



语法分析

——自下向上





■ 自下而上语法分析

从给定的输入串 r 开始;
不断寻找子串与某个产生式的候选式匹配;
用产生式的左部代替候选式(归约);
最终归约到 S 。

 关键：

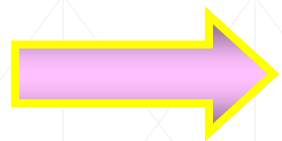
- 1) 确定可归约串 \longrightarrow 归约条件;
- 2) 如何归约 \longrightarrow 归约原则。



- 5.1 “移进—归约”分析法 ←
- 5.3 LR分析概述
- 5.4 LR (0)分析
- 5.5 SLR (1)分析
- 5.6 LR (1)分析
- 5.7 LALR (1)分析
- 5.8 LR分析对二义文法的应用
- 5.9 LR分析的错误处理与恢复
- 5.10 语法分析器的自动生成与YACC(自学)



5.1 “移进—归约”分析法



5.1.1 直观的“移进—归约”分析法

5.1.2 规范归约与句柄



例：文法 $G(S)$:

$$S \rightarrow aABe$$

$$A \rightarrow Abc / b$$

$$B \rightarrow d$$

分析串 $abbcd e$ 是该文法的合法句子。



步骤	分析栈	待分析串	动作
初始	#	<i>abbcede</i> #	移进
(1)	# <i>a</i>	<i>bbcede</i> #	移进
(2)	# <i>ab</i>	<i>bcde</i> #	$A \rightarrow b$ 归约
(3)	# <i>aA</i>	<i>bcde</i> #	移进
(4)	# <i>aAb</i>	<i>cde</i> #	移进
(5)	# <i>aAbc</i>	<i>de</i> #	$A \rightarrow Abc$ 归约
(6)	# <i>aA</i>	<i>de</i> #	移进
(7)	# <i>aAd</i>	<i>e</i> #	$B \rightarrow d$ 归约
(8)	# <i>aAB</i>	<i>e</i> #	移进
(9)	# <i>aABe</i>	#	$S \rightarrow aABe$ 归约
(10)	# <i>S</i>	#	分析成功

规范推导:

$$S \Rightarrow \underline{aABe} \quad \textcircled{4}$$

$$\Rightarrow aA \underline{de} \quad \textcircled{3}$$

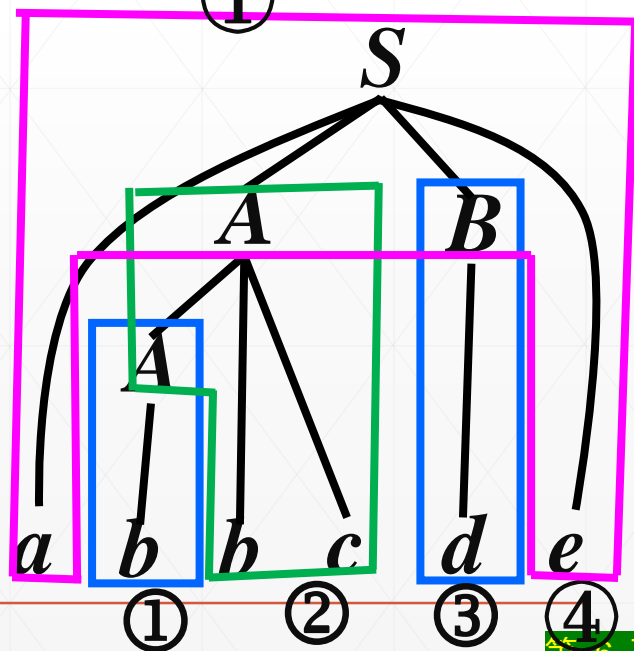
$$\Rightarrow a \underline{Abcde} \quad \textcircled{2}$$

$$\Rightarrow \underline{abbcede} \quad \textcircled{1}$$

$$S \rightarrow aABe$$

$$A \rightarrow Abc/b$$

$$B \rightarrow d$$

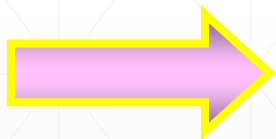




5.1 “移进—归约”分析法

5.1.1 直观的“移进—归约”分析法

5.1.2 规范归约与句柄





- 定义(短语, 对应句型 $\alpha\beta\delta$ 分析树中的一个“子树”: 根节点 A , 叶节点 β)

S 是文法 G 的开始符号, $\alpha\beta\delta$ 是 G 的一个句型,
若 $S \xRightarrow{*} \alpha A \delta$ 且 $A \xRightarrow{\pm} \beta$,
则 β 是句型 $\alpha\beta\delta$ 相对于 A 的**短语**。

- 定义(直接短语)

S 是文法 G 的开始符号, $\alpha\beta\delta$ 是 G 的一个句型,
若 $S \xRightarrow{*} \alpha A \delta$ 且 $A \rightarrow \beta$,
则 β 是句型 $\alpha\beta\delta$ 相对于 A 的**直接短语**。



■ 定义(句柄)

一个句型的最左直接短语称为**句柄**。



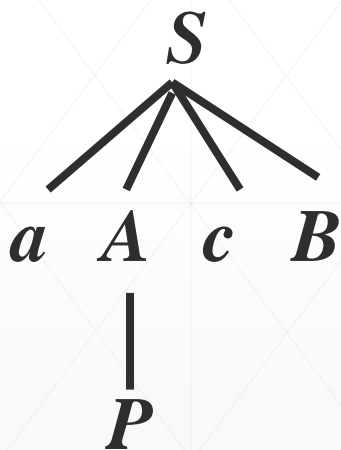
- ◆ 直接短语是短语
- ◆ 句柄是直接短语且具有最左性
- ◆ 短语、直接短语是句型的子串
- ◆ 在规范句型的规范推导序列中，最后使用的产生式的右部是句柄
- ◆ 句型对应的语法分析树中观察短语与句柄更直观



例：设有文法 G 和串 $aPcB$

$G: S \rightarrow aAcB \quad A \rightarrow P \quad P \rightarrow ab \quad B \rightarrow d$

存在推导 $S \Rightarrow \underline{aAcB} \Rightarrow a\underline{P}cB$



P 是句型 $aPcB$ 相对于 A 的短语，

也是相对于 A 的直接短语。

$aPcB$ 是句型 $aPcB$ 相对于 S 的短语。

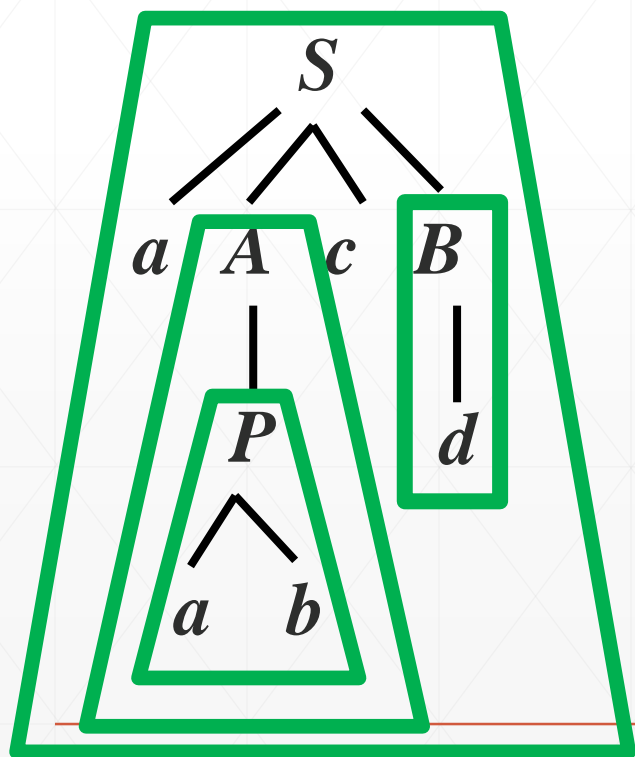
P 是句型 $aPcB$ 的句柄。



例： 设有文法 G 和串 $aabcd$

$G: S \rightarrow aAcB \quad A \rightarrow P \quad P \rightarrow ab \quad B \rightarrow d$

存在推导 $S \Rightarrow \underline{aAcB} \Rightarrow aA\underline{cd} \Rightarrow a\underline{P}cd \Rightarrow a\underline{abcd}$
 $S \Rightarrow \underline{aAcB} \Rightarrow a\underline{PcB} \Rightarrow a\underline{abcB} \Rightarrow aabcd\underline{d}$



ab 是句子 $aabcd$ 相对于 P 的直接短语；
 相对于 A 的短语；

d 是句子 $aabcd$ 相对于 B 的直接短语；

$aabcd$ 是句子 $aabcd$ 相对于 S 的短语

ab 是是句子 $aabcd$ 的句柄。



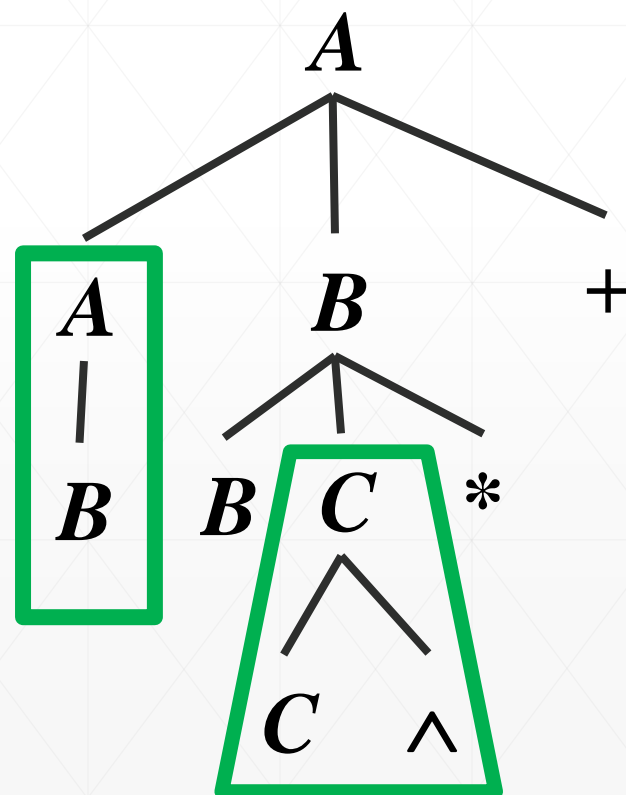
例：设有文法 $G[A]$:

$A \rightarrow AB+|B$ $B \rightarrow BC*|C$ $C \rightarrow C\wedge|a$

句型 $BBC\wedge*+$ 的直接短语是 【 B 和 $C\wedge$ 】

句柄是 【 B 】。

方法：构造句型对应的语法分析树





$S \Rightarrow \underline{aABe} \Rightarrow aA\underline{de} \Rightarrow a\underline{Abc}de \Rightarrow a\underline{b}bcde$

S



$S \rightarrow aABe$

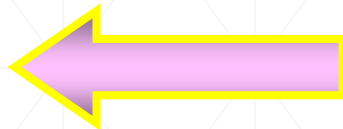
$A \rightarrow Abc/b$

$B \rightarrow d$



5.1 “移进—归约”分析法

5.3 LR分析概述



5.4 LR (0)分析

5.5 SLR (1)分析

5.6 LR (1)分析

5.7 LALR (1)分析

5.8 LR分析对二义文法的应用

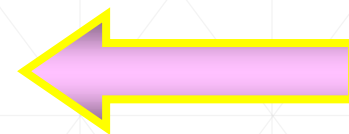
5.9 LR分析的错误处理与恢复

5.10 语法分析器的自动生成与YACC



5.3 LR分析概述

5.3.1 LR分析



5.3.2 LR分析器结构、组成与工作原理

5.3.3 LR分析实例



LR分析技术是编译系统中语法分析器实现最常用、最有效的一种分析方法。

- (1) 理论上比较完善;
- (2) 适用性强 , 对 G 限定少;
- (3) 便于自动生成。



LR分析：一类对源程序串进行**自左向右扫描**并进行**规范归约**的语法分析方法。

LR (k)

在LR分析的每一步，仅据分析栈当前已经移进和归约的全部语法符号并最多再向前看 k 个输入字符，就能确定适合于文法规则的**句柄**是否已在栈顶形成，从而立即确定当前的分析动作。

分析模式：最右推导的逆序(规范归约)

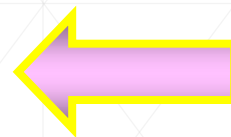
扫描模式：自左向右



5.3 LR分析概述

5.3.1 LR分析

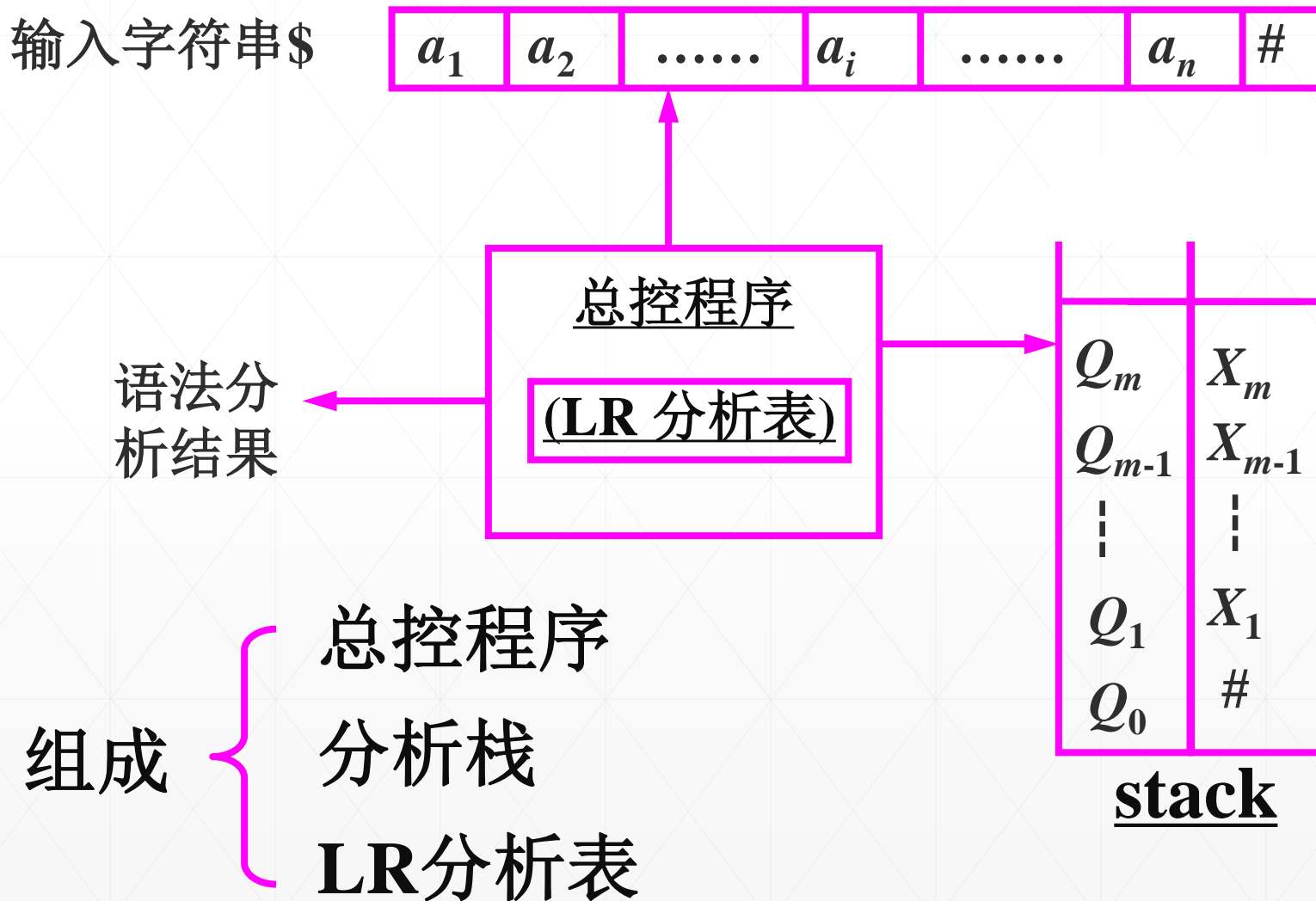
5.3.2 LR分析器结构、组成与工作原理



5.3.3 LR分析实例




■ LR分析器逻辑结构



(1) 分析栈

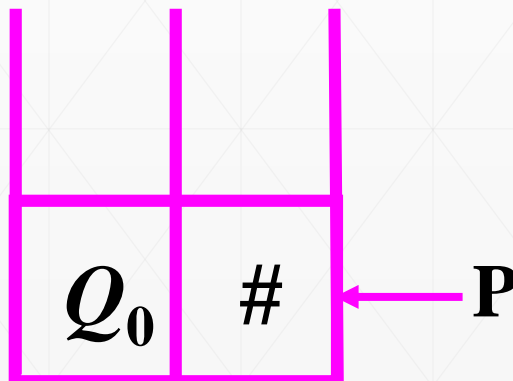
辅助完成LR分析的数据结构。

stack { **状态** Q_i : 记录分析过程中每一步的“历史”和“展望”信息;
文法符号 X_i : 分析过程中移进(V_T)和归约(V_N)的符号;



