中山大學
SUN YAT-SEN UNIVERSITY

国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS290

Compilation Principle

编 译 原 理

第四章 语法分析（5）

郑馥丹
zhengfd5@mail.sysu.edu.cn

CONTENTS

目 录

1. LR分析概述

- 由Donald Ervin Knuth（高德纳）提出
  - 《计算机程序设计的艺术》[The Art of Computer Programming]系列
  - 《计算机与排版》（Computers and Typesetting）
  - 提出“算法”和“数据结构”概念
  - 提出LR分析法
  - 提出双向链表
  - 提出KMP算法(Knuth-Morris-Pratt)
  - 提出Tex排版系统
  - 算法大师Knuth如何看待GPT?
  - 最年轻的图灵奖获得者记录的保持者（36岁）

1. LR分析概述

- 自底向上分析法的关键问题是在分析过程中**如何确定句柄**
- LR分析法：根据当前符号栈中的符号串和向前顺序查看输入串的k个符号（k>=0）就可唯一地确定句柄
- 目前最流行的自底向上语法分析器都是基于LR(k)语法分析，其中
  - **L**表示**从左到右**扫描输入串
  - **R**表示最左规约（即**最右推导**的逆过程）
  - **k**表示**向前查看输入串符号的个数**
    - ✓ 当k=1时，能满足当前绝大多数高级语言编译程序的需要，所以着重介绍LR(0), SLR(1), LR(1), LALR(1)方法
    - ✓ 省略(k)时，一般指k=1

1. LR分析概述

- LR分析的特点
  - 是**规范归约**（**最右推导/规范推导**的逆过程）
  - **适用范围广**，适用于大多数上下文无关文法描述的语言（比LL(k)分析方法和算符优先分析方法对文法的限制要少得多）
    - ✓ LR(k)：只要在一个最右句型中看到某个产生式右部，再向前看k个符号就可以决定是否使用这个产生式进行归约
    - ✓ LL(k)：要向前查看某个产生式右部推导出的串的前k个符号，才能决定是否使用这个产生式进行推导
    - ✓ 算符优先分析：仅适用于算符优先文法（如表达式）
  - **分析速度快**
  - 在对输入进行自左到右扫描时能尽早准确定位错误
  - 缺点：**LR分析器构造的工作量大**

2. LR(0)分析

- LR(0)分析
  - 根据当前符号栈中的符号串和向前顺序查看输入串的**0个符号**就可唯一地确定句柄以**进行归约**，即，**仅凭符号栈中的符合串即可确定句柄，做出归约决定，不需要向前查看输入符号**
  - 对文法的限制较大，对绝大多数高级语言的语法分析器不适用
  - 构造LR(0)分析表的思想和方法是构造其他LR分析表的基础

2. LR(0)分析

- 项目[item]
  - LR语法分析器通过维护一些状态，用这些状态来表明我们在语法分析过程中所处的位置，从而做出**shift或reduce决定**
  - **状态由项目[item]集表示**
  - 在文法G中每个产生式的**右部适当位置添加一个圆点**构成项目
  - 例：产生式A→XYZ 对应有4个项目

    [0] A→ • XYZ　　[1] A→X • YZ

    [2] A→XY • Z　　[3] A→XYZ •

    产生式A→ε只有一个项目A→•

2. LR(0)分析

- 项目的含义
  - 圆点在最左部（A→•XYZ）表示希望用产生式的右部归约
  - 圆点的左部（A→X•YZ）表示分析过程中已识别过的部分
  - 圆点的右部（A→X•YZ）表示待识别部分
  - 圆点达到最右边（A→XYZ•）表示句柄已形成，可以进行归约

2. LR(0)分析

- 项目分类
  - **移进项目[shift item]**
    - ✓ 形如**A→α•aβ**的项目，其中**a∈V_T**, α,β∈V*
    - ✓ 即圆点后面为**终结符**的项目
    - ✓ 分析时把a移进符号栈
  - **待约项目[reduce-expected item]**
    - ✓ 形如**A→α•Bβ**的项目，其中**B∈V_N**, α,β∈V*
    - ✓ 即圆点后面为**非终结符**的项目
    - ✓ 表明：用产生式A的右部归约时，首先要将B的产生式右部归约为B，对A的右部才能继续进行分析，即，期待着继续分析过程中**首先能归约得到B**

2. LR(0)分析

- 项目分类
  - **归约项目[reduce item]**
    - ✓ 形如**A→α•**的项目，α∈V*，α=ε对应的项目为A→•
    - ✓ 即**圆点在最右端的项目**
    - ✓ 表明该产生式的右部已分析完，**句柄已形成**，可以把α归约为A
  - **接受项目[accept item]**
    - ✓ 当归约项目为**S'→S•**，其中**S'是文法开始符**
    - ✓ 即对文法开始符的归约项目
    - ✓ **表明输入串可归约为文法开始符，分析结束**

