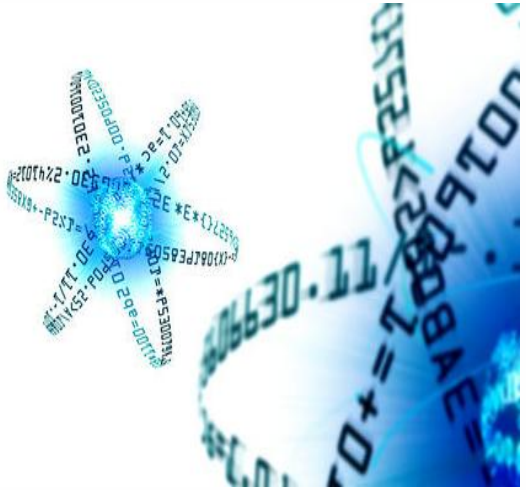
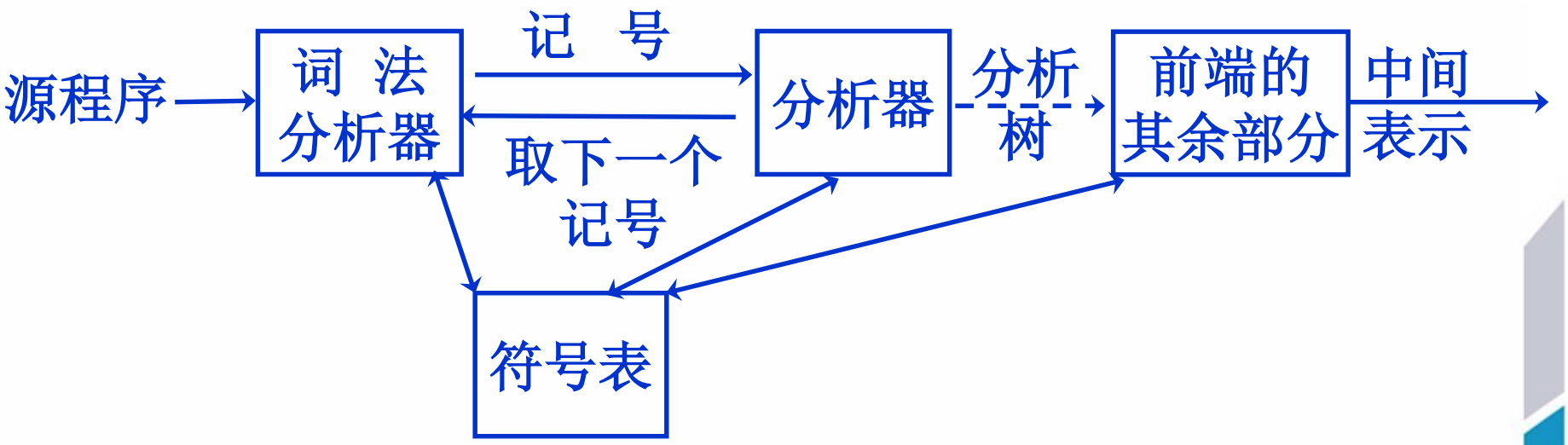
编译原理

第四章 语法分析



计算机科学与技术学院

第四章 语法分析



- 本章内容
  - 上下文无关文法
  - 自上而下分析和自下而上分析
  - 围绕分析器的自动生成展开

4.1 上下文无关文法

4.1.1 上下文无关文法的定义

- 正规式能定义一些简单的语言，能表示给定结构的固定次数的重复或者没有指定次数的重复

  例：$a\,(ba)^5$, $a\,(ba)^*$

- 正规式不能用于描述配对或嵌套的结构

  例1：配对括号串的集合

  例2：{*wcw* | *w*是*a*和*b*的串}

4.1 上下文无关文法

- 上下文无关文法是四元组（V_T, V_N, S, P）

  V_T：　终结符集合

  V_N：　非终结符集合

  S：　开始符号，非终结符中的一个

  P：　产生式集合，　产生式形式：$A \to \alpha$

- 例　({id, +, *, –, (,)}, {*expr*, *op*}, *expr*, *P*)

  expr → *expr op expr*　　*expr* → (*expr*)

  expr → – *expr*　　*expr* → id

  op → +　　*op* → *

4.1 上下文无关文法

- 简化表示

  $expr \rightarrow expr\ op\ expr \quad | (expr) | - expr | id$

  $op \rightarrow + | *$

- 简化表示

  $E \rightarrow E\ A\ E | (E) | -E | id$

  $A \rightarrow + | *$

4.1.2 推导

- 把产生式看成重写规则，把符号串中的非终结符用其产生式右部的串来代替

- 例　$E \to E + E \mid E * E \mid (E) \mid - E \mid \mathrm{id}$

  $E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E + E) \Rightarrow -(\mathrm{id} + E) \Rightarrow -(\mathrm{id} + \mathrm{id})$