Q_m	X_m
Q_{m-1}	X_{m-1}
\vdots	\vdots
Q_1	X_1
Q_0	$\#$

stack初始化:





- $$\textcircled{4} \quad E \rightarrow b$$

④ $E \rightarrow b$ **F** ↓
输入字符串 $a, b, a \#$

$b \Rightarrow E$ (归约)

，(移进栈)

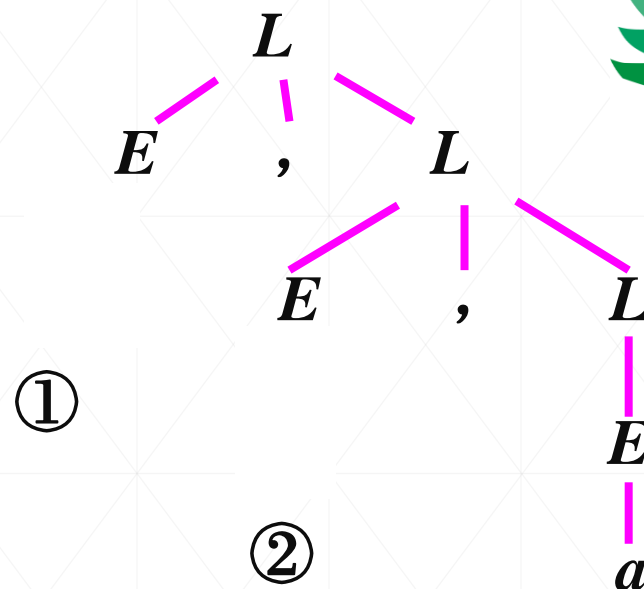
$$a \Rightarrow E \text{ (归约)}$$


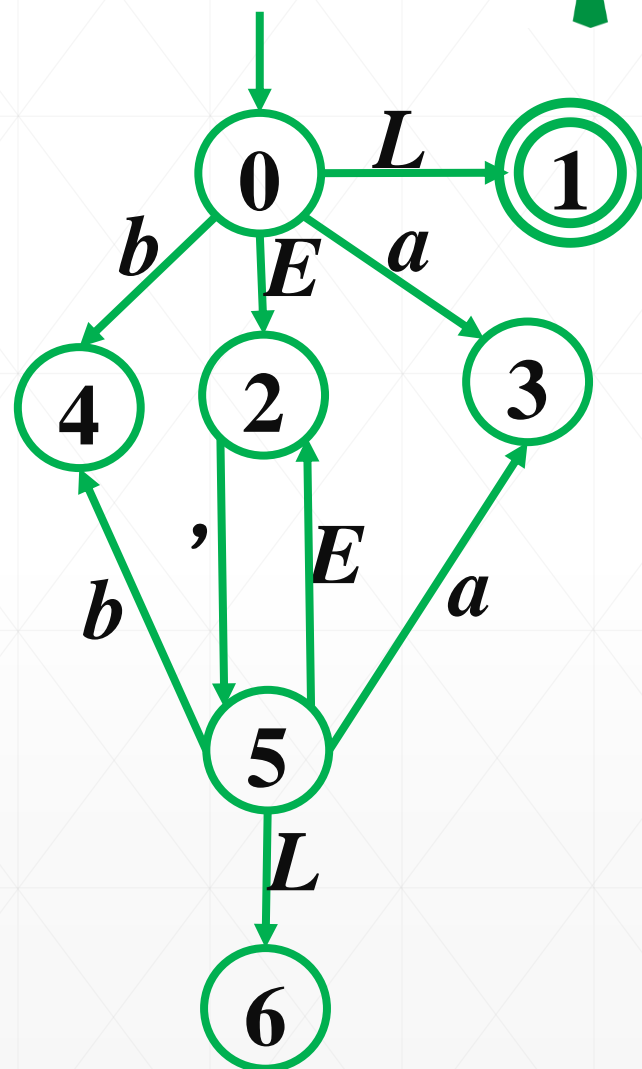
Diagram illustrating the stack state during LR item set transitions:

$b \Rightarrow E$ (归约)	2	E	P
, (移进栈)	5	,	
$a \Rightarrow E$ (归约)	2	E	
	0	#	



文法G(L)的LR分析表

状态	ACTION表				GOTO表	
	<i>a</i>	<i>b</i>	,	#	<i>E</i>	<i>L</i>
0	S ₃	S ₄			2	1
1				acc		
2			S ₅	r ₂		
3			r ₃	r ₃		
4			r ₄	r ₄		
5	S ₃	S ₄			2	6
6				r ₁		





(2) LR分析表

LR分析表是LR分析器的核心。

分析表 { 分析动作表(ACTION表)
 状态转换表(GOTO表)

分析动作表 $\longrightarrow Q \times (V_T \cup \{\#\})$

状态转换表 $\longrightarrow Q \times V_N$



LR 分析表

table state V	Action (V_T)				Goto (V_N)			
	V_{T1}	V_{T2}	$\dots V_{Tn}$	#	V_{N1}	V_{N2}	\dots	V_{Nm}
Q_0			
Q_1			
...			
Q_n			



状态转换表 (Goto)

state V_N	X_1	X_2	...	X_n
Q_0	goto(Q_0, X_1)	goto(Q_0, X_2)	...	goto(Q_0, X_n)
Q_1	goto(Q_1, X_1)	goto(Q_1, X_2)	...	goto(Q_1, X_n)
...
Q_m	goto(Q_m, X_1)	goto(Q_m, X_2)	...	goto(Q_m, X_n)

$Q_i \in$ 状态; $X_i \in$ 非终结符集;



$$\text{goto}(Q_i, X_i) = \begin{cases} Q_j & (\text{移进: 将 } Q_j \text{ 状态压入状态栈}) \\ \text{空} & (\text{不会出现的情况, 没有动作}) \end{cases}$$

意注

$\text{goto}(Q_i, X_i)$ 的 Q_i 、 X_i 意指当前栈顶的 X_i 和次栈顶的 Q_i 元素。





分析动作表 (Action)

state $\backslash V_T$	a_1	a_2	...	a_n
Q_0	$\text{action}(Q_0, a_1)$	$\text{action}(Q_0, a_2)$...	$\text{action}(Q_0, a_n)$
Q_1	$\text{action}(Q_1, a_1)$	$\text{action}(Q_1, a_2)$...	$\text{action}(Q_1, a_n)$
...
Q_n	$\text{action}(Q_n, a_1)$	$\text{action}(Q_n, a_2)$...	$\text{action}(Q_n, a_n)$

$Q_i \in \text{状态}$; $a_i \in \text{终结符集或}\#$;

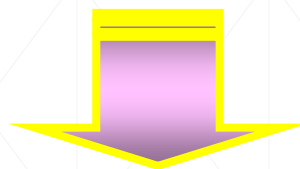


$\text{action}(Q_i, a_i) = \left\{ \begin{array}{l} S_{Q_j} \text{ (移进: 将 } a_i \text{ 和 } Q_j \text{ 状态压入栈)} \end{array} \right.$



Q_0	Q_1	\dots	Q_m
$\#$	X_1	\dots	X_m

\uparrow
P



Q_0	Q_1	\dots	Q_m	Q_k
$\#$	X_1	\dots	X_m	a_i

\uparrow
P

$\$: a_1 a_2 \dots a_i a_{i+1} \dots a_n \#$

\uparrow
F

$\text{action}(Q_m, a_i) = S_{Q_k}$

$\$: a_1 a_2 \dots a_i a_{i+1} \dots a_n \#$

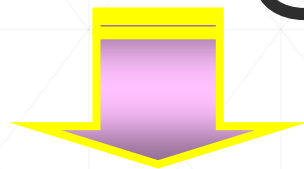
\uparrow
F



$$\text{action}(Q_i, a_i) = \begin{cases} S_{Q_j} \text{ (移进: 将 } a_i \text{ 和 } Q_j \text{ 状态压入栈)} \\ r_j \text{ (归约: 用第 } j \text{ 个产生式归约)} \end{cases}$$



Q_0	Q_1	\dots	Q_{m-r}	Q_{m-r+1}	\dots	Q_m
#	X_1	\dots	X_{m-r}	X_{m-r+1}	\dots	X_m



句柄

P

$$\text{action}(Q_m, a_i) = r_j$$

① 归约。设G第j个产生式为： $A \rightarrow X_{m-r+1} X_{m-r+2} \dots X_m$

Q_0	Q_1	\dots	Q_{m-r}		
#	X_1	\dots	X_{m-r}	A	

P



② 查goto表。 $\text{goto}(Q_{m-r}, A) = Q_j$

Q_0	Q_1	...	Q_{m-r}	Q_j
#	X_1	...	X_{m-r}	A

P

A red oval highlights the row containing Q_{m-r} and Q_j , and the column containing A . A red arrow points from the label P to the cell containing A .

$\$: a_1 a_2 \dots a_i a_{i+1} \dots a_n \#$ (扫描指针不变)

F

A pink arrow points from the label F to the symbol a_i in the string above.



$$\text{action}(Q_i, a_i) = \begin{cases} S_{Qj} \text{ (移进: 将 } a_i \text{ 和第 } j \text{ 个状态压入栈)} \\ r_j \text{ (归约: 用第 } j \text{ 个产生式归约)} \\ \text{acc (接受: 分析成功)} \\ \text{error (出错: 语法错, 调出错处理程序)} \end{cases}$$



文法G(L)的LR分析表

状态	ACTION表				GOTO表	
	<i>a</i>	<i>b</i>	,	#	<i>E</i>	<i>L</i>
0	S ₃	S ₄			2	1
1				acc		
2			S ₅	r ₂		
3			r ₃	r ₃		
4			r ₄	r ₄		
5	S ₃	S ₄			2	6
6				r ₁		



(3) LR分析总控程序

① 分析开始，将开始状态 Q_0 及“#”压入分析栈；
// 初始化

② 据当前分析栈栈顶 Q_m ，当前输入符号 a_i 查action表：

i) 若 $\text{action}(Q_m, a_i) = S_{Q_j}$ ，完成移进动作；

ii) 若 $\text{action}(Q_m, a_i) = r_j$ ，完成归约动作。

iii) 若 $\text{action}(Q_m, a_i) = \text{acc}$ ，分析成功；

iv) 若 $\text{action}(Q_m, a_i) = \text{error}$ ，出错处理。

③ 转②。

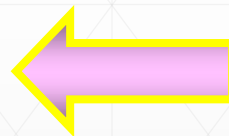


5.3 LR分析概述

5.3.1 LR分析

5.3.2 LR分析器结构、组成与工作原理

5.3.3 LR分析实例





例：设有文法 $G(L)$ 和
 $G(L)$ 的LR分析表。

$$\textcircled{1} \quad L \rightarrow E, L$$

$$\textcircled{2} \quad L \rightarrow E$$

$$\textcircled{3} \quad E \rightarrow a$$

$$\textcircled{4} \quad E \rightarrow b$$

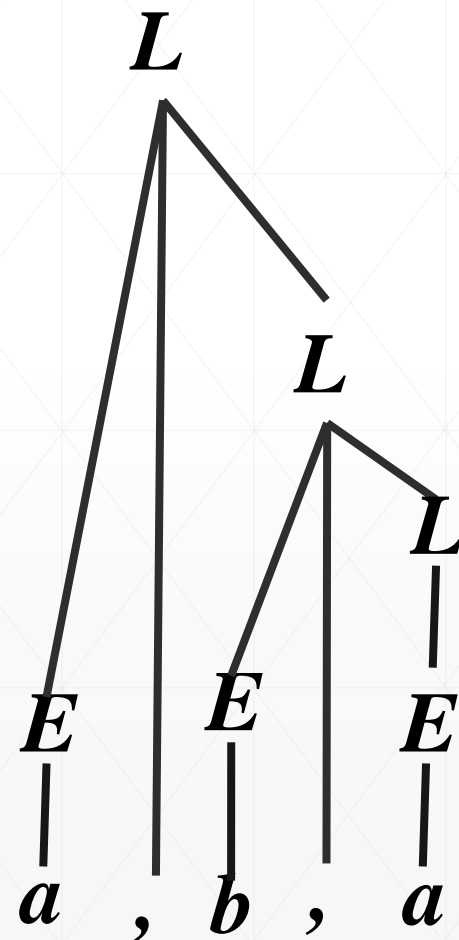
分析字符串 a, b, a



文法 $G(L)$ 的句子 a, b, a 的分析过程

① $L \rightarrow E, L$ ② $L \rightarrow E$ ③ $E \rightarrow a$ ④ $E \rightarrow b$

步	栈		余留串	分析动作
	符号栈	状态栈		
0	#	0	$a, b, a \#$	S_3
1	# a	03	$, b, a \#$	r_3
2	# E	02	$, b, a \#$	S_5
3	# $E,$	025	$b, a \#$	S_4
4	# E, b	0254	$, a \#$	r_4
5	# E, E	0252	$, a \#$	S_5
6	# $E, E,$	02525	$a \#$	S_3
7	# E, E, a	025253	#	r_3
8	# E, E, E	025252	#	r_2
9	# E, E, L	025256	#	r_1
10	# E, L	0256	#	r_1
11	# L	01	#	acc