2. LR(0)分析

- LR(0)项目集[canonical LR(0) collection]
  - 用一个LR(0)项目的集合来表示LR(0)自动机的一个状态
  - 为了构造文法的LR(0)项目集，需定义**拓广文法[augmented grammar]**和两个函数：**CLOSURE函数**和**GOTO函数**
- **拓广文法**
  - 在原有文法上，**新增开始符号S'**和**产生式S'→S**
  - 例：文法G[S]：S→aAcBe，A→b，A→Ab，B→d

    其拓广文法G'[S']为：S'→S，S→aAcBe，A→b，A→Ab，B→d
  - 拓广原因：文法开始符S可能出现在产生式的右部，在归约过程中，不能判断是否已经归约到文法的最初开始符，还是归约到在产生式右部出现的开始符，而S'只在产生式左部出现，确保不会混淆

2. LR(0)分析

- **CLOSURE函数**
  - 如果I是文法G的一个项目集，定义和构造I的闭包CLOSURE(I)如下：
    ① I的项目均在CLOSURE(I)中
    ② 若A→α•Bβ属于CLOSURE(I)，**B∈V_N**，则每一形如**B→•r**的项目也属于CLOSURE(I)
    ③ 重复②直到CLOSURE(I)不再扩大为止
  - 说明：圆点后为终结符或圆点在产生式的最后，求闭包时不会增加新的项目
  - 例：S'→E，E→aA|bB，A→cA|d，B→cB|d，若I={ S'→•E }
    则 CLOSURE(I)={ S'→•E **E→•aA，E→•bB** }
- CLOSURE(I)作为DFA的一个状态

2. LR(0)分析

- **GOTO函数**
  - 由DFA的一个状态求其他状态通过**状态转换函数GOTO**
  - 设I为文法G的某一项目集(状态)，$X \in V_N \cup V_T$，则:

    GOTO(I, X)=CLOSURE(J)

    其中J = {任何形如$A \to \alpha X \bullet \beta$的项目 | $A \to \alpha \bullet X\beta \in I$}，称J为“**核**”
  - 例： S’→E，E→aA|bB，A→cA|d，B→cB|d

    若I = {S’→•E，E→•aA，E→•bB}，则:
    - ✓ GOTO(I,E)=CLOSURE({S’→E•})={S’→E•}
    - ✓ GOTO(I,a)=CLOSURE({E→a•A})={E→a•A, **A→•cA, A→•d**}
    - ✓ GOTO(I,b)=CLOSURE({E→b•B})={E→b•B, **B→•cB, B→•d**}

