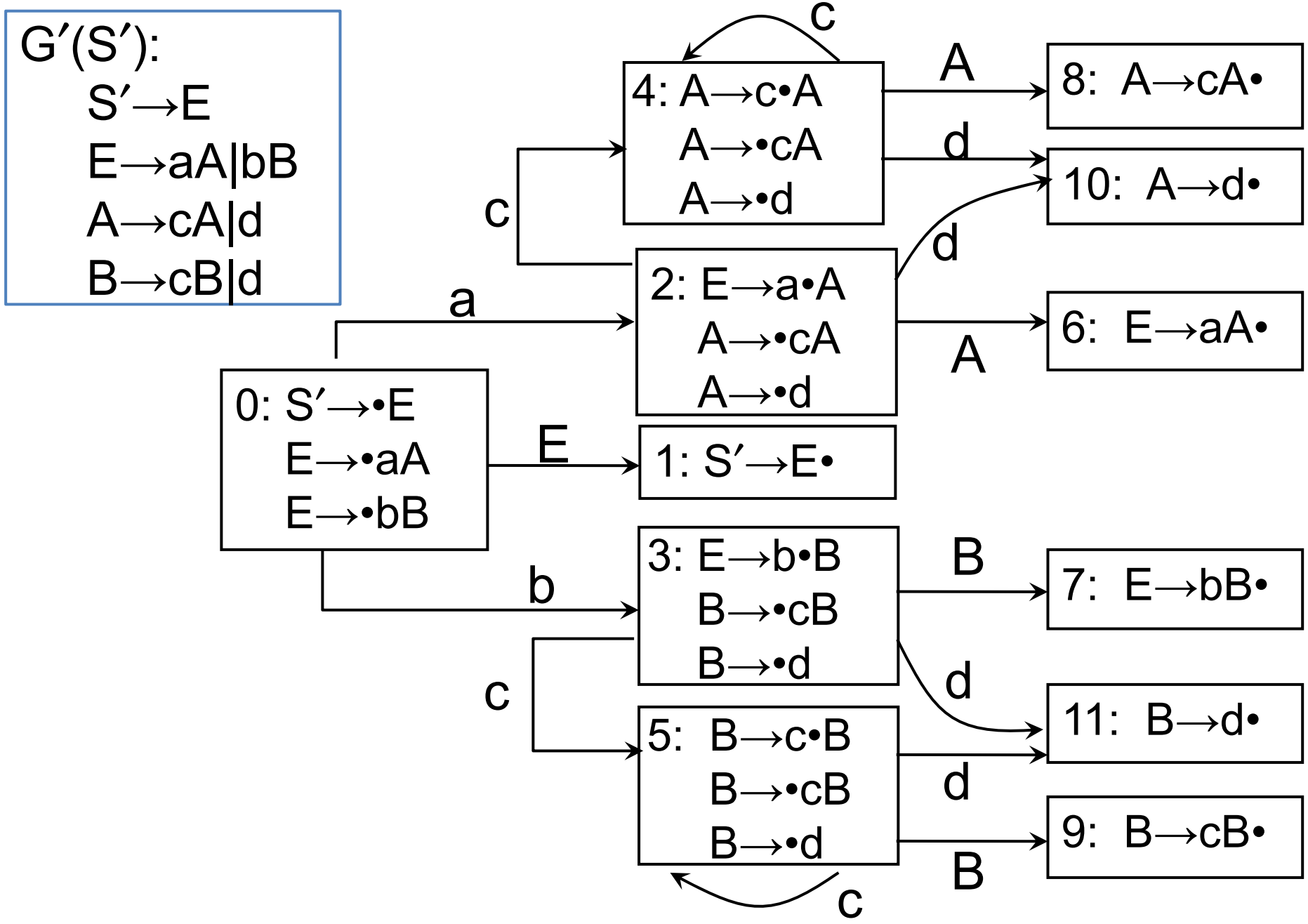
LR(0)项目集规范族的构造算法

```
PROCEDURE  ITEMSETS(G');
BEGIN
 C:={CLOSURE({S'→•S})};
 REPEAT
     FOR  C中每个项目集I和G'的每个符号X  DO
         IF  GO(I，X)非空且不属于C   THEN
              把GO(I，X)放入C族中;
 UNTIL C   不再增大
END
```