分析表



综述：

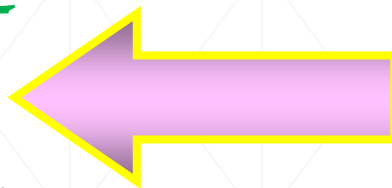
- (1) 处理直观简单；
- (2) 基本实现思想：引入状态，状态埋伏了分析的“历史”和“展望”信息；
- (3) 应用范围广，对 G 限定少；
- (4) LR分析的关键——LR分析表：集成了全部分析信息。



5.1 “移进—归约”分析法

5.3 LR分析概述

5.4 LR (0)分析



5.5 SLR (1)分析

5.6 LR (1)分析

5.7 LALR (1)分析

5.8 LR分析对二义文法的应用

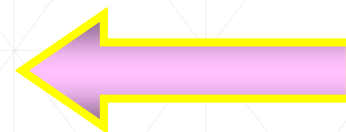
5.9 LR分析的错误处理与恢复

5.10 语法分析器的自动生成与YACC



5.4 LR (0)分析

5.4.1 LR(0)分析实现思想



5.4.2 构造LR(0)项目集规范族

5.4.3 LR (0)分析表的构造



■ 定义（活前缀）

规范句型的不含句柄之后任何符号的前缀，称为该句型的活前缀。



注意：

- (1) $\exists S \xrightarrow{*}_R \alpha A \omega \Rightarrow \alpha \beta \omega$ ，若串 γ 是 $\alpha \beta$ 的前缀，则 γ 是活前缀；当 $\gamma = \alpha \beta$ ，则 γ 称为可归前缀；
- (2) 活前缀特点：不含句柄之后的任何符号；
- (3) LR分析中栈中符号始终是活前缀，当构成刚好包含句柄的活前缀(可归前缀)时，实施归约。



文法 $G(L)$ 的句子 a, b, a 的分析过程

① $L \rightarrow E, L$ ② $L \rightarrow E$ ③ $E \rightarrow a$ ④ $E \rightarrow b$

步	栈		余留串	分析动作
	符号栈	状态栈		
0	#	0	$a, b, a \#$	S_3
1	$\#a$	03	$, b, a \#$	$r_3(E \rightarrow a)$
2	$\#E$	02	$, b, a \#$	S_5
3	$\#E,$	025	$b, a \#$	S_4
4	$\#E, b$	0254	$, a \#$	$r_4(E \rightarrow b)$
5	$\#E, E$	0252	$, a \#$	S_5
6	$\#E, E,$	02525	$a \#$	S_3
7	$\#E, E, a$	025253	$\#$	$r_3(E \rightarrow a)$
8	$\#E, E, E$	025252	$\#$	$r_2(L \rightarrow E)$
9	$\#E, E, L$	025256	$\#$	$r_1(L \rightarrow E, L)$
10	$\#E, L$	0256	$\#$	$r_1(L \rightarrow E, L)$
11	$\#L$	01	$\#$	acc

L



■ 定义：（**LR(0)项目**）

在文法G的每个产生式的右部(候选式)的任何位置上添加一个圆点，这样构成的每个产生式称为**LR(0)项目**。

约定：若产生式形式为 $A \rightarrow \epsilon$ 则其**LR(0)项目**为： $A \rightarrow \cdot$

圆点的位置标记相应的候选式已被匹配（放入栈中）了多少。



例： 设文法 $G(S)$

$$S' \rightarrow S \quad S \rightarrow A \mid B \quad A \rightarrow aA \mid b \mid \varepsilon \quad B \rightarrow c$$

则 $G(S)$ 的LR(0)项目有：

$$S' \rightarrow \cdot S \quad S' \rightarrow S \cdot$$

$$S \rightarrow \cdot A \quad S \rightarrow A \cdot$$

$$S \rightarrow \cdot B \quad S \rightarrow B \cdot$$

$$A \rightarrow \cdot aA \quad A \rightarrow a \cdot A \quad A \rightarrow aA \cdot$$

$$A \rightarrow \cdot b \quad A \rightarrow b \cdot$$

$$A \rightarrow \cdot$$

$$B \rightarrow \cdot c \quad B \rightarrow c \cdot$$



LR (0) 项目分类

唯一项目

(1) 接受项目: $S \rightarrow \alpha \cdot$ (S 是开始符号)

分析栈中内容恰好为开始符号 S 的候选式 α ，用 $S \rightarrow \alpha$ 进行归约，则整个分析成功。

(2) 归约项目: $A \rightarrow \alpha \cdot$ (A 不是开始符号)

句柄 α 恰好包含在栈中，即当前栈中的内容构成了刚好含句柄 α 的活前缀，应按 $A \rightarrow \alpha$ 进行归约。

(3) 移进项目: $A \rightarrow \alpha \cdot a\beta$ ($a \in V_T$)

分析栈中是不完全包含句柄的活前缀，为构成含有句柄 $\alpha a\beta$ 的活前缀，需将 a 移进分析栈。

(4) 待约项目: $A \rightarrow \alpha \cdot B\beta$ ($B \in V_N$)

分析栈中是不完全包含句柄的活前缀，为构成含有句柄 $\alpha B\beta$ 的活前缀，应先把当前分析的字符串中的相应内容归约到 B 。



例：设文法 $G(S)$

$$S' \rightarrow S \quad S \rightarrow A \mid B \quad A \rightarrow aA \mid b \mid \varepsilon \quad B \rightarrow c$$

则 $G(S)$ 的LR(0)项目有：

$$S' \rightarrow \cdot S$$

$$S \rightarrow \cdot A$$

$$S \rightarrow \cdot B$$

$$A \rightarrow \cdot aA$$

$$A \rightarrow \cdot b$$

$$A \rightarrow \cdot$$

$$B \rightarrow \cdot c$$

$$S' \rightarrow S \cdot$$

$$S \rightarrow A \cdot$$

$$S \rightarrow B \cdot$$

$$A \rightarrow a \cdot A$$

$$A \rightarrow b \cdot$$

$$B \rightarrow c \cdot$$

$$A \rightarrow aA \cdot$$



构造识别文法 G 的所有可归前缀的非确定有限自动机：

基本项目

- ①所有LR(0)项目分别对应NFA的一个状态。
- ②开始符号的第一个LR(0)项目对应的状态为NFA的初态。

圆点在候选式的最左侧的LR(0)项目
- ③LR(0)项目为归约或接受项目的状态为NFA的终态。



④若状态*i*和状态*j*出自同一产生式，两个状态的LR(0)项目的圆点只相差一个位置，即：

*i*中项目为： $A \rightarrow \alpha \cdot X \beta$ ；*j*中项目为： $A \rightarrow \alpha X \cdot \beta$

则状态*j*为状态*i*识别字符*X*的后继状态，即在状态图中有：



⑤若状态*i*为待约项目，(设待约到的非终结符号为*B*，即 $A \rightarrow \alpha \cdot B \beta$)，状态*j*为待约到的非终结符号对应的第一个LR(0)项目(即 $B \rightarrow \cdot \gamma$)，则状态*j*为状态*i*识别 ϵ 的后继状态，即在状态图中有：



其中 $\alpha, \beta \in V^*$, $X \in V$, $A, B \in V_N$



识别文法G的所有可归前缀的NFA

NFA $M = (Q, \Sigma, f, q_0, Z)$

LR(0)项目

G的V

基本项目

归约+接受项目

若 i 为: $A \rightarrow \alpha \cdot X \beta$

j 为: $A \rightarrow \alpha X \cdot \beta$

则 $j \in f(i, X)$ 。

若 i 为: $A \rightarrow \alpha \cdot B \beta (B \in V_N)$,

j 为: $B \rightarrow \cdot \gamma$

则 $j \in f(i, \varepsilon)$ 。

其中 $\alpha, \beta \in V^*$, $X \in V$, $A, B \in V_N$

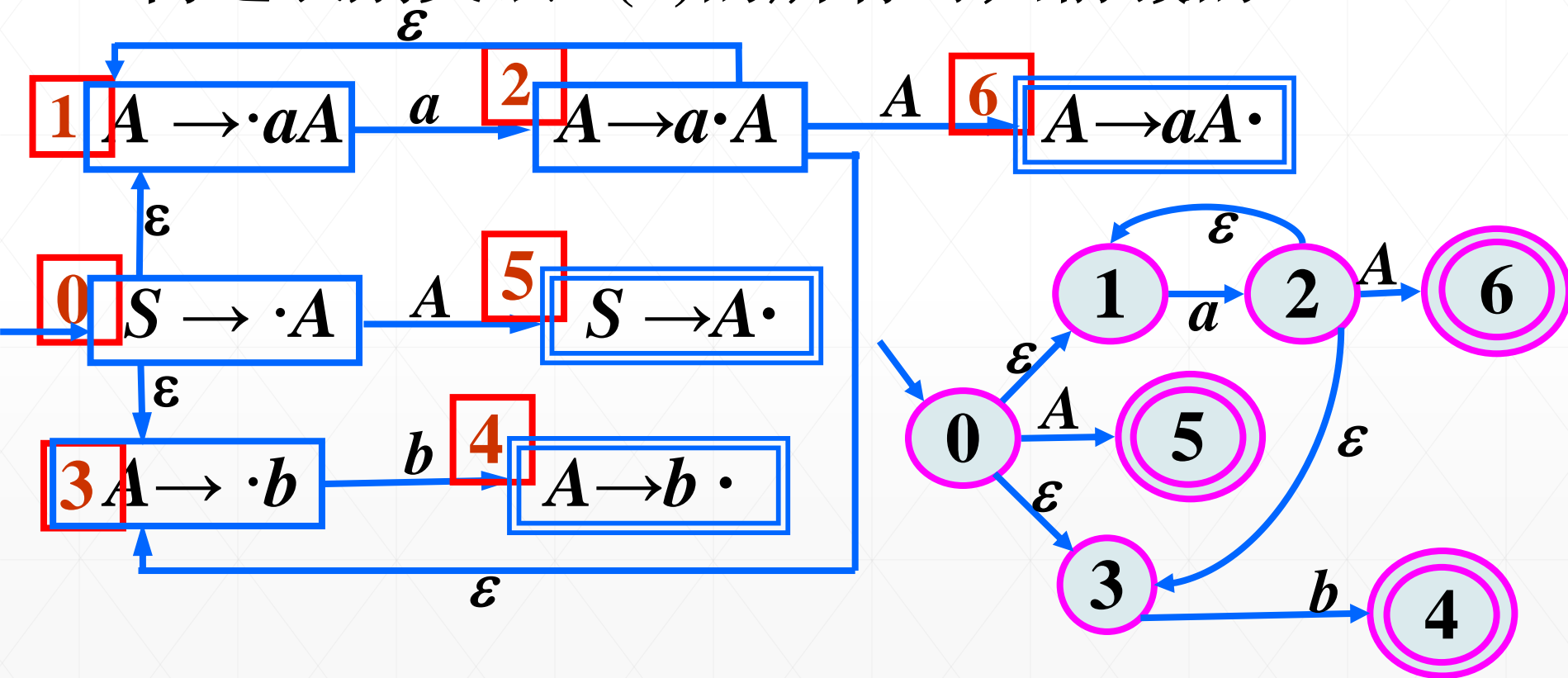


例: 设文法 $G(S)$

$$S \rightarrow A$$

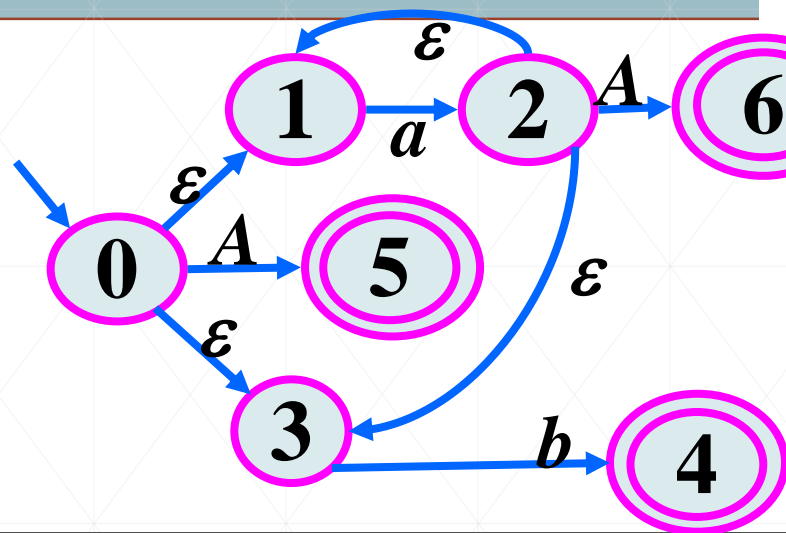
$$A \rightarrow aA \mid b$$

构造识别文法 $G(S)$ 的所有可归前缀的NFA。





识别文法 $G(S)$ 的所有可归前缀的
DFA:



I	I_a	I_A	I_b
0 {0,1,3}	1 {2,1,3}	2 {5}	3 {4}
1 {2,1,3}	1 {2,1,3}	4 {6}	3 {4}
2* {5}	∅	∅	∅
3* {4}	∅	∅	∅
4* {6}	∅	∅	∅



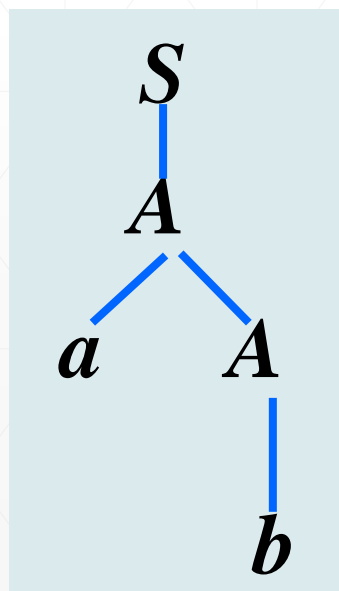
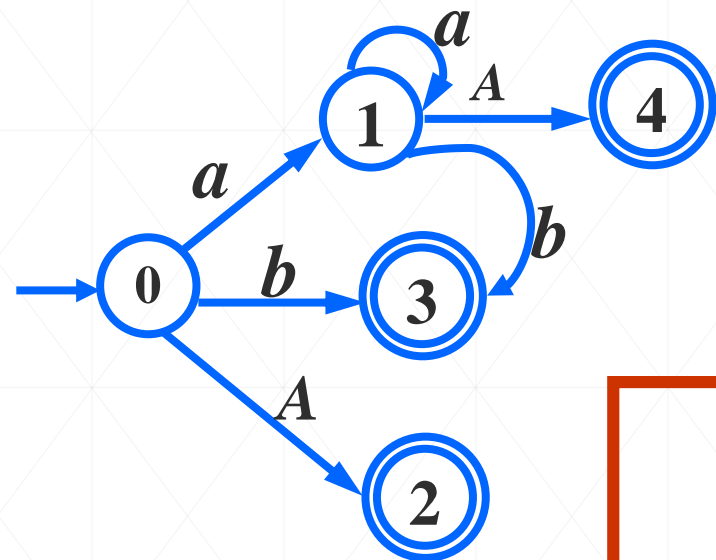
识别可归前缀的DFA