2. LR(0)分析

- 构造以LR(0)项目集为状态的DFA

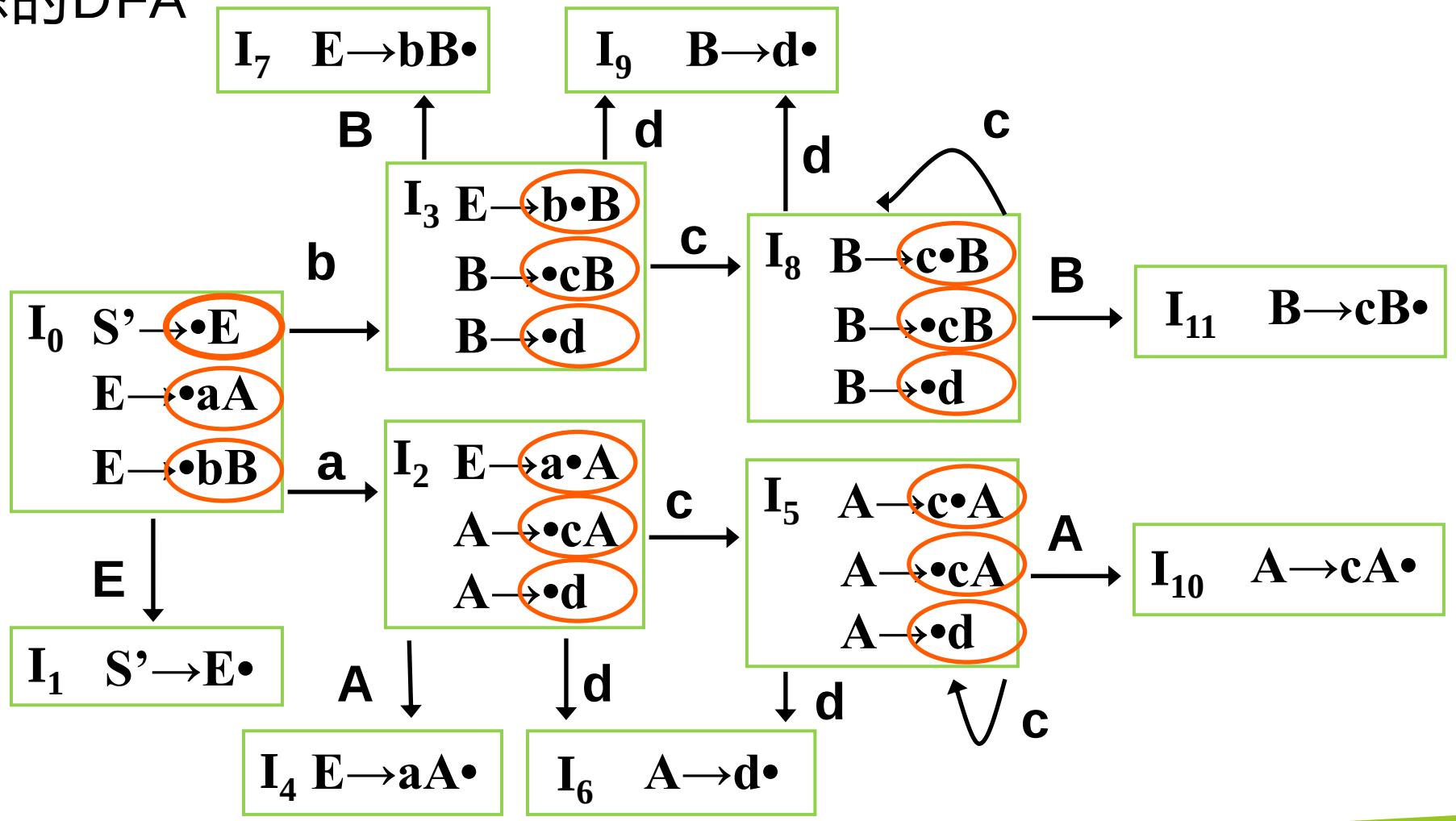
① 构造**初始状态IS_0=CLOSURE({S'→•S})**，并且它是未被标记的；

② 从已经构造的DFA部分图中选择一个未被标记的状态IS_i，标记IS_i，若项目集**IS_i中含有项目U->x•Ry**(R∈V,x,y为任一符号串)，且**GOTO(IS_i,R)=IS_j**，若IS_j不在DFA中，则将IS_j作为未被标记的加入DFA中，且从IS_i出发引一条标记为R的弧到IS_j；

③ 重复②直到没有未被标记的状态为止。

2. LR(0)分析

- 例：拓广文法G'：S'→E，E→aA|bB，A→cA|d，B→cB|d，构造以LR(0)项目集为状态的DFA



2. LR(0)分析

- LR(0)DFA识别的语言
  - 可行前缀[viable prefix]
    - ✓ 一个**可行前缀是**一个**最右句型(规范句型)的前缀**，并且它没有越过该最右句型的最右句柄的右端
    - ✓ 形成**句柄**之前(**包括句柄在内**)的所有规范句型的前缀称为**可行前缀**，即**规范句型的不含句柄右边符号的前缀称为可行前缀**