G′(S′):
S′→E
E→aA|bB
A→cA|d
B→cB|d
0: S′→•E
E→•aA
E→•bB
1: S′→E•
2: E→a•A
A→•cA
A→•d
3: E→b•B
B→•cB
B→•d
4: A→c•A
A→•cA
A→•d
5: B→c•B
B→•cB
B→•d
6: E→aA•
7: E→bB•
8: A→cA•
9: B→cB•
10: A→d•
11: B→d•
a
b
c
E
A
B
d
两种构造识别活前缀的DFA的方法
1. 项目 → NFA → DFA
2. Closure → GO → DFA

编译原理

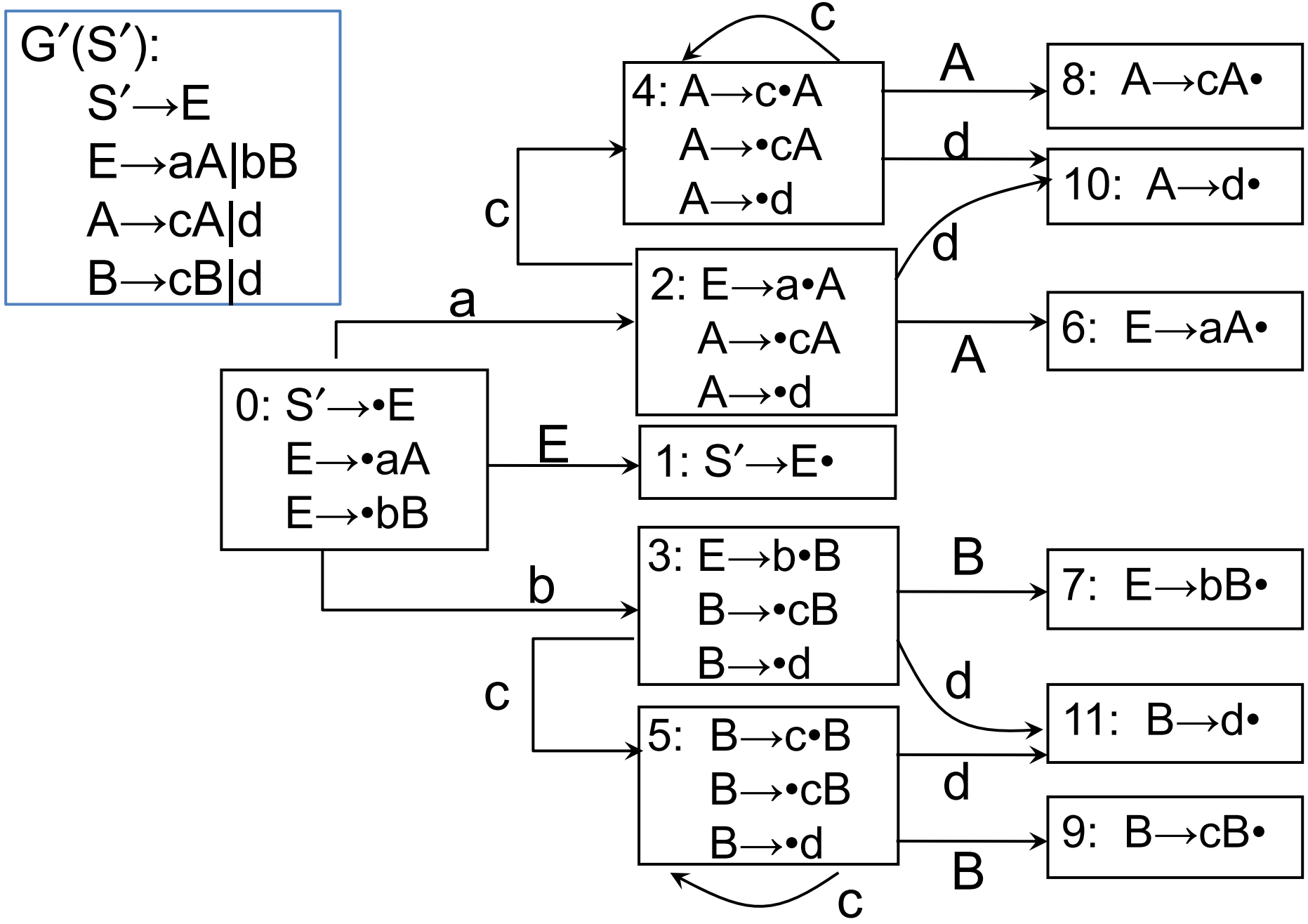
构造LR(0)分析表的算法


G′(S′):
S′→E
E→aA|bB
A→cA|d
B→cB|d
0: S′→•E
E→•aA
E→•bB
1: S′→E•
2: E→a•A
A→•cA
A→•d
3: E→b•B
B→•cB
B→•d
4: A→c•A
A→•cA
A→•d
5: B→c•B
B→•cB
B→•d
6: E→aA•
7: E→bB•
8: A→cA•
9: B→cB•
10: A→d•
11: B→d•
a
b
c
c
c
c
E
A
A
B
B
d
d
d
d

LR(0)分析表的构造

- 假若一个文法G的拓广文法G′的活前缀识别自动机中的每个状态(项目集)不存在下述情况：
  - 既含移进项目又含归约项目；
  - 含有多个归约项目

  则称G是一个LR(0)文法。


G′(S′):
S′→E