- 概念
  - 上下文无关语言、等价的文法、句型

- 记号

  $S \Rightarrow^* \alpha$、 $S \Rightarrow^+ w$

4.1 上下文无关文法

- 例 $E \rightarrow E + E \mid E * E \mid (E) \mid - E \mid \mathrm{id}$

- 最左推导

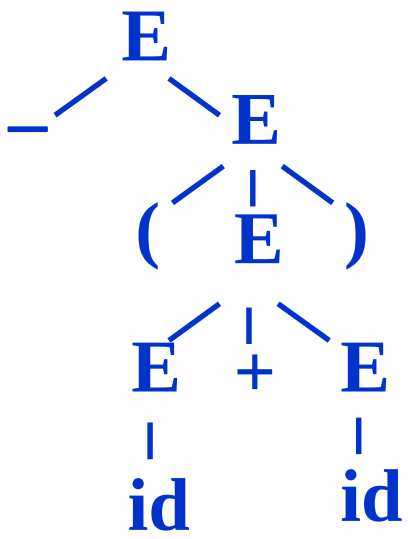
$$E \Rightarrow_{lm} -E \Rightarrow_{lm} -(E) \Rightarrow_{lm} -(E + E)$$
$$\Rightarrow_{lm} -(\mathrm{id} + E) \Rightarrow_{lm} -(\mathrm{id} + \mathrm{id})$$

- 最右推导（规范推导）

$$E \Rightarrow_{rm} -E \Rightarrow_{rm} -(E) \Rightarrow_{rm} -(E + E)$$
$$\Rightarrow_{rm} -(E + \mathrm{id}) \Rightarrow_{rm} -(\mathrm{id} + \mathrm{id})$$

4.1.3 分析树

- 例 $E \rightarrow E + E \mid E * E \mid (E) \mid - E \mid \mathrm{id}$



4.1.4 二义性

$E \Rightarrow E * E$
$\Rightarrow id * E$
$\Rightarrow id * E + E$
$\Rightarrow id * id + E$
$\Rightarrow id * id + id$

$E \Rightarrow E + E$
$\Rightarrow E * E + E$
$\Rightarrow id * E + E$
$\Rightarrow id * id + E$
$\Rightarrow id * id + id$

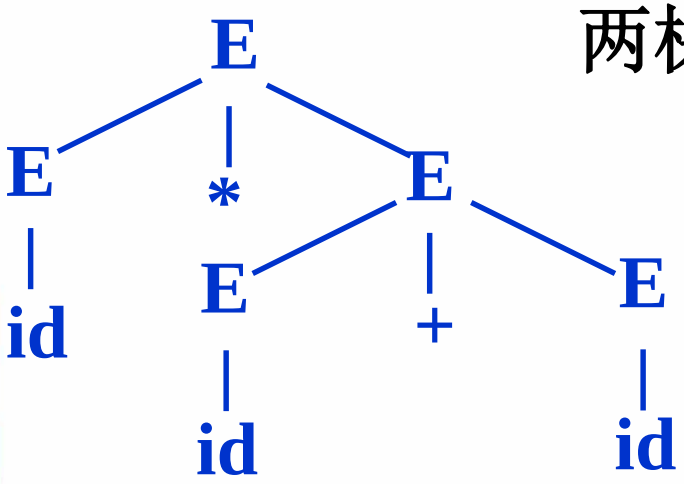
两个不同的最左推导

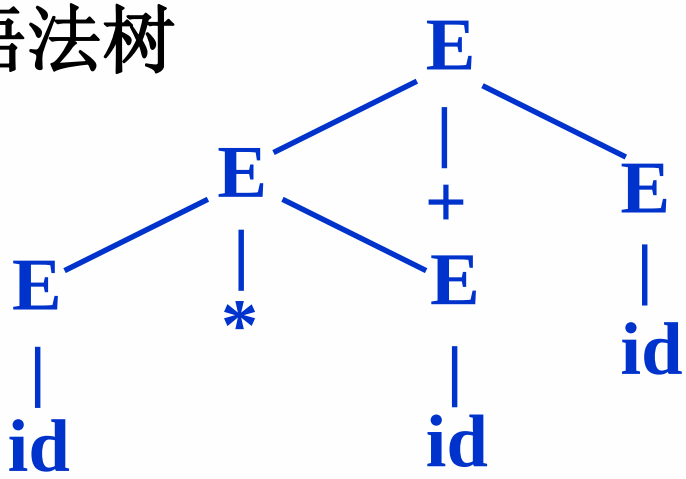
4.1.4 二义性

$E \Rightarrow E * E$
$\Rightarrow id * E$
$\Rightarrow id * E + E$
$\Rightarrow id * id + E$
$\Rightarrow id * id + id$