      例：文法G[S]：S→aAcBe，A→b，A→Ab，B→d，输入串abbcde的规范归约：

      abbcde ⊢ a**A**bcde ⊢ a**A**cde ⊢ aAc**B**e ⊢ S

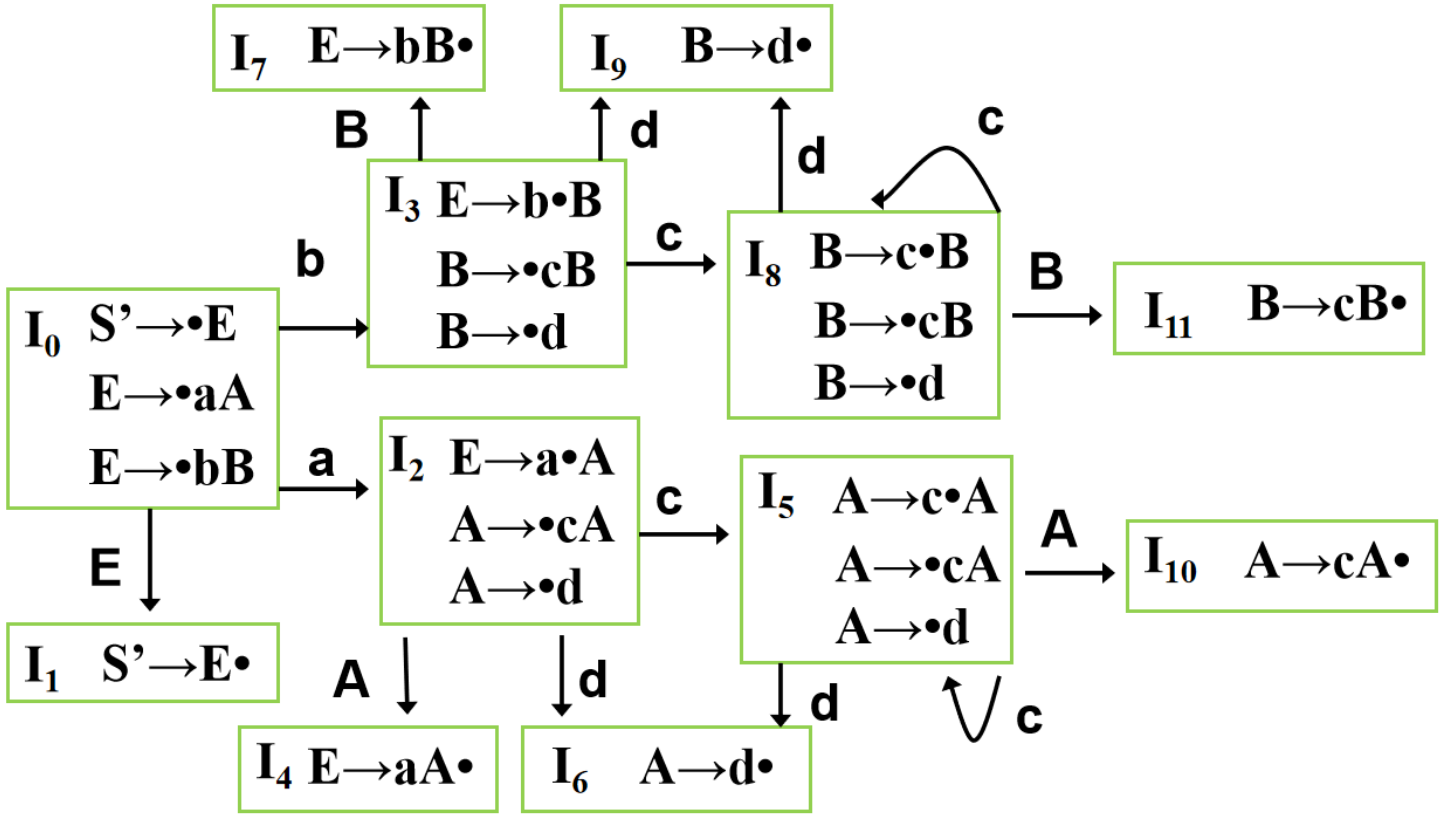
      以规范句型a**A**bcde为例，其句柄为Ab，其可行前缀为：ε，a，aA，aAb

      以规范句型aAc**B**e为例，其句柄为aAcBe，其可行前缀为：ε，a，aA，aAc，aAcB，aAcBe

2. LR(0)分析

- LR(0)DFA识别的语言
  - 句型acd的归约过程：

    acd|-acA|-aA|-E|-S’
  - 对句型acd：
    - ✓ 可行前缀为ε，a，ac，acd
  - 对句型acA：
    - ✓ 可行前缀为ε，a，ac，acA
  - 对句型aA：
    - ✓ 可行前缀为ε，a，aA



  - 可见，**可行前缀是已被DFA正确识别的规范句型的一部分**，其对应的LR分析的操作为**移进(符号栈)**

2. LR(0)分析

- 可行前缀[viable prefix]
  - 可行前缀一定是某个规范句型(右句型)的前缀，即，$S\Rightarrow^{*}_{rm}\alpha\omega$，其中$\alpha$为符号栈中的内容，$\omega$不含非终结符
  - 并不是所有规范句型的前缀是可行前缀

    例：对文法G[E]：E→TE’，E’→+TE’|ε，T→FT’，T’→*FT’|ε，F→(E)|n，

    $E\Rightarrow^{*}_{rm}F\times n\Rightarrow_{rm}(E)\times n$，**(**，**(E**，**(E)**是可行前缀，但**(E)×不是**可行前缀