E→aA|bB
A→cA|d
B→cB|d
0: S′→•E
E→•aA
E→•bB
1: S′→E•
2: E→a•A
A→•cA
A→•d
3: E→b•B
B→•cB
B→•d
4: A→c•A
A→•cA
A→•d
5: B→c•B
B→•cB
B→•d
6: E→aA•
7: E→bB•
8: A→cA•
9: B→cB•
10: A→d•
11: B→d•
a
b
c
c
c
c
E
A
A
B
B
d
d
d
d

构造LR(0)分析表的算法

- 令每个项目集I_k的下标k作为分析器的状态，包含项目S′→•S的集合I_k的下标k为分析器的初态。
- 构造LR(0)分析表的ACTION和GOTO子表

LR(0)分析表的ACTION和GOTO子表构造

1. 若项目$A\rightarrow\alpha\bullet a\beta$属于$I_k$且$GO(I_k, a) = I_j$，a为终结符，则置ACTION[k, a] 为“sj”。
2. 若项目$A\rightarrow\alpha\bullet$属于$I_k$，那么，对任何终结符a(或结束符#)，置ACTION[k, a]为 “rj”(假定产生式$A\rightarrow\alpha$是文法G′的第j个产生式)。
3. 若项目$S'\rightarrow S\bullet$属于I_k，则置ACTION[k,#]为“acc”。
4. 若$GO(I_k, A) = I_j$，A为非终结符，则置GOTO[k, A]=j。
5. 分析表中凡不能用规则1至4填入信息的空白格均置上“报错标志”。