$E \Rightarrow E + E$
$\Rightarrow E * E + E$
$\Rightarrow id * E + E$
$\Rightarrow id * id + E$
$\Rightarrow id * id + id$

两棵不同的语法树





4.2语言和文法

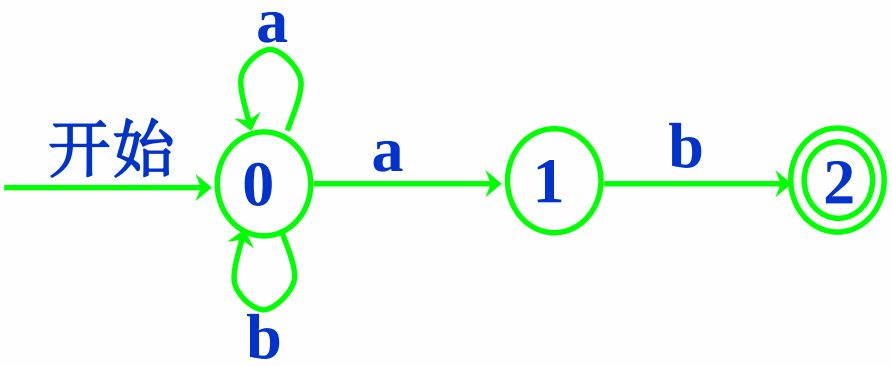
- 文法的优点
  - 文法给出了精确的，易于理解的语法说明
  - 自动产生高效的分析器
  - 可以给语言定义出层次结构
  - 以文法为基础的语言的实现便于语言的修改
- 文法的问题
  - 文法只能描述编程语言的大部分语法，不能描述语言中上下文有关的语法特征

4.2 语言和文法

4.2.1 正规式和上下文无关文法的比较

- 正规式

  $(a|b)^*ab$



- 文法

  $A_0 \rightarrow a\,A_0 \mid b\,A_0 \mid a\,A_1$

  $A_1 \rightarrow b\,A_2$

  $A_2 \rightarrow \varepsilon$

4.2.2 分离词法分析器理由

- 为什么要用正规式定义词法
  - 词法规则非常简单，不必用上下文无关文法
  - 对于词法记号，正规式描述简洁且易于理解
  - 从正规式构造出的词法分析器效率高

4.2 语言和文法

- 从软件工程角度看，词法分析和语法分析的分离有如下好处
  - 简化设计
  - 编译器的效率会改进
  - 编译器的可移植性加强
  - 便于编译器前端的模块划分

4.2 语言和文法

- 能否把词法分析并入到语法分析中，直接从字符流进行语法分析
  - 若把词法分析和语法分析合在一起，则必须将语言的注解和空白的规则反映在文法中，文法将大大复杂
  - 注解和空白由自己来处理的分析器，比注解和空格已由词法分析器删除的分析器要复杂得多

4.2.3 验证文法产生的语言

$G : S \rightarrow (S)\ S \mid \varepsilon$　$L(G)$ = 配对的括号串的集合

4.2.3 验证文法产生的语言

$G: S \rightarrow (S)\,S \mid \varepsilon$　$L(G)$ = 配对的括号串的集合

- 按推导步数进行归纳：推出的是配对括号串
  - 归纳基础：　$S \Rightarrow \varepsilon$
  - 归纳假设：少于n步的推导都产生配对的括号串
  - 归纳步骤：n步的最左推导如下：

$$S \Rightarrow (S)S \Rightarrow^* (x)\,S \Rightarrow^* (x)\,y$$

4.2.3 验证文法产生的语言

$G: S \rightarrow (S)\,S \mid \varepsilon$　$L(G)$ = 配对的括号串的集合

- 按串长进行归纳：配对括号串可由S推出
  - 归纳基础：　$S \Rightarrow \varepsilon$
  - 归纳假设：长度小于$2n$的都可以从S推导出来
  - 归纳步骤：考虑长度为$2n(n \geq 1)$的$w = (x)\,y$

$$S \Rightarrow (S)S \Rightarrow^* (x)\,S \Rightarrow^* (x)\,y$$

4.2.4 适当的表达式文法

- 用一种层次观点看待表达式

  id * id * (id+id) + id * id + id

4.2.4 适当的表达式文法

- 用一种层次观点看待表达式

  id * id * (id+id) + id * id + id

  id * id * (id+id)