句子 ab 的分析:

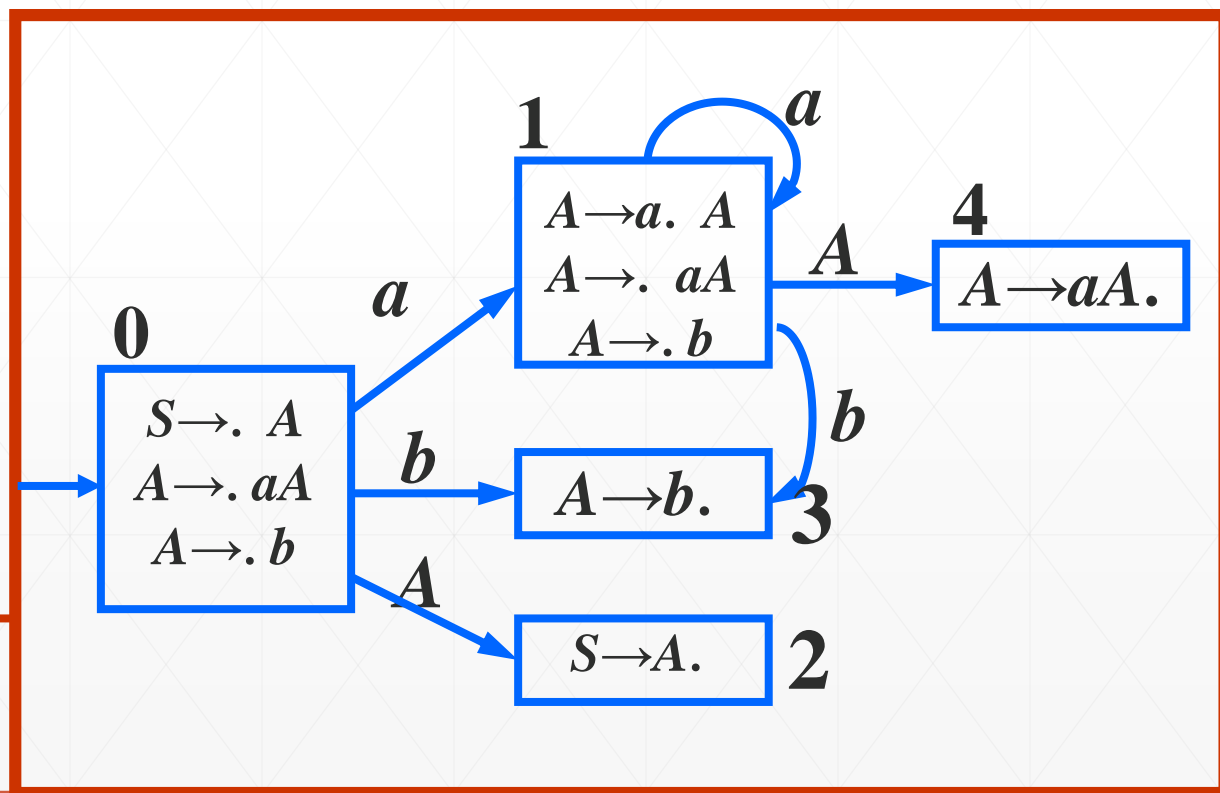
$0 \rightarrow 1 \rightarrow 3$ ab (b 归约到 A)

$0 \rightarrow 1 \rightarrow 4$ aA (aA 归约到 A)

$0 \rightarrow 2$ A (A 归约到 S)



C



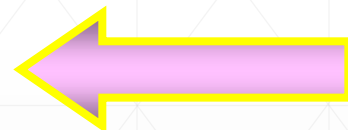
返回



5.4 LR (0)分析

5.4.1 LR(0)分析实现思想

5.4.2 构造LR(0)项目集规范族



5.4.3 LR (0)分析表的构造



■ 定义

识别文法 G 的可归前缀的DFA，其状态对应的LR(0)项目的集合的全体称为文法 G 的LR(0)项目集规范族。

LR(0)项目集

上节例文法 $G(S')$ 的LR(0)项目集规范族 C 为：

$$C = (\{ S \rightarrow \cdot A, A \rightarrow \cdot aA, A \rightarrow \cdot b \} \\ \{ A \rightarrow a \cdot A, A \rightarrow \cdot aA, A \rightarrow \cdot b \} \\ \{ S' \rightarrow A \cdot \} \quad \{ A \rightarrow b \cdot \} \quad \{ A \rightarrow aA \cdot \})$$



构造LR(0)项目集规范族的方法 (之一)

第一步: 构造文法 G 的LR(0)项目;

第二步: 基于LR(0)项目, 构造识别文法 G 所有可归前缀的NFA;

第三步: NFA确定化为DFA, 该DFA的所有状态所对应的项目集的集合, 即构成了 G 的LR(0)项目集规范族。



构造LR(0)项目集规范族的方法（之二）

引入两个函数 { **closure**（项目集闭包）函数
GO（项目集转移）函数



项目集闭包closure

I 是文法 G 的任一项目集，则构造 I 的项目集闭包closure(I)的方法如下：

- ① I 中的每一个项目皆属于closure(I);
- ② 若形如 $A \rightarrow \alpha \cdot B\beta$ ($B \in V_N$) 的项目属于 I ,
则对 G 中的任何产生式 $B \rightarrow \gamma$ 的第一个
LR(0)项目($B \rightarrow \cdot \gamma$)也属于closure(I);
- ③ 重复上述步骤，直至不再有新的项目加入
closure(I)为止;



项目集转移函数 $GO(I, X)$:

■ 定义

若 I 是文法 G 的一个项目集, X 为 G 的符号,
则 $GO(I, X) = \text{closure}(J)$ 。

其中

$J = \{\text{形如 } A \rightarrow \alpha X \cdot \beta \text{ 的项目} \mid \exists A \rightarrow \alpha \cdot X\beta \in I\}$ 。



构造LR(0)项目集规范族的方法（之二）

亦作为方法一的第一步

第一步：拓广文法

设文法 G 的开始符号是 S ，则 G 的拓广文法 G' 为：
在 G 的基础上增加一新的产生式 $S' \rightarrow S$ ，
 S' 为 G' 的开始符号。

目的：用来指示语法分析器什么时候应该停止分析并宣布接受输入(已把分析串归约到了 S)。



例，设文法 $G(S)$ 为：

$$S \rightarrow A \mid B \quad A \rightarrow aA \mid b \mid \varepsilon \quad B \rightarrow c$$

则文法 $G(S)$ 的拓广文法 $G'(S')$ 为

$$S' \rightarrow S$$

$$S \rightarrow A \mid B \quad A \rightarrow aA \mid b \mid \varepsilon \quad B \rightarrow c$$

例，设文法 $G(A)$ 为：

$$A \rightarrow aA \mid b$$

则文法 $G(A)$ 的拓广文法 $G'(S)$ 为

$$S \rightarrow A$$

$$A \rightarrow aA \mid b$$



第二步: 构造文法 G 的LR(0)项目集规范族。

利用函数closure和GO, 完成LR(0)项目集规范族的构造:

itemsets(G')

{ $C = \{ \text{closure}\{ S' \rightarrow \cdot S \} \}$ };

do{

if (对 C 的每个项目集 I 和每个文法符号 X ,
若 $\text{GO}(I, X)$ 非空且不在 C 中)

把 $\text{GO}(I, X)$ 加入 C 中;

} while (没有更多的项目集可以加入 C);

}

基本项目

利用构造LR(0)项目集规范族的方法 (之二)求
如下文法 $G(A)$ 的项目集规范族

$A \rightarrow aA \mid b$

拓广文法为: $S \rightarrow A \quad A \rightarrow aA \mid b$

$\text{closure}(S \rightarrow \cdot A) = \{S \rightarrow \cdot A, A \rightarrow \cdot aA \mid \cdot b\} = I_0$

$\text{GO}(I_0, A) = \text{closure}(S \rightarrow A \cdot) = \{S \rightarrow A \cdot\} = I_1$

$\text{GO}(I_0, a) = \text{closure}(A \rightarrow a \cdot A) = \{A \rightarrow a \cdot A, A \rightarrow \cdot aA \mid \cdot b\} = I_2$

$\text{GO}(I_0, b) = \text{closure}(A \rightarrow b \cdot) = \{A \rightarrow b \cdot\} = I_3$

$\text{GO}(I_2, A) = \text{closure}(A \rightarrow aA \cdot) = \{A \rightarrow aA \cdot\} = I_4$

$\text{GO}(I_2, a) = \text{closure}(A \rightarrow a \cdot A) = I_2$

$\text{GO}(I_2, b) = \text{closure}(A \rightarrow b \cdot) = I_3$



求得项目集规范族为:

$\{I_0, I_1, I_2, I_3, I_4\}$

$=\{\{S' \rightarrow \cdot A, A \rightarrow \cdot aA \mid \cdot b\}, \{S' \rightarrow A \cdot\},$

$\{A \rightarrow a \cdot A, A \rightarrow \cdot aA \mid \cdot b\}, \{A \rightarrow b \cdot\}, \{A \rightarrow aA \cdot\}\}$

项目集GO函数为:

	<i>a</i>	<i>b</i>	<i>A</i>
0	2	3	1
1			
2	2	3	4
3			
4			



应用求项目集规范族的方法二直接得出识别可归前缀的DFA $= \{Q, \Sigma, f, q_0, Z\}$ 。

1. $Q = C$

2. $\Sigma = V$

3. $f = \text{GO函数}$

4. q_0 为基本项目的闭包对应的状态。

5. 包含归约或接受项目的项目集构成终态集 Z 。



应用求项目集规范族的方法二直接求出识别可归前缀的DFA = $\{Q, V, f, 0, Z\}$ 。

1. 拓广文法。

2. 利用函数closure和GO, 构造DFA:

初始状态 $0 = \{ \text{closure}(S' \rightarrow \cdot S) \}$

do{

if (对每个状态*i* 和每个文法符号*X*

若 $GO(i, X) = j$ 非空且不在*Q*中)

把*j*加入*Q*中且 $f(i, X) = j$;

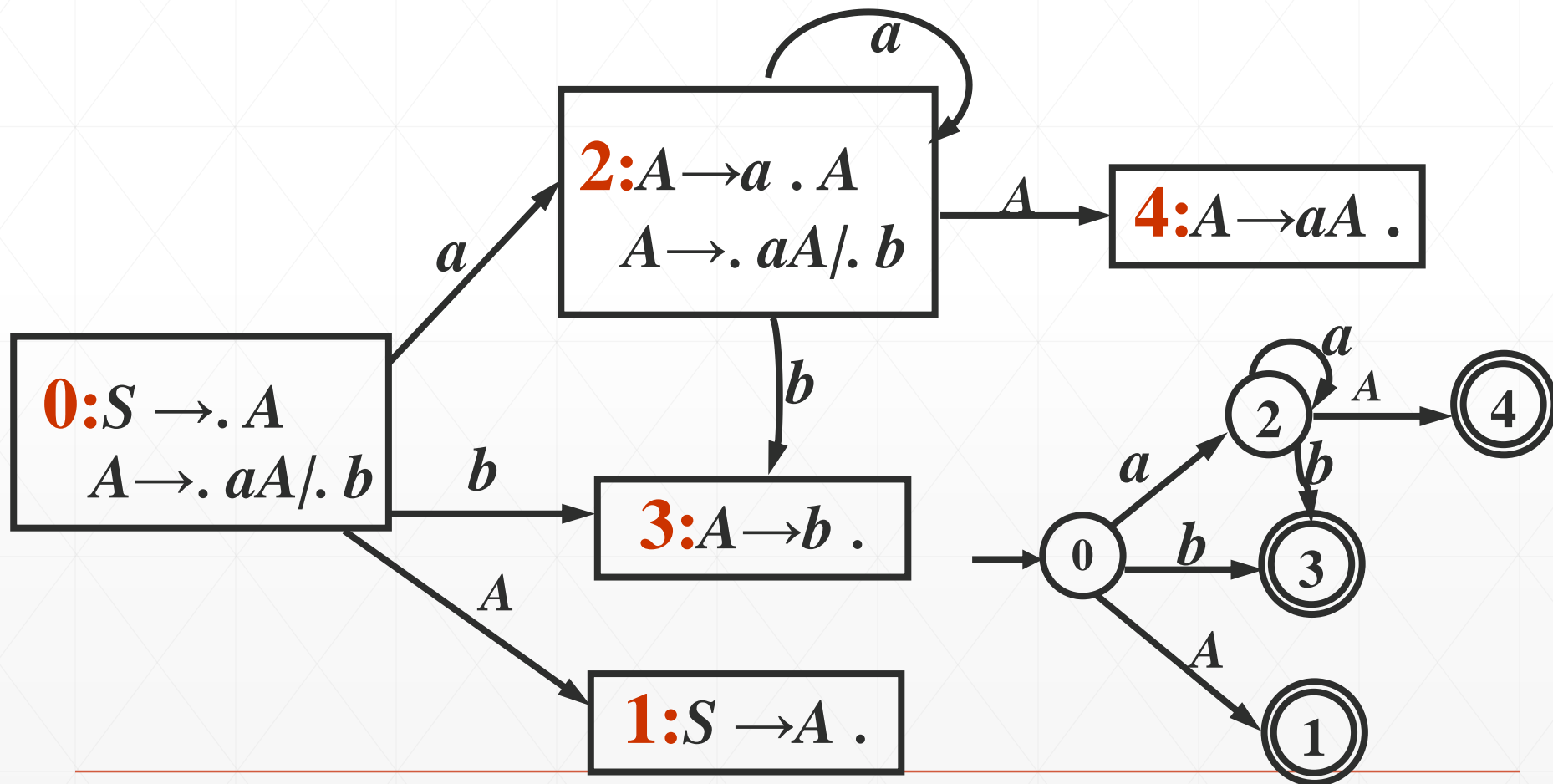
} while (没有更多的状态可以加入*Q*); }

3. 包含归约或接受项目的状态为终态。

$$A \rightarrow aA \mid b$$

应用求项目集规范族的方法二直接
求出识别可归前缀的DFA。

拓广文法为: $S \rightarrow A \quad A \rightarrow aA \mid b$



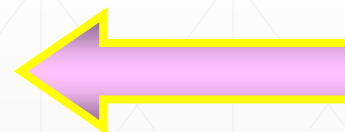


5.4 LR (0)分析

5.4.1 LR(0)分析实现思想

5.4.2 构造LR(0)项目集规范族

5.4.3 LR (0)分析表的构造





■ 定义

若一个文法 G 的识别可归前缀的DFA的每一个状态(项目集规范族中的每一个LR(0)项目集)中不存在

移进-归约冲突

① 即含移进项目又含归约项目;

或 ② 含有多个归约项目;

