

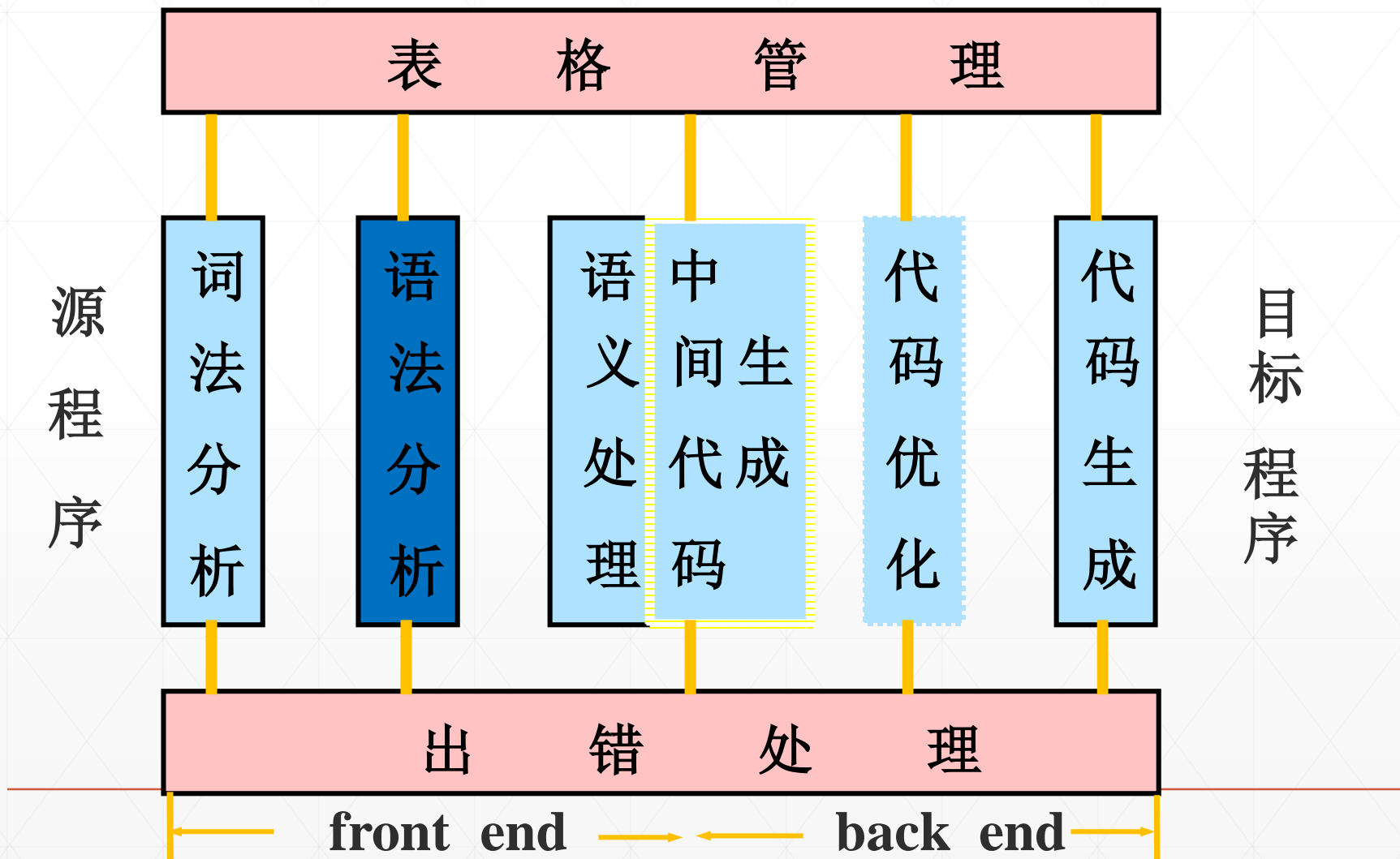


编译原理与设计

北京理工大学 计算机学院



语法分析：自下而上分析





自下而上分析：LR(0)分析

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

	a_1	a_2	...	a_n
...				
S_i	r_j	r_j	...	r_j
...				