- 文法

  expr → *expr* + *term* | *term*

  term → *term* * *factor* | *factor*

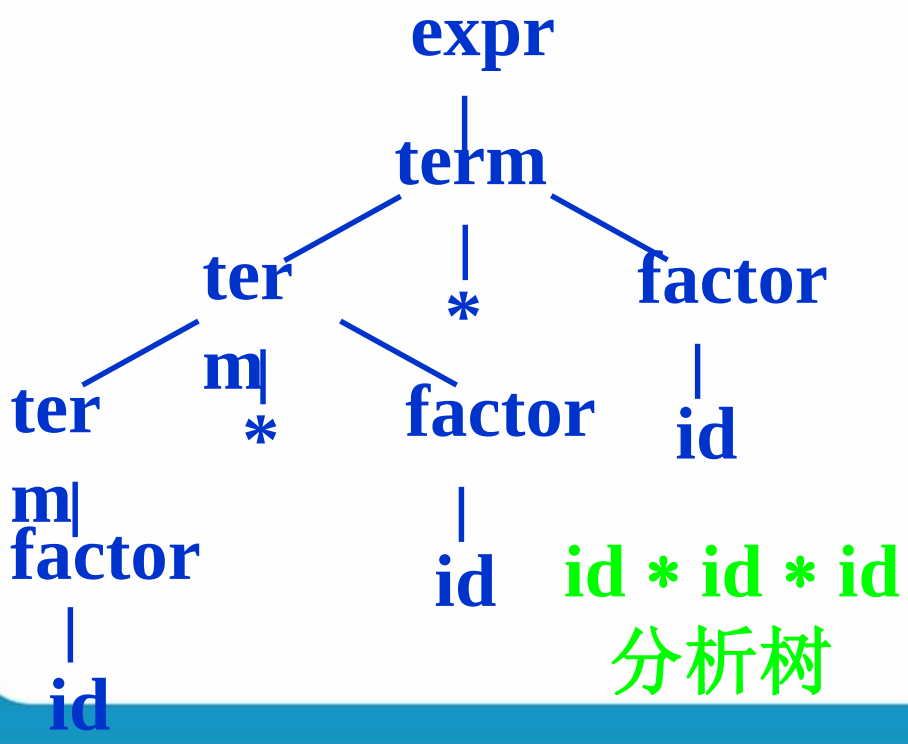
  factor → id | (*expr*)

4.2 语言和文法

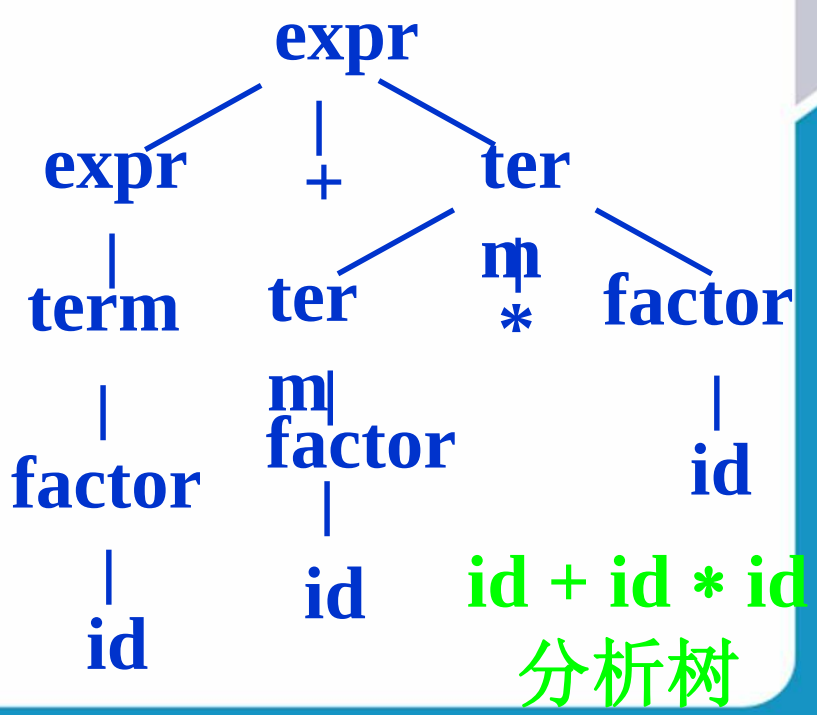
expr → *expr* + *term* | *term*

term → *term* * *factor* | *factor*

factor → id | (*expr*)



分析树



分析树

4.2.5 消除二义性

stmt → if *expr* then *stmt*
| if *expr* then *stmt* else *stmt*
| other

- 句型：if *expr* then if *expr* then *stmt* else *stmt*
- 两个最左推导：

stmt ⇒ if *expr* then *stmt*
⇒ if *expr* then if *expr* then *stmt* else *stmt*

stmt ⇒ if *expr* then *stmt* else *stmt*
⇒ if *expr* then if *expr* then *stmt* else *stmt*

4.2 语言和文法

- 无二义的文法

stmt → ***matched _stmt***
| ***unmatched_stmt***

matched_stmt → **if** ***expr*** **then** ***matched_stmt***
else ***matched_stmt***
| **other**

unmatched_stmt → **if** ***expr*** **then** ***stmt***
| **if** ***expr*** **then** ***matched_stmt***
else ***unmatched_stmt***