    因为**(E)是句柄**，句柄一旦出现，就被归约，不可能与后面的字符构成可行前缀。

2. LR(0)分析

- 有效项目[valid item]
  - 如果存在一个推导过程$S' \Rightarrow_{rm}^{*} \alpha A\omega \Rightarrow_{rm} \alpha\beta_1\beta_2\omega$，则称项目$A \rightarrow \beta_1 \bullet \beta_2$对于可行前缀$\alpha\beta_1$有效（是$\alpha\beta_1$的有效项目），其中ω仅含终结符
  - 可帮助决定分析时是进行移进或归约：
    - ✓ 若**$\beta_2 \neq \varepsilon$**，则表明句柄还没有被全部移进栈中，应**移进[shift]**
    - ✓ 若**$\beta_2 = \varepsilon$**，则β_1为句柄，应选用产生式$A \rightarrow \beta_1$进行**归约[reduce]**
  - 可用于构造DFA的状态：
    - ✓ CLOSURE()：若$A \rightarrow \beta_1 \bullet B\beta_2$对于可行前缀$\alpha\beta_1$有效且有产生式$B \rightarrow \gamma$，则$B \rightarrow \bullet\gamma$也可行前缀$\alpha\beta_1$有效（$S' \Rightarrow_{rm}^{*} \alpha A\omega \Rightarrow_{rm} \alpha\beta_1 B\beta_2\omega \Rightarrow_{rm} \alpha\beta_1\gamma\beta_2\omega$）
    - ✓ GOTO()：若$A \rightarrow \beta_1 \bullet X\beta_2$是$\alpha\beta_1$的有效项目，则$A \rightarrow \beta_1 X \bullet \beta_2$是$\alpha\beta_1 X$的有效项目（$S' \Rightarrow_{rm}^{*} \alpha A\omega \Rightarrow_{rm} \alpha\beta_1 X\beta_2\omega$）

2. LR(0)分析

- LR(0)分析同样是**table-driven**的
- LR(0)分析表
  - **动作表(ACTION)**：表示当前状态下面临输入符（终结符和‘$’）应做的动作是移进、归约、接受或出错
  - **转换表(GOTO)**：表示在当前状态下面临文法符号（可能是终结符或非终结符）时应转向的下一个状态
  - 把关于**终结符**部分的GOTO表和ACTION表重叠，也就是把当前状态下面临**终结符**应做的移进-归约动作和转向动作表示在一起
- 进行LR(0)分析前应构造LR(0)分析表

2. LR(0)分析

- LR(0)分析表的构造算法
  - **含S'→•S项目的项目集对应的状态为初始状态**
  - 分析表的ACTION表和GOTO表构造步骤为：
    - ✓ 若项目A→α•aβ∈k，**a∈V_T**，且**GOTO(k,a)=j**，则置**ACTION[k,a]='S_j'**，**移进**；
    - ✓ 若项目A→α•Bβ∈k，**B∈V_N**，且**GOTO(k,B)=j**，则置**GOTO[k,B]='j'**；
    - ✓ 若项目A→α•∈k，且产生式A→α的编号为**j**，则对任何a（终结符和'$'），置**ACTION[k,a]='$r_j$'**，**归约**；**对任何a都归约，不用考虑a具体是哪个字符——LR(0)的0之所在**
    - ✓ 若项目S'→**S**•∈k，**S是原文法开始符号**，则置**ACTION[k,$]='acc'**，**ACCEPT**；
    - ✓ 不能用上述方法填入的分析表的元素可空着，表示**ERROR**。