自下而上分析：SLR(1)分析

按照SLR(1)方法构造的文法G的LR分析表，如果每个入口不含多重定义，则称它为G的SLR(1)分析表。具有SLR(1)分析表的文法G称为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

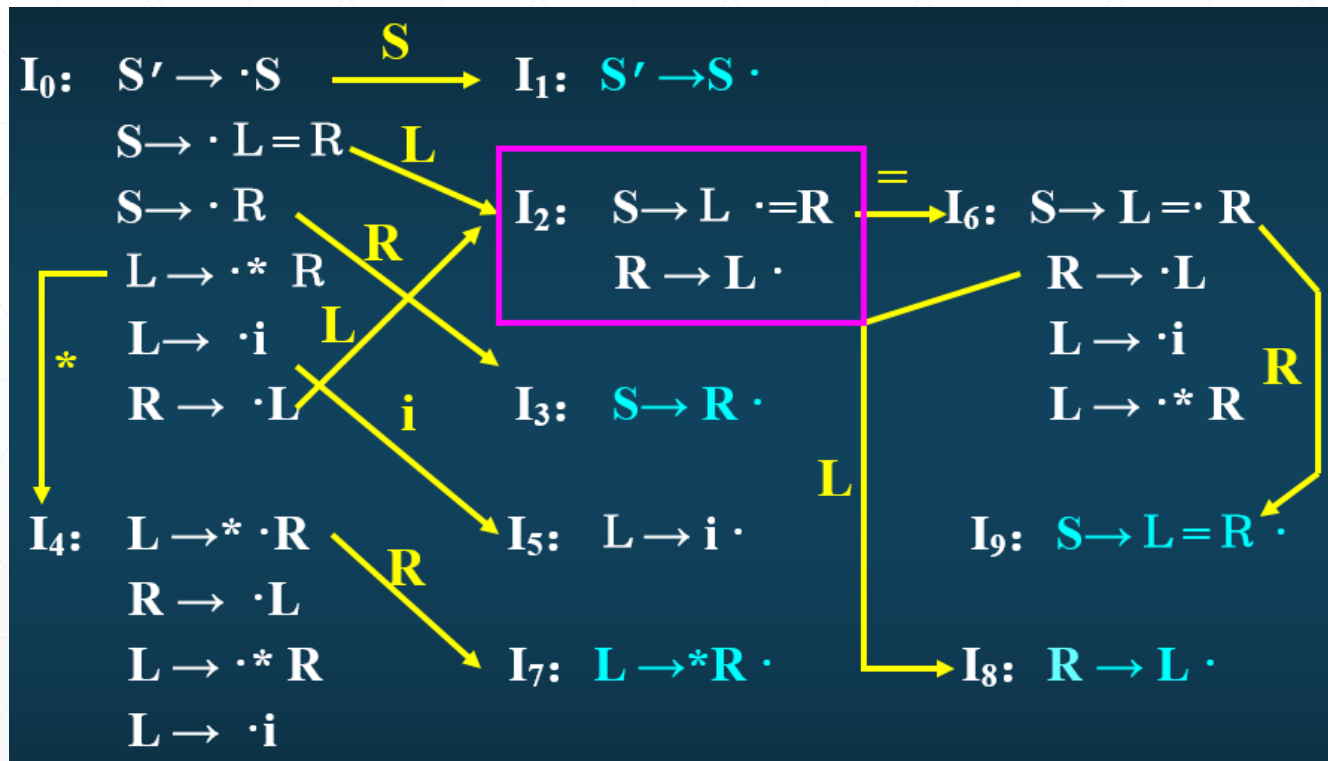
SLR(1)方法 = LR(0)方法 + “冲突”解决方法



自下而上分析：SLR(1)分析的缺陷

例：设文法 $G(S')$ 为

- (1) $S' \rightarrow S$
- (2) $S \rightarrow L=R$
- (3) $S \rightarrow R$
- (4) $L \rightarrow *R$
- (5) $L \rightarrow i$
- (6) $R \rightarrow L$





自下而上分析：SLR(1)分析的缺陷

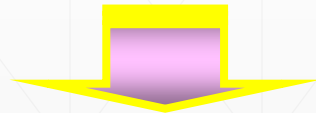
$I_2 = \{ S \rightarrow L = R, R \rightarrow L \cdot \}$ 存在移进 - 归约冲突
据SLR(1)方法有:

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

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

\therefore 文法G是非SLR(1)文法

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



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

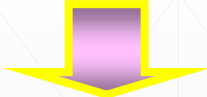


自下而上分析：SLR(1)分析的缺陷

若 $a \in \text{FOLLOW}(A) \Rightarrow \text{action}(k, a) = \{A \rightarrow \alpha\}$
归约未必有效！

\therefore 可能不存在一个规范句型含有 $\beta A a$
 \therefore 用 $A \rightarrow \alpha$ 归约不一定有效。

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



α	k
β	\dots
$\#$	0

对 a 的求解未限定是规范推导

$S: \dots a \dots \#$





自下而上分析：LR(1)分析方法

■ LR(k)项目

文法G的一个LR(k)项目是 $[A \rightarrow \alpha \cdot \beta, a_1 a_2 \dots a_k]$

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

$a_1 a_2 \dots a_k$ ：搜索字符串， $a_i \in V_T$ ；

■ 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

$$S \Rightarrow \delta A \omega \Rightarrow \delta \underline{\alpha} \beta \omega$$

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



自下而上分析：LR(1)分析方法

■ LR(0)项目

若文法G的一个LR(0)项目 $[A \rightarrow \beta_1 \cdot \beta_2]$ 对活前缀 $\alpha\beta_1$ 是有效的，当且仅当存在规范推导

$$S \Rightarrow \alpha A \omega \Rightarrow \alpha \beta_1 \beta_2 \omega$$

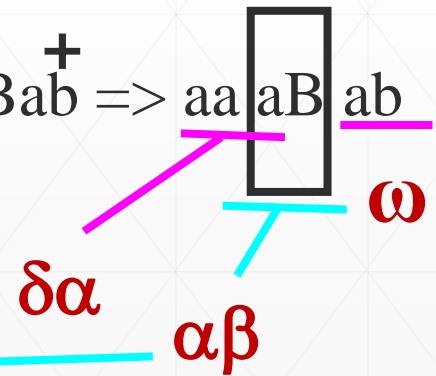
设有文法G: $S \rightarrow BB$ $B \rightarrow aB \mid b$

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

$\because \exists$ 规范推导 $S \Rightarrow BB \Rightarrow BaB \Rightarrow Bab \Rightarrow aBab \Rightarrow aa \boxed{aB} ab$

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

$\because \exists$ 规范推导 $S \Rightarrow BB \Rightarrow BaB \Rightarrow BaaB$

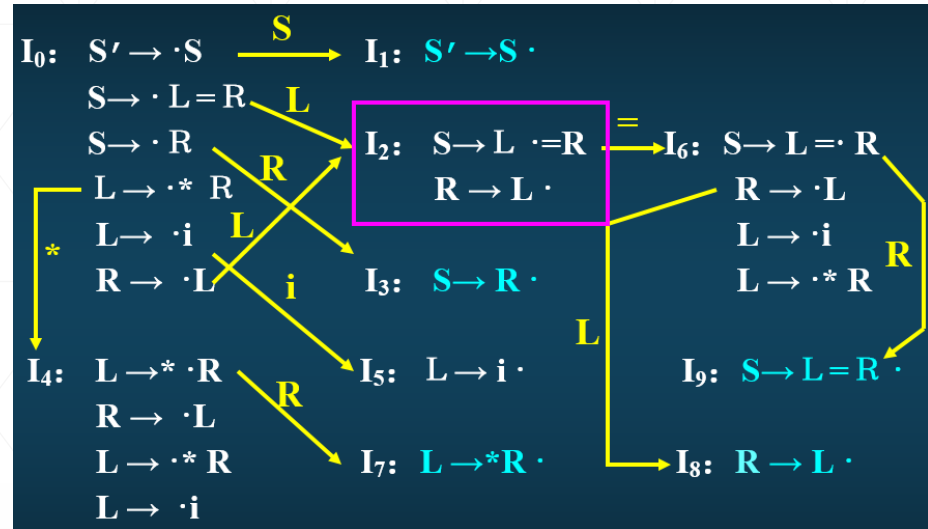




自下而上分析：LR(1)分析方法

例：设文法 $G(S')$ 为

- (1) $S' \rightarrow S$
- (2) $S \rightarrow L=R$
- (3) $S \rightarrow R$
- (4) $L \rightarrow *R$
- (5) $L \rightarrow i$
- (6) $R \rightarrow L$