4.2.6 消除左递归

- 文法左递归 $A \Rightarrow^{+} A\alpha$
- 直接左递归 $A \rightarrow A\alpha \mid \beta$
  - 串的特点 $\beta\alpha \ldots \alpha$
- 消除直接左递归

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

4.2 语言和文法

- 例 算术表达文法

$E \to E + T \mid T$　　（$T + T \ldots + T$）

$T \to T * F \mid F$　　（$F * F \ldots * F$）

$F \to (E) \mid \mathrm{id}$

消除左递归后文法

$E \to TE'$

$E' \to + TE' \mid \varepsilon$

$T \to FT'$

$T' \to * F T' \mid \varepsilon$

$F \to (E) \mid \mathrm{id}$

4.2 语言和文法

- 非直接左递归

  $S \rightarrow Aa \mid b$

  $A \rightarrow Sd \mid \varepsilon$

- 先变换成直接左递归

  $S \rightarrow Aa \mid b$

  $A \rightarrow Aad \mid bd \mid \varepsilon$

- 再消除左递归

  $S \rightarrow Aa \mid b$

  $A \rightarrow bd\,A' \mid A'$

  $A' \rightarrow adA' \mid \varepsilon$

4.2.7 提左因子

- 有左因子的文法

$$A \to \alpha\beta_1 \mid \alpha\beta_2$$

- 提左因子

$$A \to \alpha A'$$

$$A' \to \beta_1 \mid \beta_2$$

4.2 语言和文法

- 例 悬空*else*的文法

  stmt → **if *expr* then *stmt* else *stmt***
  | if *expr* then *stmt*
  | other

  提左因子

  stmt → **if *expr* then *stmt optional_else_part***
  | other

  optional_else_part → **else *stmt***
  | ε

4.2.8 非上下文无关的语言构造

- $L_1 = \{wcw \mid w$属于$(a \mid b)^*\}$
  - 标识符的声明应先于其引用的抽象

- $L_2 = \{a^n b^m c^n d^m \mid n \geq 0, m \geq 0\}$
  - 形参个数和实参个数应该相同的抽象

- $L_3 = \{a^n b^n c^n \mid n \geq 0\}$
  - 早先排版描述的一个现象的抽象
    b e g i n：5个字母键，5个回退键，5个下划线键

4.2 语言和文法

- $L_1' = \{wcw^R \mid w \in (a|b)^*\}$

  $S \rightarrow aSa \mid bSb \mid c$

- $L_2' = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

  $S \rightarrow aSd \mid aAd$

  $A \rightarrow bAc \mid bc$

- $L_2'' = \{a^n b^n c^m d^m \mid n \geq 1,\ m \geq 1\}$

  $S \rightarrow AB$

  $A \rightarrow aAb \mid ab$

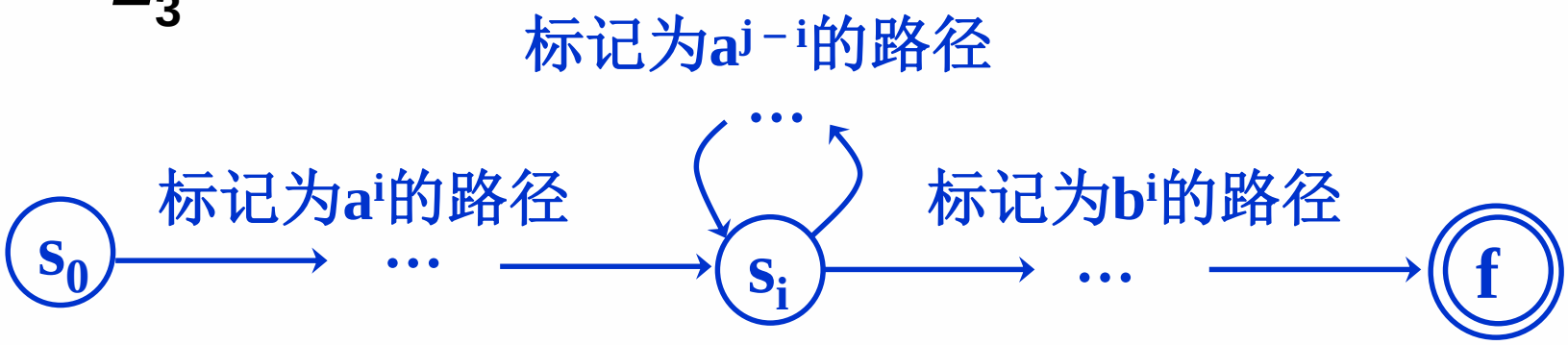
  $B \rightarrow cBd \mid cd$

4.2 语言和文法

- $L_3' = \{a^n b^n \mid n \geq 1\}$

  $S \rightarrow aSb \mid ab$

- L_3'是不能用正规式描述的语言的一个范例
  - 若存在接受L_3' 的DFA D，状态数为k个
  - 设D读完$\varepsilon, a, aa, \ldots, a^k$ 分别到达状态$s_0, s_1, \ldots, s_k$
  - 至少有两个状态相同，例如是s_i和s_j，则$a^j b^i$属于L_3'