归约-归约冲突

则每个项目集的项目相容, 称 G 是一个LR(0)文法。



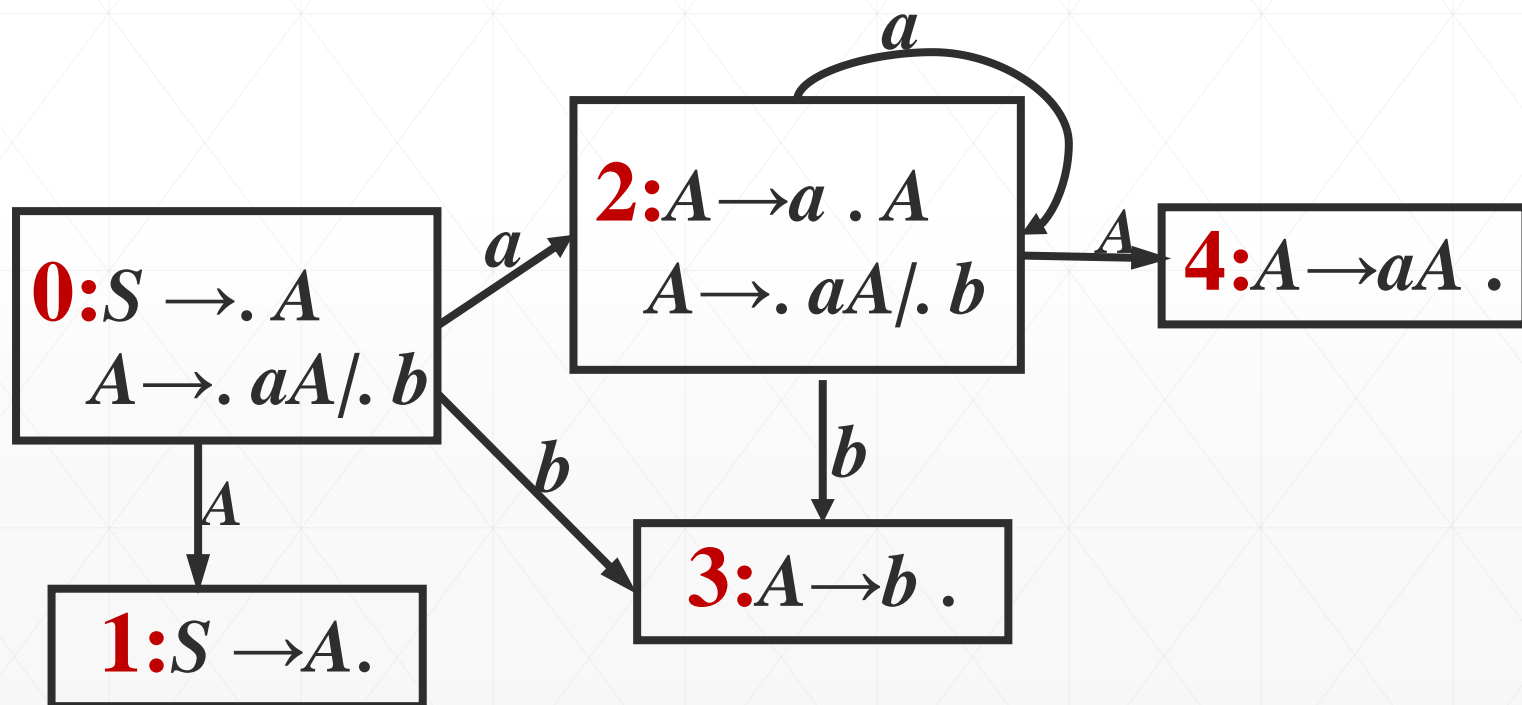
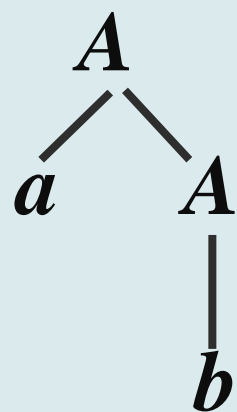
句子 ab 的归约:

$0 \rightarrow 2 \xrightarrow{\text{red}} 3$ ab (b 归约到 A)

$0 \rightarrow 2 \xrightarrow{\text{red}} 4$ aA (aA 归约到 A)

$0 \xrightarrow{\text{red}} 1$ A (分析完成)

$A \rightarrow aA \mid b$





识别可归前缀的DFA与LR(0)分析的对比

①在DFA中从状态 M 出发，经过一条 $a(a \in V_T)$ 弧到达状态 N ;

栈顶还没有出现句柄，分析表中 $\text{action}(M, a) = S_N$

②在DFA中从状态 M 出发，经过一条 $B(B \in V_N)$ 弧到达状态 N ;

栈顶归约出非终结符号，对分析表中 $\text{GOTO}(M, B) = N$

③在DFA中 M 为终态(含归约项目)，该状态中的文法产生式编号为 n ;

栈顶出现句柄，分析表中 $\text{action}(M, a) = r_n$;

④在DFA中 M 是终态(接受项目对应的状态)。

栈中为开始符号，分析表中 $\text{action}(M, \#) = \text{“acc”}$



从识别可归前缀的DFA构造LR(0)分析表

■ 算法：构造LR(0)分析表

输入：文法 G 识别可归前缀的DFA。

输出：文法 G 的LR(0)分析表

方法：

分析表的状态集为DFA的状态集

设为 $Q = \{0, 1, 2, \dots, n\}$

- ① 若 $f(K, a) = J (a \in V_T)$ ，则置
 $\text{action}(K, a) = S_J$ ；



②若 $f(K, A)=J$ ($A \in V_N$), 则置

$GOTO(K, A)=J$;

③若 $A \rightarrow \alpha \cdot$ (A 不是文法的开始符号) $\in K$ (含归约项目状态), 则对**所有**终结符 a 和结束符“#”,

置 $action(K, a)=r_j$ 和 $action(K, \#)=r_j$ 。

(其中假设产生式 $A \rightarrow \alpha \cdot$ 是文法第 j 个产生式)

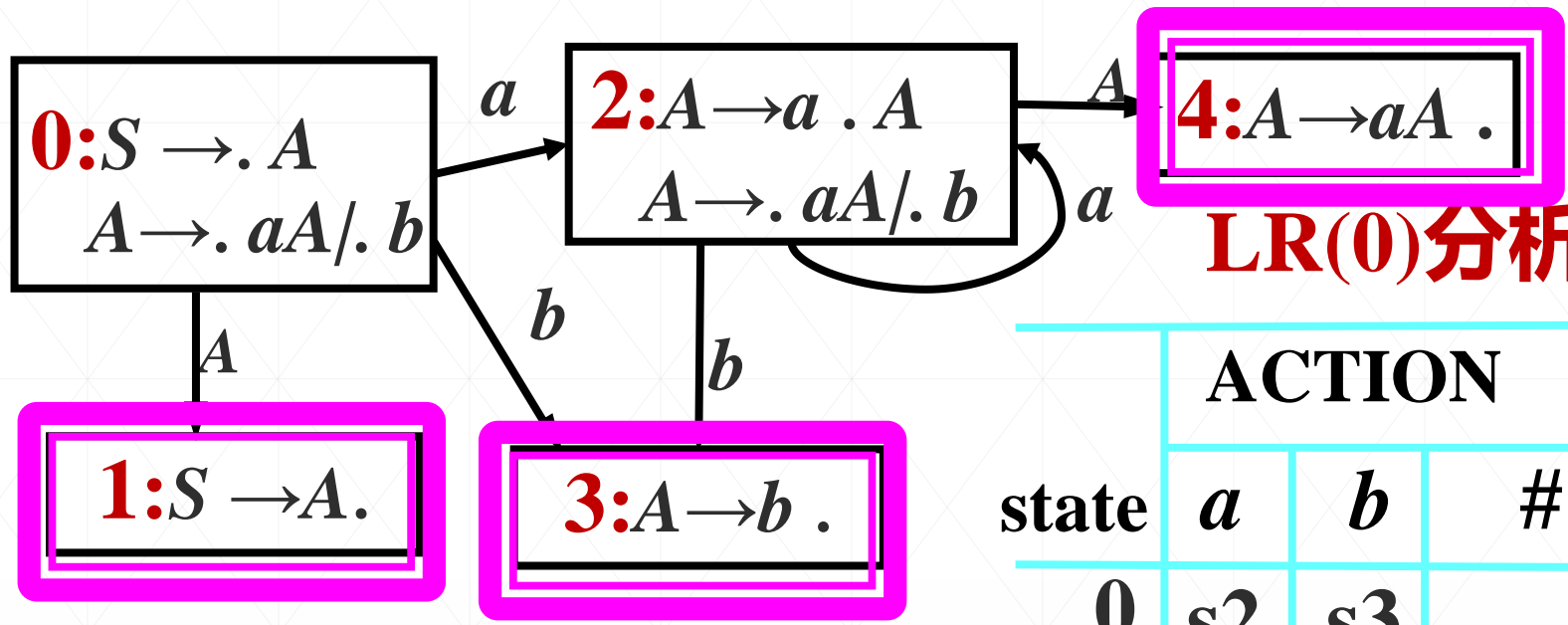
④ 若 $S' \rightarrow S \cdot$ (S' 是拓广文法的开始符号) $\in K$, (含接受项目状态)

置 $action(K, \#)=acc$

⑤ $action$ 表中空白置出错标志。



例:对前例文法G(A)有识别可归前缀的DFA如下



LR(0)分析表

state	ACTION			GOTO
	<i>a</i>	<i>b</i>	#	
0	s2	s3		1
1			acc	
2	s2	s3		4
3	r2	r2	r2	
4	r1	r1	r1	

文法是LR(0)文法

$S \rightarrow A$
 $A \rightarrow aA \text{ ① } \mid b \text{ ②}$



从LR(0)项目集规范族C和GO函数构造LR(0)分析表

■ 算法 构造LR(0)分析表

输入： 文法 G 和文法 G 的LR(0)项目集规范族
 C 和GO函数

输出： 文法 G 的LR(0)分析表

方法：

设 $C = \{I_0, I_1, \dots, I_n\}$ ，每个项目集 I_K
的下标 K 作为分析器的状态。

① 若 $GO(I_K, a) = I_J (a \in V_T)$ ，则置
 $action(K, a) = S_J$;



② 若 $\text{GO}(I_K, A) = I_J (A \in V_N)$, 则置

$\text{GOTO}(K, A) = J$;

③ 若 $A \rightarrow \alpha \cdot \in I_K$, 则对**所有**终结符 a 和结束符“#”, 置 $\text{action}(K, a) = r_j$ 和 $\text{action}(K, \#) = r_j$ 。

(其中假设产生式 $A \rightarrow \alpha \cdot$ 是文法第 j 个产生式并且 **A 不是文法的开始符号**)

④ 若 $S' \rightarrow S \cdot \in I_K$, **S' 是文法的开始符号**, 则置 $\text{action}(K, \#) = \text{acc}$;

⑤ action 表中空白置出错标志。



对例5.12的文法 $G(S')$ 有LR(0)项目集规范族及GO函数为:

$I_0: \{S' \rightarrow \cdot A, A \rightarrow \cdot aA, A \rightarrow \cdot b\}$

$I_1: \{S' \rightarrow A \cdot\}$

$I_2: \{A \rightarrow a \cdot A, A \rightarrow \cdot aA, A \rightarrow \cdot b\}$

$I_3: \{A \rightarrow b \cdot\}$

$I_4: \{A \rightarrow aA \cdot\}$

$S' \rightarrow A$

$A \rightarrow aA \text{ ①} \mid b \text{ ②}$

GO函数:

文法 $G(S')$ 的LR(0)分析表

state	ACTION表			GOTO表
	a	b	$\#$	A
0	s2	s3		1
1			acc	
2	s2	s3		4
3	r2	r2	r2	
4	r1	r1	r1	

	a	b	A
0	2	3	1
1			
2	2	3	4
3			
4			



例：有文法 $G(S)$:

$$S \rightarrow aABe$$

$$A \rightarrow Abc / b$$

$$B \rightarrow d$$

分析串 $abbcd e$ 是该文法的合法句子。



$S \rightarrow aABe$
 $A \rightarrow Abc/b$
 $B \rightarrow d$

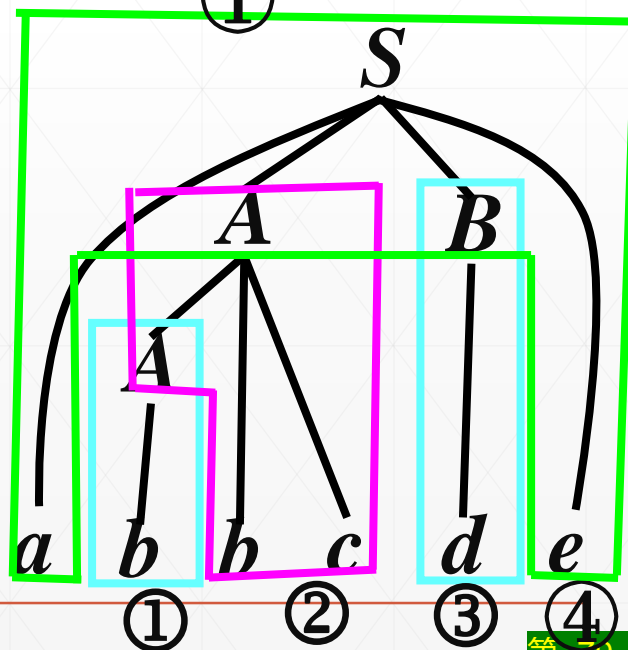
规范推导:

$S \Rightarrow \underline{aABe}$
 ④

$\Rightarrow aA\underline{de}$
 ③

$\Rightarrow a\underline{Abc}de$
 ②

$\Rightarrow \underline{abbcde}$
 ①



步骤	分析栈	待分析串	动作
初始	#	abbcde #	移进
(1)	# a	bbcde #	移进
(2)	# ab	bcde #	$A \rightarrow b$ 归约
(3)	# aA	bcde #	移进
(4)	# aAb	cde #	移进
(5)	# aAbc	de #	$A \rightarrow Abc$ 归约
(6)	# aA	de #	移进
(7)	# aAd	e #	$B \rightarrow d$ 归约
(8)	# aAB	e #	移进
(9)	# aABe	#	$S \rightarrow aABe$ 归约
(10)	# S	#	分析成功