$I_2 = \{ S \rightarrow L = R, R \rightarrow L \cdot \}$ FOLLOW(R) = { =, # }

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

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

$S' \Rightarrow S \Rightarrow L=R$ ，存在规范句型L是前缀，下一个符号是#，L不在栈顶

$S' \Rightarrow S \Rightarrow R \Rightarrow L$ ，存在规范句型L是前缀，下一个符号是#，L已经在栈顶



自下而上分析：LR(1)分析方法

- 构造LR (1) 项目集规范族C
 - step1 :求函数 $\text{closure}(I)$;
 - step2 :求GO函数 ;
 - step3 :用算法items构造LR(1)项目集规范族C。

```
closure ( I ) {  
    do {  
        if ( 对 I 的每个项目  $[A \rightarrow \alpha B\beta, a]$ ,  $G'$  中的每个产生式  $B \rightarrow \gamma$  和  $\text{FRIST}(\beta a)$  的每个终结符  $b$ , 如果  $[B \rightarrow \bullet \gamma]$  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 ;  
}
```



自下而上分析：LR(1)分析方法

- 构造LR(1)项目集规范族C
 - step1 :求函数 $\text{closure}(I)$;
 - step2 :求GO函数 ;
 - step3 :用算法items构造LR(1)项目集规范族C。

```
Go ( I, X )
```

```
{
```

```
    令J是项目  $[A \rightarrow \alpha X \bullet \beta, a]$  的集合, 使得  $[A \rightarrow \alpha \bullet X \beta, a]$  在 I 中 ;
```

```
    return  $\text{closure}(J)$ ;
```

```
}
```



自下而上分析：LR(1)分析方法

- 构造LR(1)项目集规范族C
 - step1 :求函数 $\text{closure}(I)$;
 - step2 :求GO函数 ;
 - step3 :用算法items构造LR(1)项目集规范族C。

```
items(G')
{
  C = closure ({S' →•S, #});
  do {
    if ( 对C的每个项目I和每个文法符号X, 若go(I, X) 非空且不在C中)
      把go(I, X) 加入C中;
  } while (没有更多的项目集可以加入C中);
}
```



自下而上分析：LR(1)分析方法

■ 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

设构造 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 为分析表状态，令含有 $S' \rightarrow \cdot S, \#$ 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 $X = a, (a \in V_T)$ 时，置 $action[I_i, a] = S_j$ 。若 $X \in V_N$ 时，则置 $GOTO[I_i, X] = j$ 。
 - ②若归约项目 $[A \rightarrow \alpha; a] \in I_i$ ， $A \rightarrow \alpha$ 为文法的第 j 个产生式，则置 $action[I_i, a] = r_j$ 。
 - ③若项目 $[S' \rightarrow S \cdot, \#] \in I_i$ ，则置： $action[I_i, \#] = \text{“acc”}$ 。
 - 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。
-