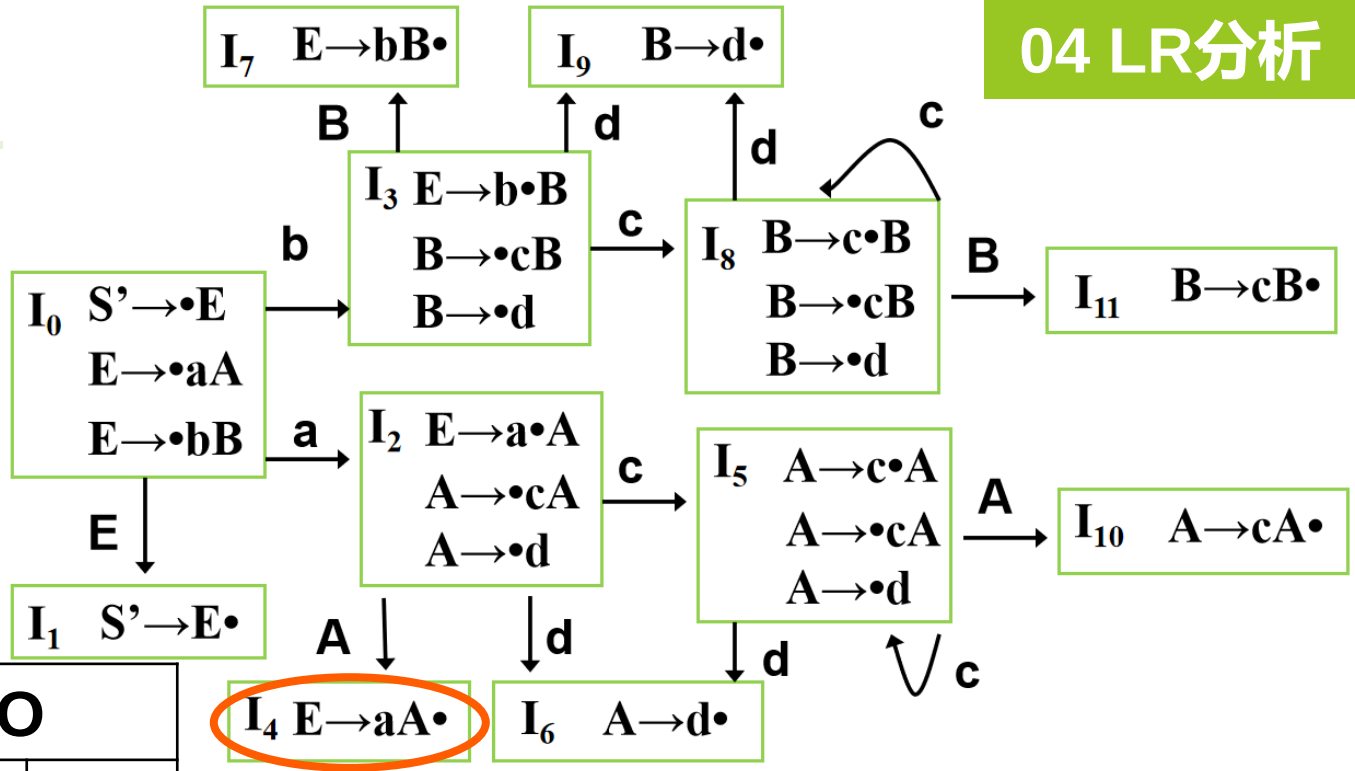
2. LR(0)分析

(0) S′→E　(1) E→aA

(2) E→bB　(3) A→cA

(4) A→d　(5) B→cB

(6) B→d



状态	ACTION					GOTO		
	a	b	c	d	$	E	A	B
0	S_2	S_3				1		
1					acc			
2			S_5	S_6			4	
3			S_8	S_9				7
4	r_1	r_1	r_1	r_1	r_1			
5			S_5	S_6			10	