构造文法*G*的识别可归前缀的DFA

拓广文法为:

$$S' \rightarrow S$$

$$S \rightarrow aABe \textcircled{1}$$

$$A \rightarrow Abc \textcircled{2} / b \textcircled{3}$$

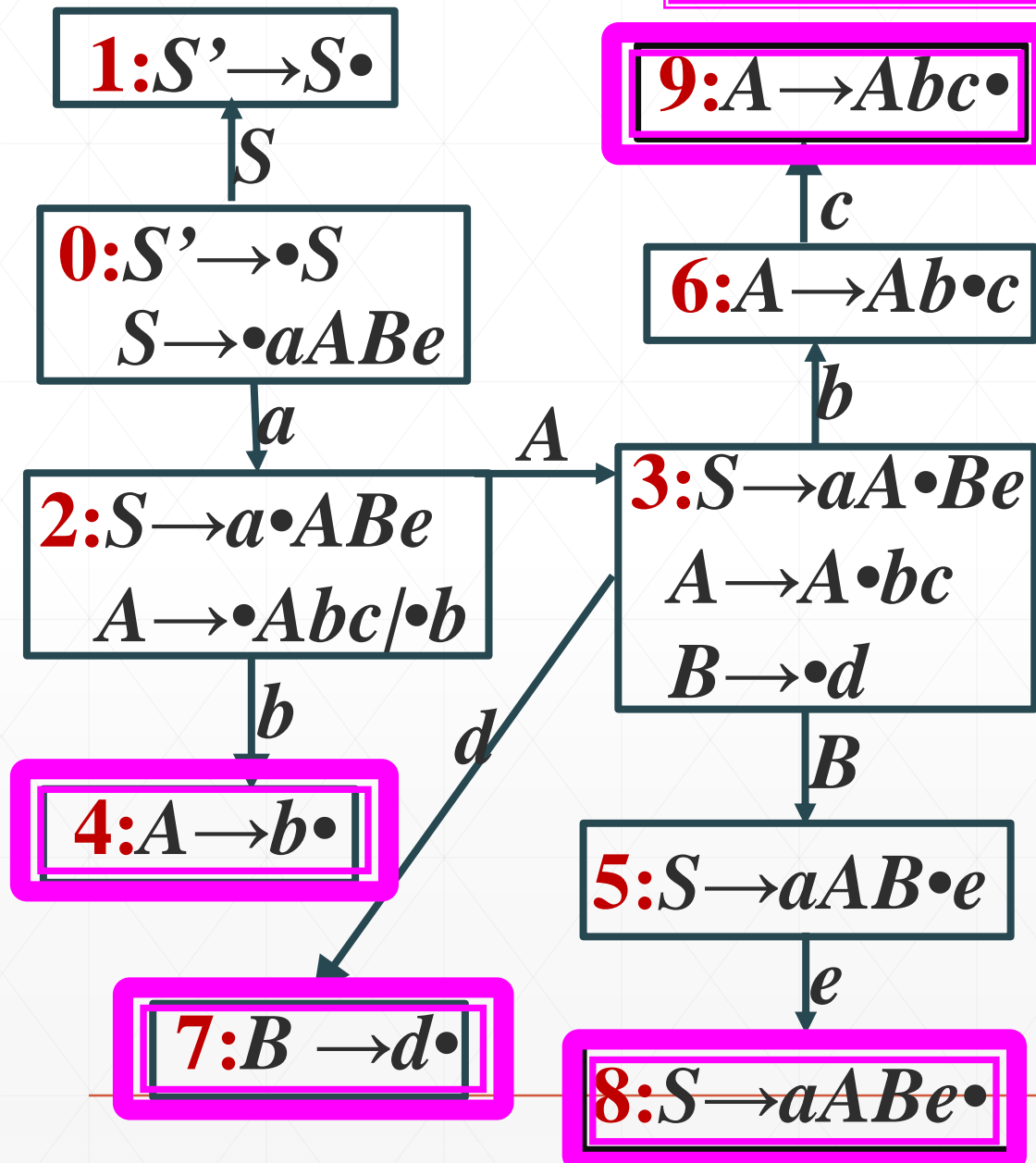
$$B \rightarrow d \textcircled{4}$$



识别可归前缀的DFA

 $S' \rightarrow S \quad S \rightarrow aABe \textcircled{1} \quad A \rightarrow Abc \textcircled{2}/b \textcircled{3} \quad B \rightarrow d \textcircled{4}$

LR(0)分析表



	ACTION						GOTO	
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	#	<i>A</i>	<i>B</i> <i>S</i>
0	S_2							1
1						<i>acc</i>		
2		S_4					3	
3		S_6		S_7				5
4	r_3	r_3	r_3	r_3	r_3	r_3		
5					S_8			
6			S_9					
7	r_4	r_4	r_4	r_4	r_4	r_4		
8	r_1	r_1	r_1	r_1	r_1	r_1		
9	r_2	r_2	r_2	r_2	r_2	r_2		



LR(0)分析表

	ACTION						GOTO		
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	#	<i>A</i>	<i>B</i>	<i>S</i>
0	S_2								1
1						<i>acc</i>			
2		S_4					3		
3		S_6		S_7				5	
4	r_3	r_3	r_3	r_3	r_3	r_3			
5					S_8				
6			S_9						
7	r_4	r_4	r_4	r_4	r_4	r_4			
8	r_1	r_1	r_1	r_1	r_1	r_1			
9	r_2	r_2	r_2	r_2	r_2	r_2			

步骤	分析栈	待分析串	动作
初始	0 #	abbcde #	移进
(1)	02 #a	bbcde #	移进
(2)	024 #ab	bcde #	$A \rightarrow b$ 归约
(3)	023 #aA	bcde #	移进
(4)	0236 #aAb	cde #	移进
(5)	02369 #aAbc	de #	$A \rightarrow Abc$ 归约
(6)	023 #aA	de #	移进
(7)	0237 #aAd	e #	$B \rightarrow d$ 归约
(8)	0235 #aAB	e #	移进
(9)	02358 #aABe	#	$S \rightarrow aABe$ 归约
(10)	01 #S	#	分析成功

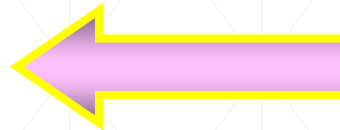


5.1 “移近—归约”分析法

5.3 LR分析鸟瞰

5.4 LR (0)分析

5.5 SLR (1)分析



5.6 LR (1)分析

5.7 LALR (1)分析

5.8 LR分析对二义文法的应用

5.9 LR分析的错误处理与恢复

5.10 语法分析器的自动生成与YACC



例: $G: A \rightarrow aA \mid a$

构造文法 G 的LR(0)分析表。

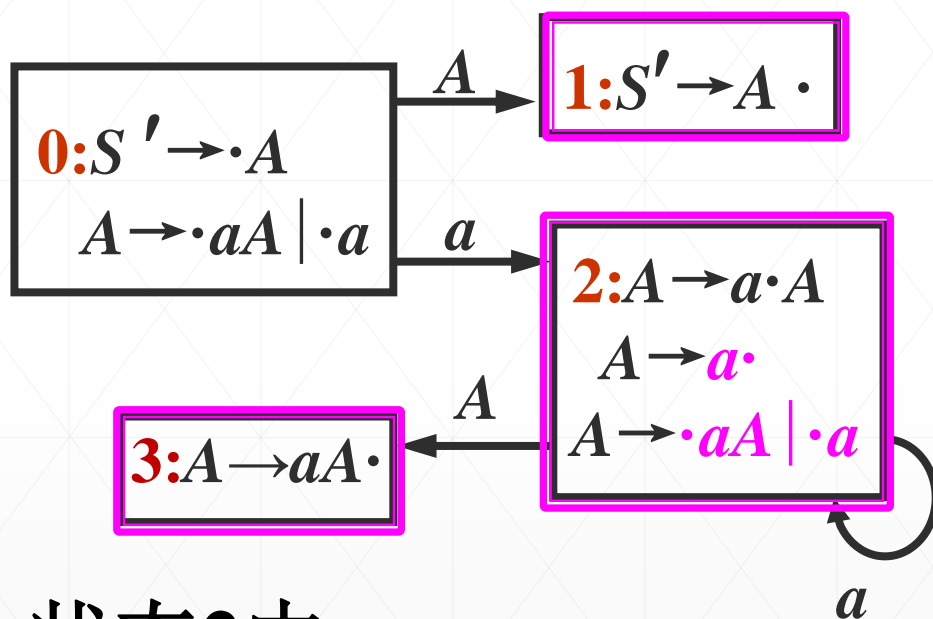
首先文法 G 拓广为

$G': S' \rightarrow A$

$A \rightarrow aA \text{ ①} \mid a \text{ ②}$



构造文法 G' 的识别可归前缀的DFA

 $S' \rightarrow A$
 $A \rightarrow aA \text{ ①} \mid a \text{ ②}$


state	action		goto
	a	$\#$	
0	S_2		1
1		acc	
2	r_2 S_2	r_2	3
3	r_1	r_1	

状态2中:

既有移进项 $A \rightarrow \cdot aA \mid \cdot a$, 又有归约项 $A \rightarrow a \cdot$,

文法 G' 是非LR(0)文法



在识别可归前缀的DFA某一状态(项目集规范族的某一项目集)中,

1. 若既含有移进项目, 又含有归约项目, 该状态(项目集)存在**移进—归约冲突**。
2. 若含有两个或两个以上的归约(接受)项目, 该状态(项目集)存在**归约—归约冲突**。



注意:

冲突情况都与归约动作相关。



■ 分析:

LR(0)分析表构造算法对含有归约项目 $A \rightarrow \beta \cdot$ 的项目集（状态） I_i ，不管当前输入符号为何，皆把action子表相应于状态 i 的那一行的诸元素都指定为 r_j （其中 j 为产生式 $A \rightarrow \beta \cdot$ 的编号）。

	a_1	a_2	...	a_n
...				
i	r_j	r_j	...	r_j
...				



$$\text{action}(i, a) = \{A \rightarrow \beta\} \quad (\forall a \in V_T \cup \{\#\})$$

归约未必有效!

∴ 可能不存在一个规范句型为... Aa ...形式。

∴ 用 $A \rightarrow \beta$ 归约不一定有效。

■ 定义 (回顾)

设上下文无关文法 G , S 是文法的开始符号,
对于文法 G 的任何非终结符 A

$$\text{FOLLOW}(A) = \{a \mid S \Rightarrow \dots Aa\dots, a \in V_T\}$$

若 $S \Rightarrow \dots A$, 则令 $\# \in \text{FOLLOW}(A)$ 。



对含有归约项目的项目集 I_i :

$$I_i = \{A \rightarrow \beta \cdot, B \rightarrow \delta \cdot, \dots\dots\}$$

填写归约动作时, 首先求解FOLLOW(A), FOLLOW(B), 再对任何输入符号 a :

(1) 当 $a \in \text{FOLLOW}(A)$ 时, 置

action (i, a) = {按产生式 $A \rightarrow \beta$ 归约} ;

(2) 当 $a \in \text{FOLLOW}(B)$ 时, 置

action (i, a) = {按产生式 $B \rightarrow \delta$ 归约} ;

(3) 当 a 不属于上述情况时, 不填写归约动作。

SLR(1)冲突解决方法, SLR(1)规则。



从识别可归前缀的DFA构造SLR(1)分析表

■ 算法 构造SLR(1)分析表

输入：文法 G 及其识别可归前缀的DFA。

输出：文法 G 的SLR(1)分析表

方法：

分析表的状态集为DFA的状态集

设为 $Q = \{0, 1, 2, \dots, n\}$

- ① 若 $f(K, a) = J (a \in V_T)$ ，则置
 $\text{action}(K, a) = S_J$ ；



② 若 $f(K, A)=J$ ($A \in V_N$), 则置

$GOTO(K, A)=J$;

③ 若 $A \rightarrow \alpha \cdot \in K$, 则对所有 $a \in FOLLOW(A)$,

置 $action(K, a)=r_j$ 。

(其中假设产生式 $A \rightarrow \alpha \cdot$ 是文法第 j 个产生式并且 A 不是文法的开始符号)

④ 若 $S' \rightarrow S \cdot \in K$, S' 是文法的开始符号,
则置 $action(K, \#)=acc$;

⑤ 表中空白置出错标志。



从LR(0)项目集规范族C和GO函数构造SLR(1)分析表

■ 算法 构造SLR(1)分析表

输入：文法G和文法G的LR(0)项目集规范族C和GO函数

输出：文法G的SLR(1)分析表

方法：

设 $C = \{I_0, I_1, \dots, I_n\}$ ，每个项目集 I_K 的下标 K 作为分析器的状态。

- ① 若 $GO(I_K, a) = I_J (a \in V_T)$ ，则置 $action(K, a) = S_J$ ；



■ ② 若 $GO(I_K, A) = I_J (A \in V_N)$, 则置

$GOTO(K, A) = J$;

③ 若 $A \rightarrow \alpha \cdot \in I_K$, 则对 **所有 $a \in FOLLOW(A)$** ,

置 $action(K, a) = r_j$ 。

(其中假设产生式 $A \rightarrow \alpha \cdot$ 是文法第 j 个产生式并且 A 不是文法的开始符号)

④ 若 $S' \rightarrow S \cdot \in I_K$, S' 是文法的开始符号,

则置 $action(K, \#) = acc$;