自下而上分析：LR(1)分析方法

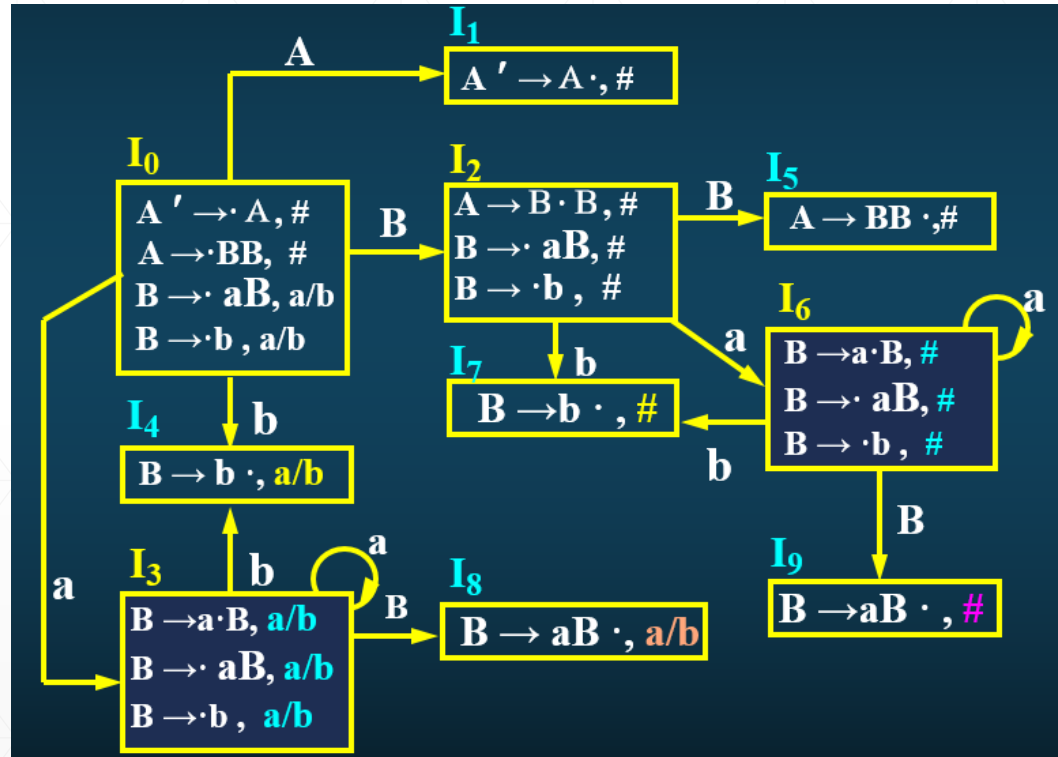
设有文法G(A)

(1) $A' \rightarrow A$

(2) $A \rightarrow BB$

(3) $B \rightarrow aB$

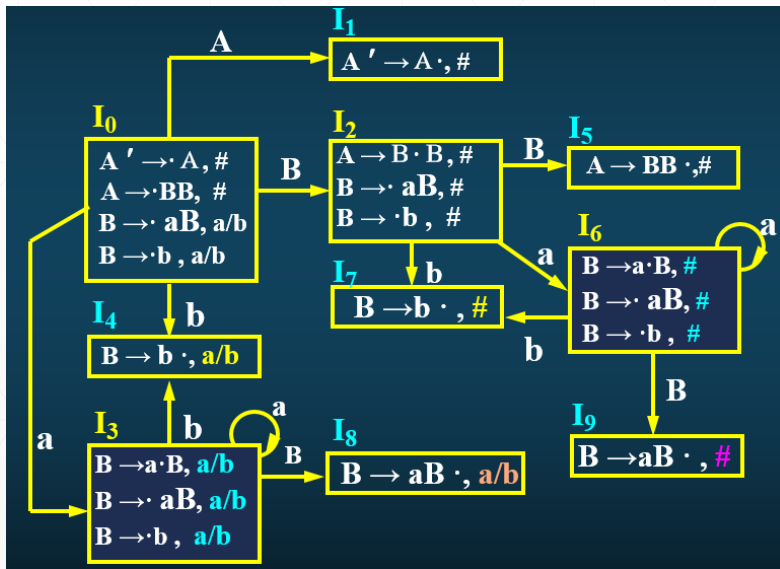
(4) $B \rightarrow b$





自下而上分析：LR(1)分析方法

- ① $A' \rightarrow A$
- ② $A \rightarrow BB$
- ③ $B \rightarrow aB$
- ④ $B \rightarrow b$