4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \rightarrow \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$

4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$
- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 短语文法

4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 1型文法：$|\alpha| \leq |\beta|$，但$S \to \varepsilon$可以例外

- 短语文法

4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 1型文法：$|\alpha| \leq |\beta|$，但$S \to \varepsilon$可以例外

- 短语文法、上下文有关文法

4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 1型文法：$|\alpha| \leq |\beta|$，但$S \to \varepsilon$可以例外
- 2型文法：$A \to \beta$，$A \in V_N$，$\beta \in (V_N \cup V_T)^*$

- 短语文法、上下文有关文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 1型文法：$|\alpha| \leq |\beta|$，但$S \to \varepsilon$可以例外
- 2型文法：$A \to \beta$，$A \in V_N$，$\beta \in (V_N \cup V_T)^*$

- 短语文法、上下文有关文法、上下文无关文法

4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 1型文法：$|\alpha| \leq |\beta|$，但$S \to \varepsilon$可以例外
- 2型文法：$A \to \beta$，$A \in V_N$，$\beta \in (V_N \cup V_T)^*$
- 3型文法：$A \to aB$或$A \to a$，$A, B \in V_N$，$a \in V_T$

- 短语文法、上下文有关文法、上下文无关文法

4.2 语言和文法

4.2.9 形式语言鸟瞰

- 文法 $G = (V_T, V_N, S, P)$

- 0型文法：$\alpha \to \beta$，$\alpha, \beta \in (V_N \cup V_T)^*$，$|\alpha| \geq 1$
- 1型文法：$|\alpha| \leq |\beta|$，但$S \to \varepsilon$可以例外
- 2型文法：$A \to \beta$，$A \in V_N$，$\beta \in (V_N \cup V_T)^*$
- 3型文法：$A \to aB$或$A \to a$，$A, B \in V_N$，$a \in V_T$

- 短语文法、上下文有关文法、上下文无关文法、正规文法

4.2 语言和文法

- 例 $L_3=\{a^nb^nc^n \mid n \geq 1\}$的上下文有关文法

$S \to aSBC$　　$S \to aBC$　　$CB \to BC$

$aB \to ab$　　$bB \to bb$　　$bC \to bc$

$cC \to cc$

$a^nb^nc^n$的推导过程如下：

$S \Rightarrow^* a^{n-1}S(BC)^{n-1}$　　用$S \to aSBC$　n-1次

$S \Rightarrow^+ a^n(BC)^n$　　用$S \to aBC$　1次

$S \Rightarrow^+ a^nB^nC^n$　　用$CB \to BC$交换相邻的CB

$S \Rightarrow^+ a^nbB^{n-1}C^n$　　用$aB \to ab$　1次

$S \Rightarrow^+ a^nb^nC^n$　　用$bB \to bb$　n-1次

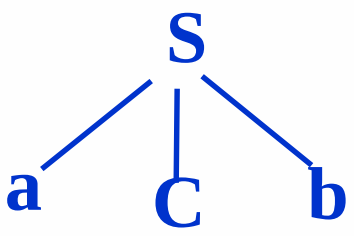
$S \Rightarrow^+ a^nb^ncC^{n-1}$　　用$bC \to bc$　1次

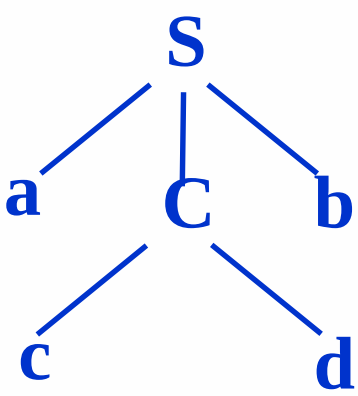
$S \Rightarrow^+ a^nb^nc^n$　　用$cC \to cc$　n-1次