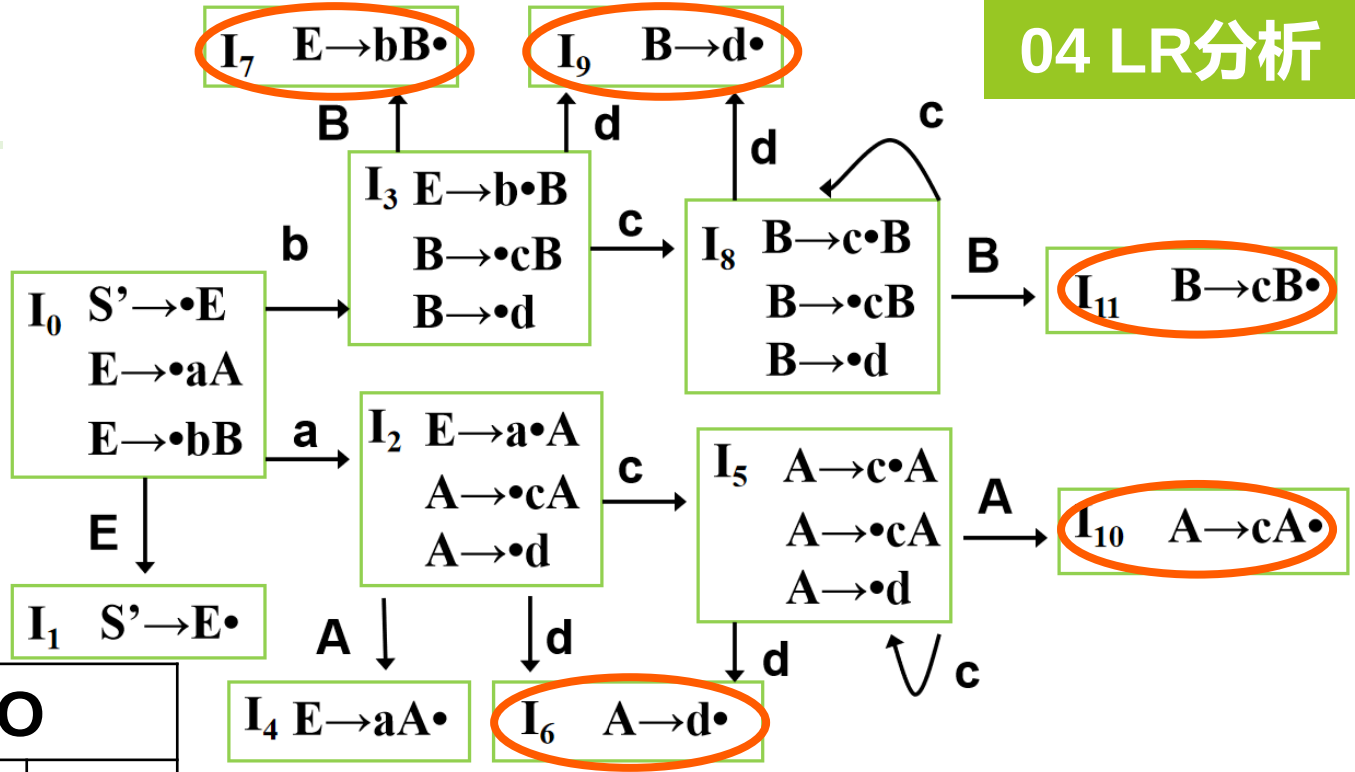
2. LR(0)分析

(0) S′→E　(1) E→aA

(2) E→bB　(3) A→cA

(4) A→d　(5) B→cB

(6) B→d



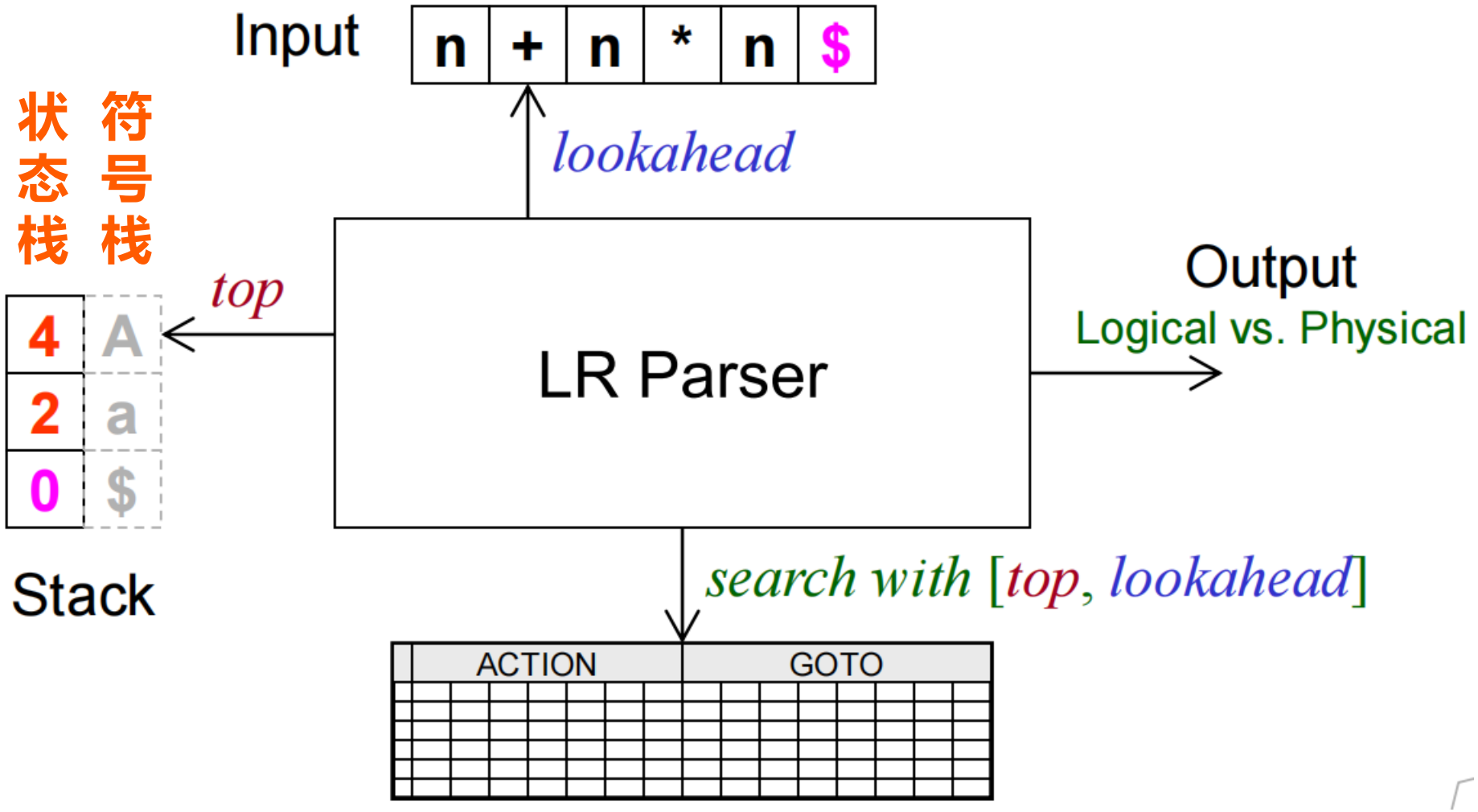
状态	ACTION					GOTO		
	a	b	c	d	$	E	A	B
6	r_4	r_4	r_4	r_4	r_4			
7	r_2	r_2	r_2	r_2	r_2			
8			S_8	S_9				11
9	r_6	r_6	r_6	r_6	r_6			
10	r_3	r_3	r_3	r_3	r_3			
11	r_5	r_5	r_5	r_5	r_5			