⑤ 表中空白置出错标志。



■ 定义

如果文法 G 的SLR(1)分析表不含多重定义入口，则称文法 G 为SLR(1)文法。

使用SLR(1)分析表的语法分析器称作SLR(1)分析器。

SLR(1)分析表构造= SLR(1)方法+ LR(0)分析表构造

SLR(1)方法 = 有归约项目的状态中归约动作的设置。



例5.14 设有文法 G 以及识别可归前缀的DFA如下

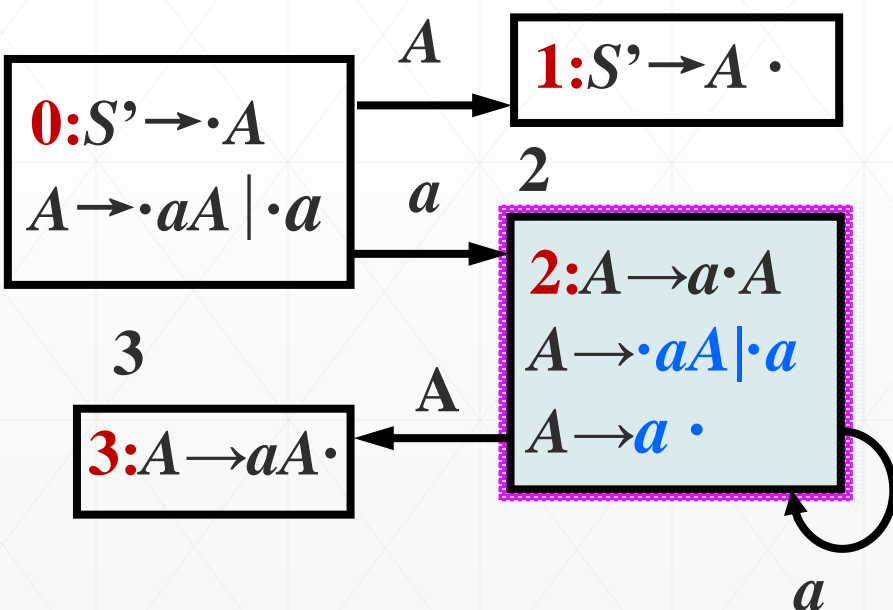
G' : $S' \rightarrow A$

$A \rightarrow aA$ ①

$A \rightarrow a$ ②

FOLLOW(A) = {#}

G' 的SLR(1)分析表



state	action		goto
	a	$\#$	
0	S_2		1
1		acc	
2	S_2	r_2	3
3		r_1	

该文法为SLR(1)文法。



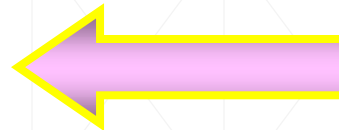
5.1 “移近—归约”分析法

5.3 LR分析鸟瞰

5.4 LR (0)分析

5.5 SLR (1)分析

5.6 LR (1)分析



5.7 LALR (1)分析

5.8 LR分析对二义文法的应用

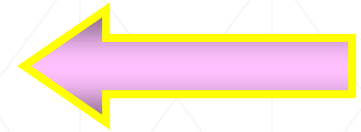
5.9 LR分析的错误处理与恢复

5.10 语法分析器的自动生成与YACC



5.6 LR (1)分析

5.6.1 SLR(1)分析的缺陷



5.6.2 LR(1)分析的实现思想

5.6.3 LR (1)项目集规范族的构造

5.6.4 LR (1)分析表的构造



例：设有文法 $G(S)$ 为

$$S \rightarrow L=R / R$$

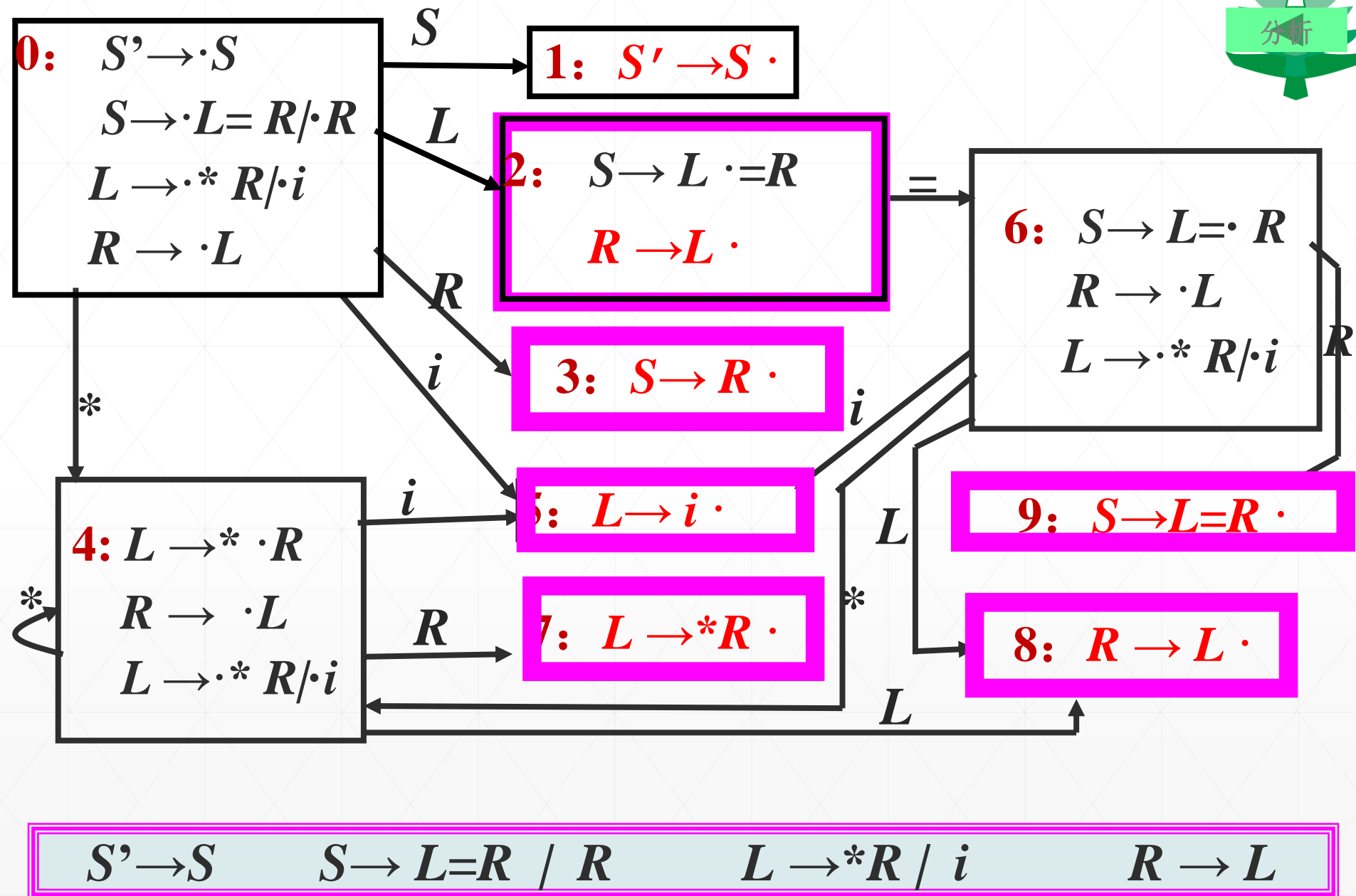
$$L \rightarrow *R / i$$

$$R \rightarrow L$$

构造文法 $G(S)$ 的识别可归前缀的**DFA**：

首先文法拓广为

$G'(S')$ ，增加一产生式 $S' \rightarrow S$





状态2 = $\{ S \rightarrow L \cdot = R, R \rightarrow L \cdot \}$

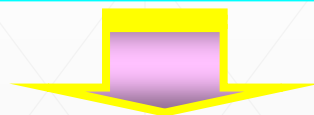
存在移进-归约冲突，据SLR(1)方法：

$\text{FOLLOW}(R) = \{ =, \# \}$

有 $\text{FOLLOW}(R) \cap \{ = \} \neq \Phi$

\therefore 文法G是非SLR(1)文法

对SLR(1)分析，存在 $I_k : A \rightarrow \beta \cdot$



若 $a \in \text{FOLLOW}(A) \Rightarrow \text{action}(k, a) = \{ A \rightarrow \beta \}$

$S' \rightarrow S \quad S \rightarrow L = R \mid R \quad L \rightarrow * R \mid i \quad R \rightarrow L$

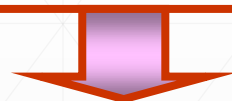


$$\text{action}(k, a) = \{A \rightarrow \beta\}$$

归约未必有效!

- ∴ 可能不存在一个规范句型含有前缀 $\delta A a$
- ∴ 用 $A \rightarrow \beta$ 归约不一定有效。

$$\text{FOLLOW}(A) = \{a \mid S \xRightarrow{*} \dots A a \dots, a \in V_T\}$$



此推导不考虑A前面的符号串

$A \leftarrow$

β	$\dots k$
δ	\dots
$\#$	0

$S: \dots a \dots \#$



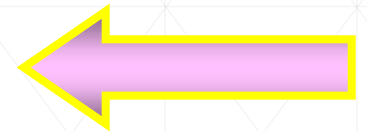
得到 $\delta A a \dots$ 形式的规范句型。



5.6 LR (1)分析

5.6.1 SLR(1)分析的缺陷

5.6.2 LR(1)分析的实现思想



5.6.3 LR (1)项目集规范族的构造

5.6.4 LR (1)分析表的构造



■ 定义（LR(k)项目）

文法 G 的一个LR(k)项目是

$$\underline{[A \rightarrow \alpha \cdot \beta, \quad a_1 a_2 \dots a_k \quad]}$$

其中：

$A \rightarrow \alpha \cdot \beta$ 是一个LR(0)项目；

$a_i \in V_T \cup \{\#\}$ ；

$a_1 a_2 \dots a_k$ ：搜索字符串；



■ 定义（LR(1)有效项目）

若文法 G 的一个LR(1)项目 $[A \rightarrow \alpha \cdot \beta, a]$ 对活前缀 γ 是有效的，当且仅当存在规范推导

$$S \xrightarrow{*} \delta A \omega \xrightarrow{\mathbf{R}} \underline{\delta \alpha} \beta \omega$$

其中： $\omega \in V_T^*$ ， $\gamma = \delta \alpha$ ， $a \in \text{FIRST}(\omega)$ 或 a 为‘#’（当 $\omega = \varepsilon$ ），称 a 为搜索符。





例: 设有文法G

$$S \rightarrow BB \quad B \rightarrow aB \mid b$$

LR(1)项目 $[B \rightarrow a.B, a]$ 对活前缀 aaa 有效

\exists 规范推导 $S \Rightarrow BB \Rightarrow BaB \Rightarrow Bab \Rightarrow aBab$

$\Rightarrow aaBab$ $\Rightarrow a a \boxed{a B} ab$
 δ ω $\delta\alpha$ $\alpha\beta$ ω

LR(1)项目 $[B \rightarrow .aB, a]$ 对活前缀 aa 有效

当 $\omega = \varepsilon$

LR(1)项目 $[B \rightarrow a.B, \#]$ 对活前缀 Baa 有效

\exists 规范推导 $S \Rightarrow BB \Rightarrow BaB \Rightarrow Ba \boxed{aB}$



例如，对例5.15 有

$$I_2 = \{ S \rightarrow L \cdot = R, R \rightarrow L \cdot \}$$

$$\text{FOLLOW}(R) = \{ =, \# \}$$

从 $[R \rightarrow L \cdot, \#]$ 项目考察知，它对 L 有效。

而 $[R \rightarrow L \cdot, =]$ 项目考察知，它对 L 无效。

$$\exists S' \Rightarrow S \Rightarrow R \Rightarrow L$$

$$\exists S' \Rightarrow S \Rightarrow L = R$$

等号前不可能只出现一个 R
即不可能有规范句型 $R = \omega$

 $S' \rightarrow S$
 $S \rightarrow L = R \mid R$
 $L \rightarrow *R \mid i$
 $R \rightarrow L$



■ LR(0)有效项目

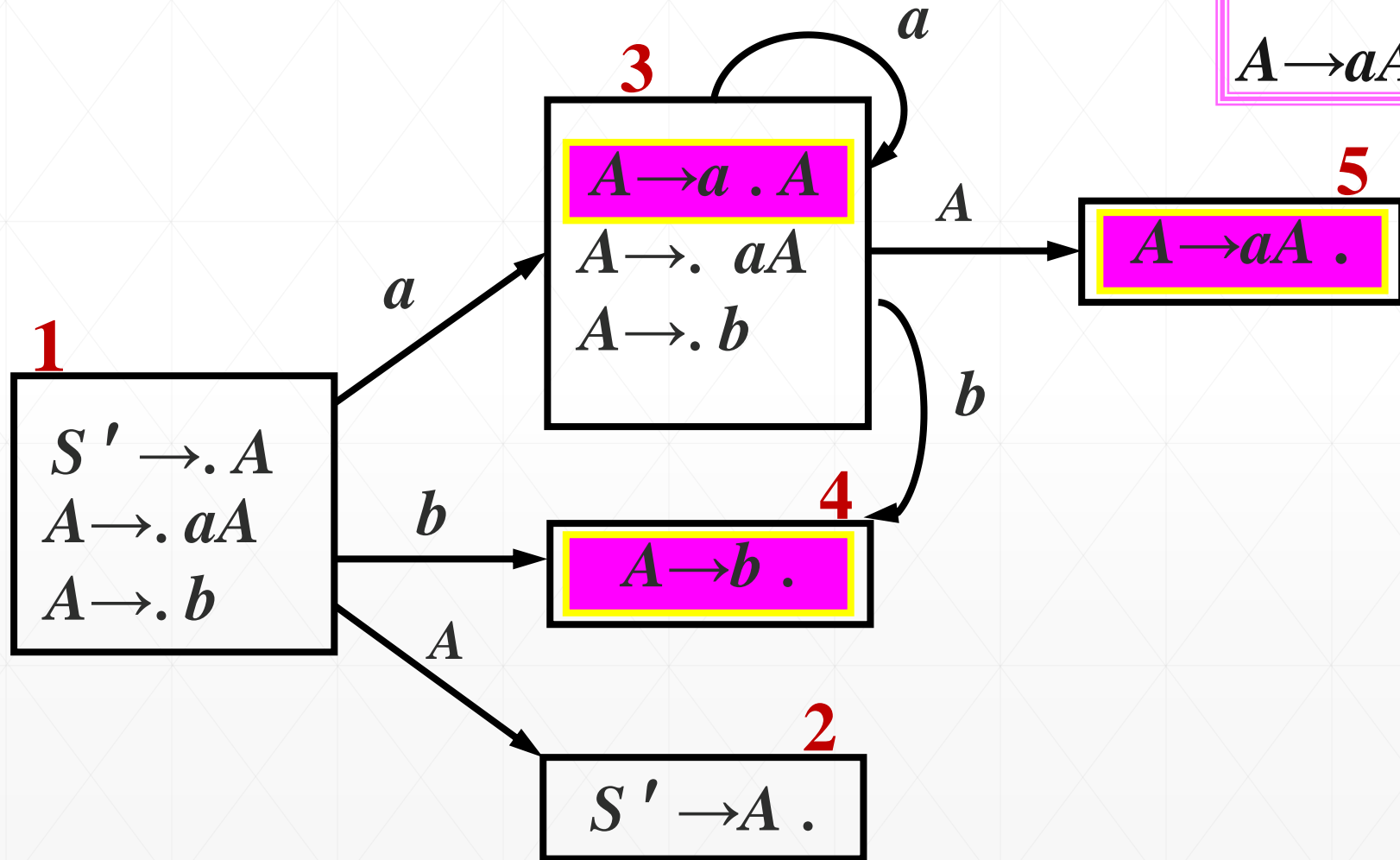
若文法 G 的一个LR(0)项目 $[A \rightarrow \beta_1 \cdot \beta_2]$

对活前缀 $\delta\beta_1$ 是有效的，当且仅当存在规范
推导

$$S \xRightarrow{*}_R \delta A \omega \xRightarrow{*}_R \underline{\delta\beta_1} \beta_2 \omega$$



例如，对例5.12的文法 $G(S')$ 有识别可归前缀的DFA如下：

 $S' \rightarrow A$ $A \rightarrow aA \mid b$

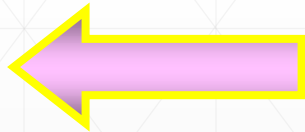


5.6 LR (1)分析

5.6.1 SLR(1)分析的缺陷

5.6.2 LR(1)分析的实现思想

5.6.3 LR (1)项目集规范族的构造



5.6.4 LR (1)分析表的构造



■ 构造LR(1)项目集规范族C

两个函数

1: 函数 $\text{closure}(I)$;

2: GO函数 ;



■ 算法5.7 closure (I)

* 注意：待约项目扩展的项目的搜索符的求法。

```
{  
    do {  
        if ( 对  $I$  的每个项目  $[A \rightarrow \alpha \cdot B\beta, a]$ , 及文法中的  
        每个产生式  $B \rightarrow \gamma$  和  $\text{FIRST}(\beta a)$  的每个符号  $b$ ,  
        若项目  $[B \rightarrow \bullet \gamma, b]$  不在  $I$  中)  
            则把  $[B \rightarrow \bullet \gamma, b]$  加到  $I$  中 ;  
        } while ( 没有更多的项目可以加入  $I$  );  
    return  $I$  ;  
}
```



■ 算法5.8

 $\text{GO}(I, X)$

{

令 $J = \{ [A \rightarrow \alpha X \bullet \beta, a] \mid [A \rightarrow \alpha \bullet X \beta, a] \in I \}$;return closure(J);

}

LR(1)项目集 I 的GO函数： $\text{GO}(I, X)$ 是 I 中LR(0)的项目圆点右移一个位置的项目且搜索符不变，然后对项目集求closure。



■ 算法5.9

$\text{items}(G')$

//LR(1)的C的构造

{

$C = \text{closure}(\{S' \rightarrow \bullet S, \# \});$ //初始化

do {

if (对C的每个项目集I和每个文法符号
X, 若 $\text{GO}(I, X)$ 非空且不在C中)

把 $\text{GO}(I, X)$ 加入C中;