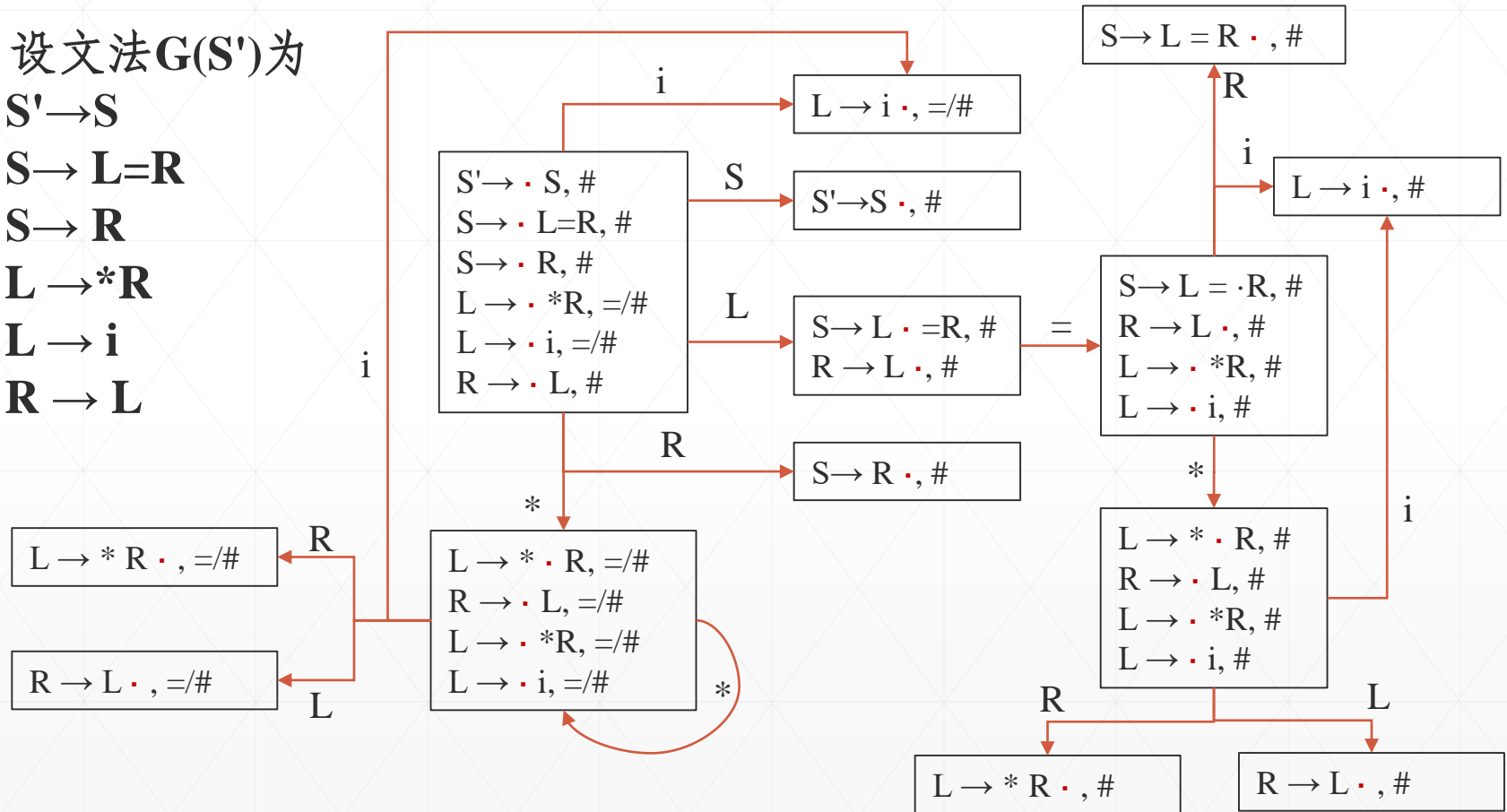
state	ACTION表			GOTO表	
	a	b	#	A	B
0	S ₃	S ₄		1	2
1			acc		
2	S ₆	S ₇			5
3	S ₃	S ₄			8
4	r ₄	r ₄			
5			r ₂		
6	S ₆	S ₇			9
7			r ₄		
8	r ₃	r ₃			
9			r ₃		



自下而上分析：LR(1)分析

例：设文法 $G(S')$ 为

- (1) $S' \rightarrow S$
- (2) $S \rightarrow L=R$
- (3) $S \rightarrow R$
- (4) $L \rightarrow *R$
- (5) $L \rightarrow i$
- (6) $R \rightarrow L$



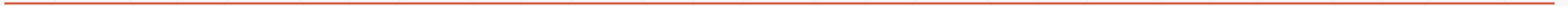


自下而上分析：LR(1)

$S \rightarrow aAd \mid bBd \mid aBe \mid bAe$

$A \rightarrow g$

$B \rightarrow g$





自下而上分析：LR(1)

$S \rightarrow aAd \mid bBd \mid aBe \mid bAe$

$A \rightarrow g$

$B \rightarrow g$