2. LR(0)分析

- LR(0)分析器
  - 使用LR(0)分析表的LR分析器称为LR(0)分析器
  - LR(**0**)分析器在分析的过程中**只根据符号栈的内容就能确定句柄，不需要向右查看输入符号**
  - 对文法的限制较大，对绝大多数高级语言的语法分析器不适用

2. LR(0)分析

- LR(0)分析器



2. LR(0)分析

- LR(0)分析器的工作过程
  - 根据**符号栈的栈顶状态**和**输入串的当前符号**查分析表确定应采取的动作（移进、归约、接受或报错），对**状态栈**和**符号栈**进行相应的操作。
    - ① 若ACTION[S,a]=S_j，a为终结符，则把**a移入**符号栈，**j移入**状态栈；
    - ② 若ACTION[S,a]=r_j，a为终结符或$，则：
      - 用第**j**个产生式(A->β)**归约**，将**两个栈弹出**k个元素，其中k为第j个产生式右部符号串β的长度，此时当前面临符号为第j个产生式左部的非终结符(A)；
      - 则非终结符**A移入符号栈**，且：若状态栈当前的栈顶状态为Q，且有GOTO[Q,A]=P，**P移入状态栈**；

      同进同出
    - ③ 若ACTION[S, $]=**acc**，则为**接受**，表明分析成功；
    - ④ 若ACTION[S,a]=**空白**，则转向**出错处理**。

2. LR(0)分析

- LR(0)分析器的工作过程

(0) S′→E　(1) E→aA

(2) E→bB　(3) A→cA

(4) A→d　(5) B→cB

(6) B→d

状态	ACTION					GOTO		
	a	b	c	d	$	E	A	B
0	S_2	S_3				1		
1					acc			
2			S_5	S_6			4	
3			S_8	S_9				7
4	r_1	r_1	r_1	r_1	r_1			
5			S_5	S_6			10	
6	r_4	r_4	r_4	r_4	r_4			
7	r_2	r_2	r_2	r_2	r_2			
8			S_8	S_9				11
9	r_6	r_6	r_6	r_6	r_6			
10	r_3	r_3	r_3	r_3	r_3			
11	r_5	r_5	r_5	r_5	r_5			

状态	ACTION					GOTO		
	a	b	c	d	$	E	A	B
0	S_2	S_3				1		
1					acc			
2			S_5	S_6			4	
3			S_8	S_9				7
4	r_1	r_1	r_1	r_1	r_1			
5			S_5	S_6			10	
6	r_4	r_4	r_4	r_4	r_4			
7	r_2	r_2	r_2	r_2	r_2			
8			S_8	S_9				11
9	r_6	r_6	r_6	r_6	r_6			
10	r_3	r_3	r_3	r_3	r_3			
11	r_5	r_5	r_5	r_5	r_5			

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

输入串
bccd$

步骤	状态栈	符号栈	输入串	ACTION	GOTO
1	0	$	bccd$	S_3	
2	03	$b	ccd$	S_8	
3	038	$bc	cd$	S_8	
4	0388	$bcc	d$	S_9	
5	03889	$bccd	$	r_6	11

状态	ACTION					GOTO		
	a	b	c	d	$	E	A	B
0	S_2	S_3				1		
1					acc			
2			S_5	S_6			4	
3			S_8	S_9				7
4	r_1	r_1	r_1	r_1	r_1			
5			S_5	S_6			10	
6	r_4	r_4	r_4	r_4	r_4			
7	r_2	r_2	r_2	r_2	r_2			
8			S_8	S_9				11
9	r_6	r_6	r_6	r_6	r_6			
10	r_3	r_3	r_3	r_3	r_3			
11	r_5	r_5	r_5	r_5	r_5			

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

输入串
bccd$

步骤	状态栈	符号栈	输入串	ACTION	GOTO
6	0388 (11)	$bccB	$	r_5	11
7	038 (11)	$bcB	$	r_5	7
8	037	$bB	$	r_2	1
9	01	$E	$	acc	

LR(0)分析

- 小结——LR(0)分析过程
  - ① 对文法进行拓广：对文法G[S]，增加一条产生式S’→S，拓广为文法G’[S’]
  - ② 根据产生式构造LR(0)项目集：CLOUSRE函数和GOTO函数
  - ③ 根据项目集构造LR(0)DFA
  - ④ 根据LR(0)DFA构造LR(0)分析表
  - ⑤ 根据LR(0)分析表进行LR(0)分析