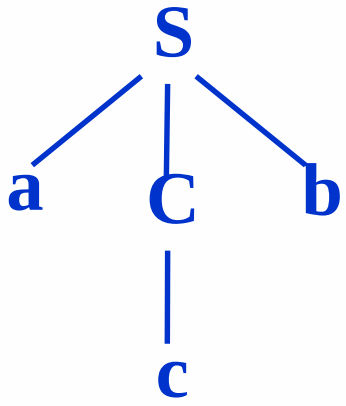
4.3 自上而下分析

4.3.1 自上而下分析的一般方法

- 例 文法 $S \rightarrow aCb$ $C \rightarrow cd \mid c$
  为输入串$w = acb$建立分析树







不能处理左递归

4.3 自上而下分析

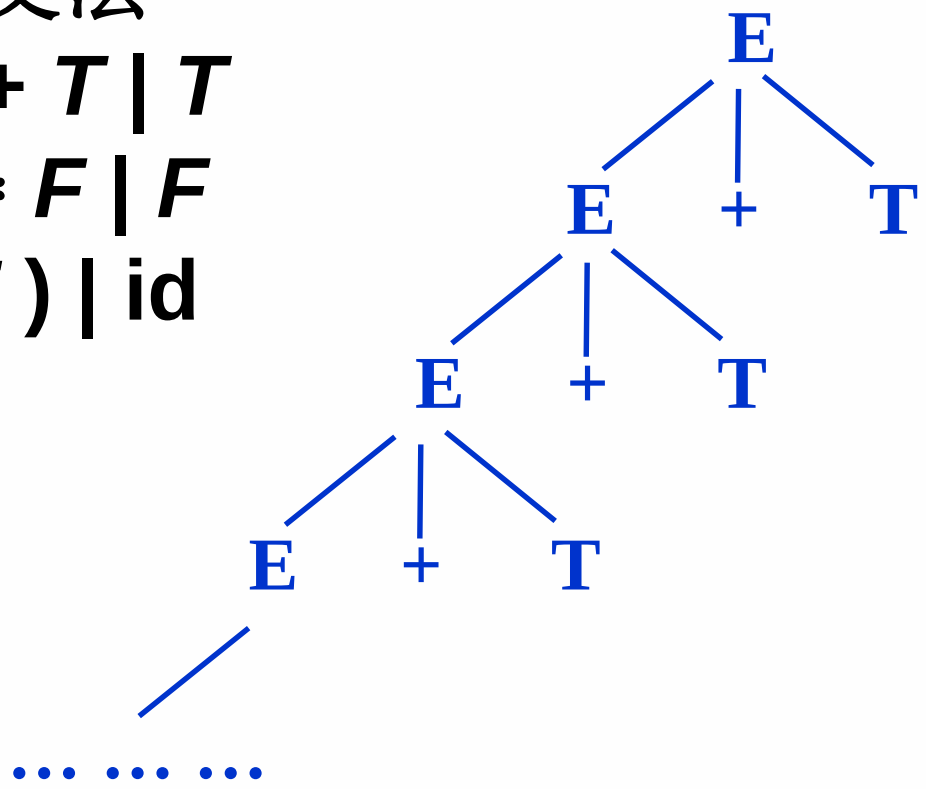
- 不能处理左递归的例子

  算术表达文法

  $E \rightarrow E + T \mid T$

  $T \rightarrow T * F \mid F$

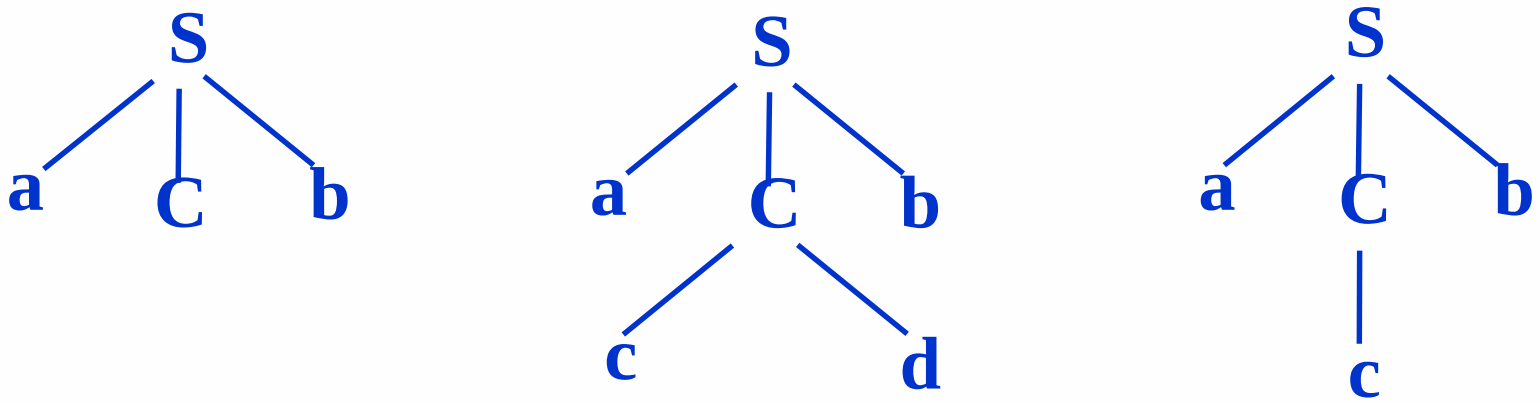
  $F \rightarrow (E) \mid \mathrm{id}$



4.3 自上而下分析

4.3.1 自上而下分析的一般方法