} while (没有更多的项目集可以加入C中);

}



■ 算法5.9'

构造识别LR(1)项目有效可归前缀的DFA

{

初态0= closure ($\{S' \rightarrow \bullet S, \# \}$); //初始化

do {

 if (对每个状态*i*和每个文法符号
 X, 若GO(*i*, *X*)=*j* 非空且不在DFA中)

*j*加入DFA中且*f*(*i*, *X*)=*j*;

 } while (没有更多的状态可以加入DFA中);

}



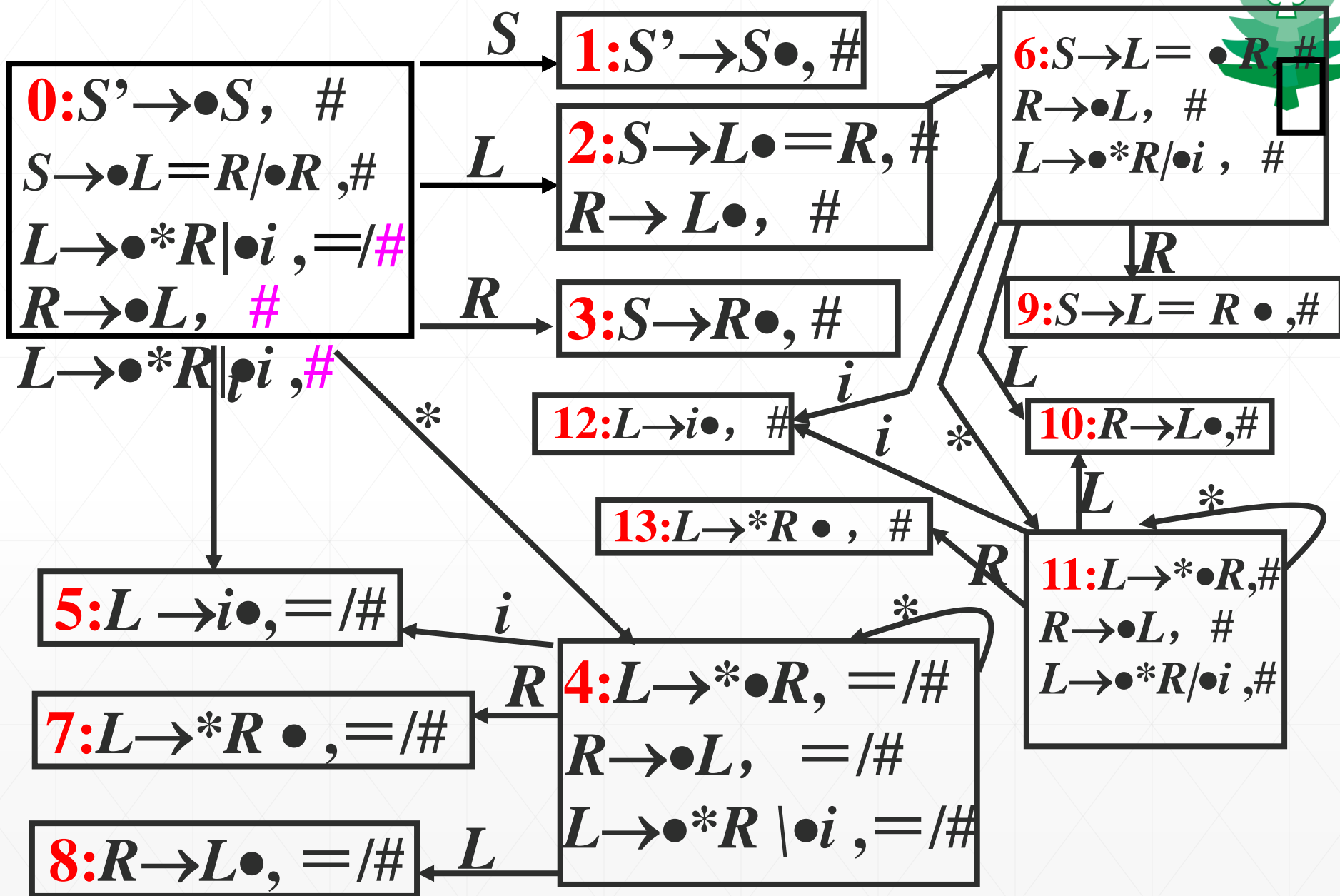
例5.15 $S' \rightarrow S$

$S \rightarrow L=R$ ①/ R ②

$L \rightarrow *R$ ③/ i ④

$R \rightarrow L$ ⑤

构造识别LR(1)项目有效可归前缀的DFA。



$S' \rightarrow S$ $S \rightarrow L = R$ ① / R ② $L \rightarrow *R$ ③ / i ④ $R \rightarrow L$ ⑤



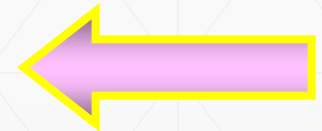
5.6 LR (1)分析

5.6.1 SLR(1)分析的缺陷

5.6.2 LR(1)分析的实现思想

5.6.3 LR (1)项目集规范族的构造

5.6.4 LR (1)分析表的构造





■ 算法 (LR(1)分析表构造)

输入： 文法 G 和文法 G 的识别LR(1)项目
有效可归前缀的DFA。

输出： 文法 G 的LR(1)分析表

方法： 设 G 的识别LR(1)项目有效可归前缀的
DFA的

$$Q = \{0, 1, \dots, n\}$$

令 Q 的每个状态 k 为分析表的状态。 初始
状态0为分析表的初态。

则有：



- ① 对于每个状态 k , 若 $f(k, a)=j$ ($a \in V_T$),
则置 $\text{Action}(k, a)=S_j$; 若 $f(k, A)=j$ ($A \in V_N$),
则置 $\text{GOTO}(k, A)=j$;
- ② 若归约项目 $[A \rightarrow \alpha \cdot, a] \in k$, 则置 $\text{action}[k, a]=r_j$ 。
(其中假设产生式 $A \rightarrow \alpha \cdot$ 是文法第 j 个产生式并且
 A 不是文法的开始符号)
- ③ 若接受项目 $[S' \rightarrow S \cdot, \#] \in k$, S' 是文法的开始符号,
则置 $\text{action}(k, \#)=\text{acc}$;
- ④ 对分析表中不能按上述规则填入信息的元素,
则置“出错”标志。



■ 算法 (LR(1)分析表构造)

输入： 拓广的文法 G' 和文法 G' 的LR(1)项目集规范族 C 和GO函数

输出： 文法 G' 的LR(1)分析表

方法： 设 G' 的LR(1)项目集规范族

$$C = \{I_0, I_1, \dots, I_n\}$$

令每个 I_k 的下标 k 为分析表的状态。 并含有 $[S' \rightarrow \cdot S, \#]$ 的项目集为分析表的初态。

则有：

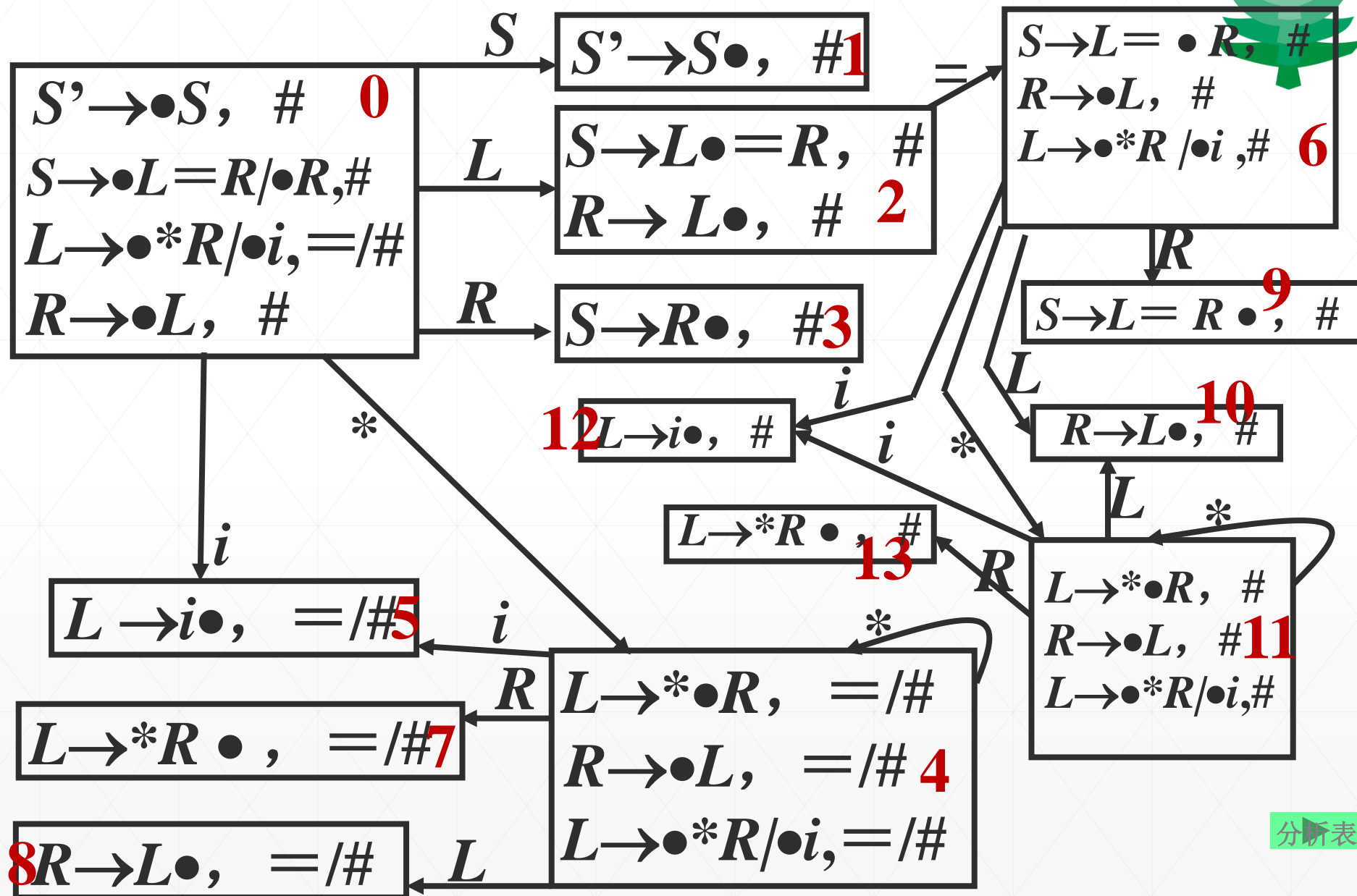


- ①对于每个项目集 I_i 中形如 $[A \rightarrow \alpha \cdot X \beta, b]$ 的项目, 若 $GO(I_i, X) = I_j$, 且 $X \in V_T$, 置 $action[i, X] = S_j$ 。
若 $X \in V_N$ 时, 则置: $GOTO[i, X] = j$ 。
- ②若归约项目 $[A \rightarrow \alpha \cdot a] \in I_i$, $A \rightarrow \alpha$ 为文法的第 j 个产生式并且 A 不是文法的开始符号, 则置 $action[i, a] = r_j$ 。
- ③若接受项目 $[S' \rightarrow S \cdot, \#] \in I_i$, S' 是文法的开始符号, 则置 $action[i, \#] = \text{"acc"}$ 。
- ④ 对分析表中不能按上述规则填入信息的元素, 则置“出错”标志。



■ 定义

按照LR(1)的项目集规范族（识别LR(1)项目有效可归前缀的DFA）构造的文法 G 的LR(1)分析表，如果每个入口不含多重定义，则称文法 G 为LR(1)文法。使用LR(1)分析表的语法分析器称作LR(1)分析器。



分析表



LR (1) 分析表

state	action				goto		
	=	*	<i>i</i>	#	L	R	S
0		S ₄	S ₅		2	3	1
1				acc			
2	S ₆			r5			
3				r2			
4		S ₄	S ₅		8	7	
5	r4			r4			
6		S ₁₁	S ₁₂		10	9	
7	r3			r3			
8	r5			r5			
9				r1			
10				r5			
11		S ₁₁	S ₁₂		10	13	
12				r4			
13				r3			



例：设有文法 $G(A)$

$$A \rightarrow BB$$

$$B \rightarrow aB/b$$

构造该文法的LR(1)分析表

首先对文法进行拓广

$$S' \rightarrow A$$

$$A \rightarrow BB \text{ ①}$$

$$B \rightarrow aB \text{ ②}/b \text{ ③}$$



构造文法的LR(1)项目集规范族

$$\{S' \rightarrow \cdot A, \#; A \rightarrow \cdot BB, \#; B \rightarrow \cdot aB / \cdot b, a/b\} \quad I_0$$

$$\{S' \rightarrow A \cdot, \#\} \quad I_1$$

$$\{A \rightarrow B \cdot B, \#; B \rightarrow \cdot aB / \cdot b, \#\} \quad I_2$$

$$\{B \rightarrow a \cdot B, a/b; B \rightarrow \cdot aB / \cdot b, a/b\} \quad I_3$$

$$\{B \rightarrow b \cdot, a/b\} \quad I_4$$

$$\{A \rightarrow BB \cdot, \#\} \quad I_5$$

$$\{B \rightarrow a \cdot B, \#; B \rightarrow \cdot aB / \cdot b, \#\} \quad I_6$$

$$\{B \rightarrow b \cdot, \#\} \quad I_7$$

$$\{B \rightarrow aB \cdot, a/b\} \quad I_8$$

$$\{B \rightarrow aB \cdot, \#\} \quad I_9$$

$$S' \rightarrow A$$

$$A \rightarrow BB \quad \textcircled{1}$$

$$B \rightarrow aB \textcircled{2} / b \textcircled{3}$$

GO函数

	<i>a</i>	<i>b</i>	<i>B</i>	<i>A</i>
0	3	4	2	1
2	6	7	5	
3	3	4	8	
6	6	7	9	

1,4,5,7,8,9是归约态。



LR(1)分析表

 $\{S' \rightarrow A \cdot, \# \} I_1$
 $\{B \rightarrow b \cdot, a/b \} I_4$
 $\{A \rightarrow BB \cdot, \# \} I_5$
 $\{B \rightarrow b \cdot, \# \} I_7$
 $\{B \rightarrow aB \cdot, a/b \} I_8$
 $\{B \rightarrow aB \cdot, \# \} I_9$
 $S' \rightarrow A$
 $A \rightarrow BB \text{ ①}$
 $B \rightarrow aB \text{ ②} / b \text{ ③}$

	<i>a</i>	<i>b</i>	<i>B</i>	<i>A</i>
0	3	4	2	1
2	6	7	5	
3	3	4	8	
6	6	7	9	

state	ACTION表			GOTO表	
	<i>a</i>	<i>b</i>	<i>#</i>	<i>B</i>	<i>A</i>
0	S ₃	S ₄		2	1
1			acc		
2	S ₆	S ₇		5	
3	S ₃	S ₄		8	
4	r ₃	r ₃			
5			r ₁		
6	S ₆	S ₇		9	
7			r ₃		
8	r ₂	r ₂			
9			r ₂		



end

结束即是新的开始
THE END IS THE BEGINNING