示例：LR(0)分析表的构造

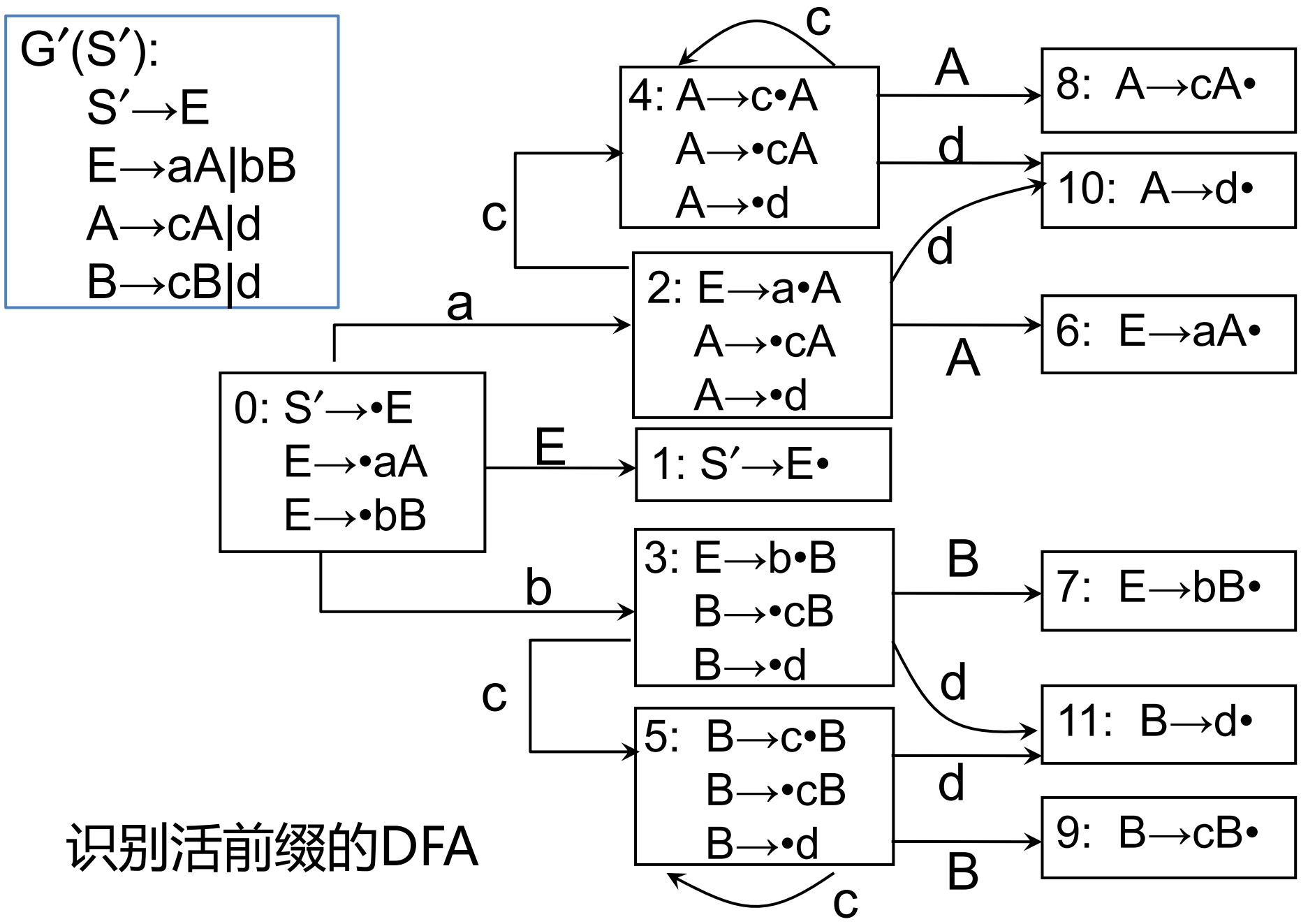
- 文法G(S′)：

  S′→E

  E→aA|bB

  A→cA|d

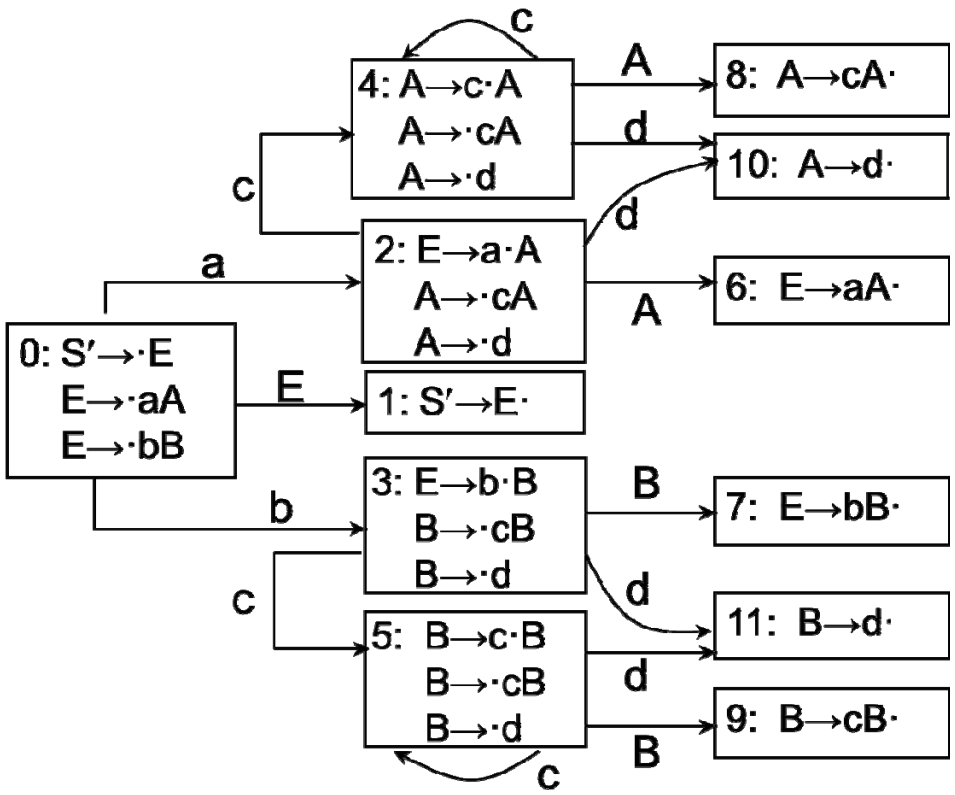
  B→cB|d


G′(S′):
S′→E
E→aA|bB
A→cA|d
B→cB|d
0: S′→•E
E→•aA
E→•bB
1: S′→E•
2: E→a•A
A→•cA
A→•d
3: E→b•B
B→•cB
B→•d
4: A→c•A
A→•cA
A→•d
5: B→c•B
B→•cB
B→•d
6: E→aA•
7: E→bB•
8: A→cA•
9: B→cB•
10: A→d•
11: B→d•
a
E
b
c
A
d
B
识别活前缀的DFA

示例：LR(0)分析表的构造

0:S′→E
1:E→aA
2:E→ bB
3:A→cA
4:A→ d
5:B→cB
6:B→ d

	ACTION					GOTO		
状态	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			



编译原理

LR(0)分析示例

LR(0)分析示例

- 按照LR(0)分析表对bcd进行分析

步骤	状态	符号	输入串
1	0	#	bcd#
2	03	#b	cd#
3	035	#bc	d#

0:S′→E
1:E→aA
2:E→ bB
3:A→cA
4:A→ d
5:B→cB
6:B→ d

	ACTION					GOTO		
状态	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			

LR(0)分析示例

- 按照LR(0)分析表对bcd进行分析
- 步骤 状态 符号 输入串

步骤	状态	符号	输入串
3	035	#bc	d#
4	03511	#bcd	#
5	0359	#bcB	#

0:S′→E
1:E→aA
2:E→ bB
3:A→cA
4:A→ d
5:B→cB
6:B→ d

	ACTION					GOTO		
状态	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			

LR(0)分析示例

- 按照LR(0)分析表对bcd进行分析
- 步骤 状态 符号 输入串

步骤	状态	符号	输入串
5	0359	#bcB	#
6	037	#bB	#
7	01	#E	#
8	接受		

0:S′→E
1:E→aA
2:E→ bB
3:A→cA
4:A→ d
5:B→cB
6:B→ d

	ACTION					GOTO		
状态	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			

小结

- 规范归约过程中，只要保证分析栈中总是活前缀，就说明分析采取的移进/归约动作是正确的
- 哪些字符串是活前缀?能不能构造一个DFA来识别活前缀?
  - 项目 ->NFA ->DFA
  - Closure->GO->DFA
- 将识别活前缀的DFA转换成LR分析表