- 例 文法 $S \rightarrow aCb$ $C \rightarrow cd \mid c$
  为输入串$w = acb$建立分析树

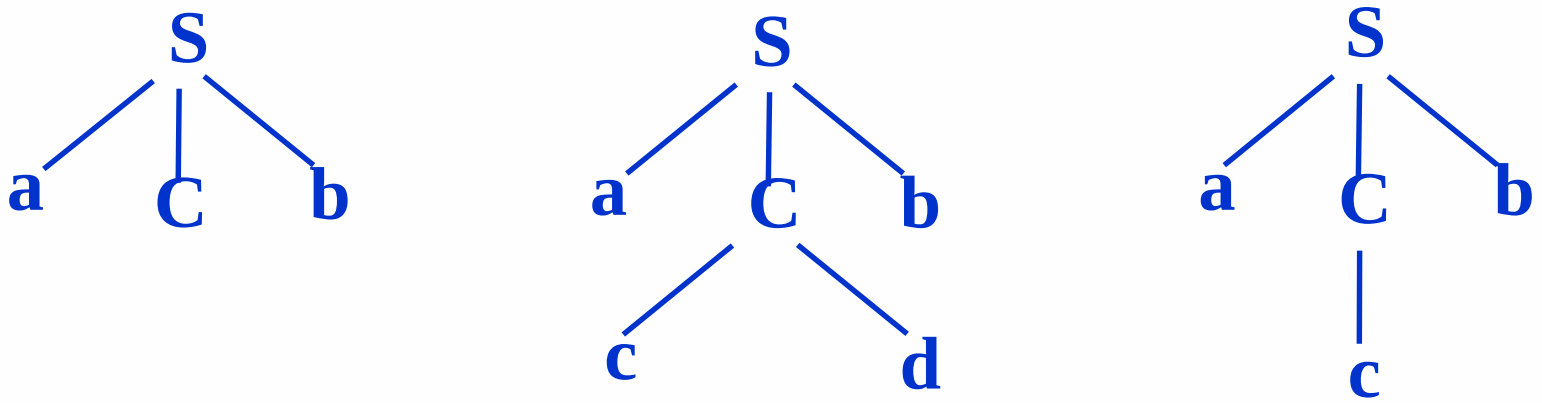


不能处理左递归、复杂的回溯技术

4.3 自上而下分析

4.3.1 自上而下分析的一般方法

- 例 文法 $S \rightarrow aCb$ $C \rightarrow cd \mid c$
  为输入串$w = acb$建立分析树

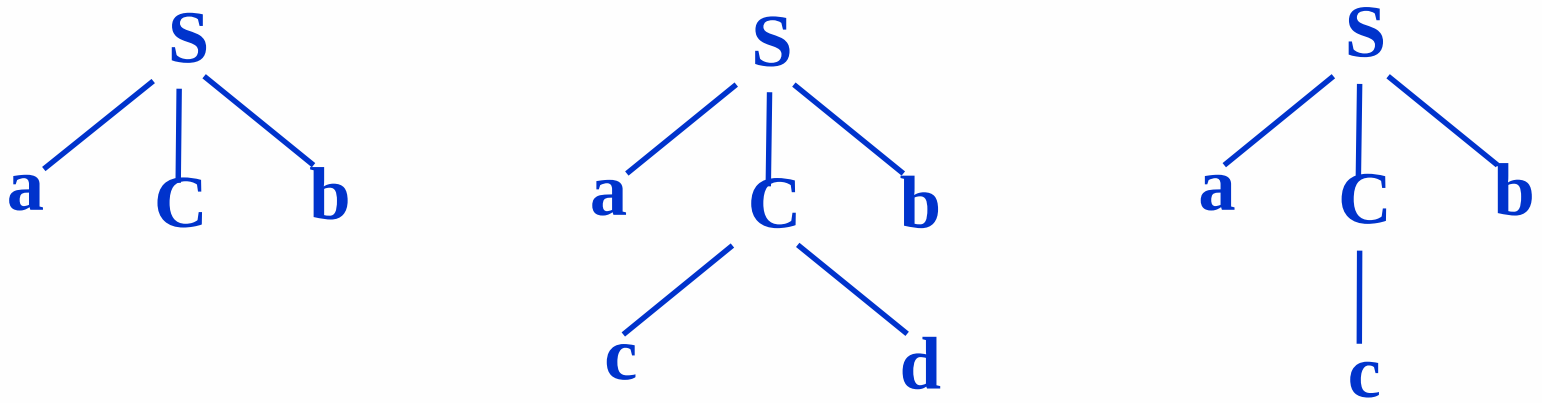


不能处理左递归、复杂的回溯技术、回溯导致语义工作推倒重来

4.3 自上而下分析

4.3.1 自上而下分析的一般方法

- 例 文法 $S \rightarrow aCb$ $C \rightarrow cd \mid c$
  为输入串$w = acb$建立分析树

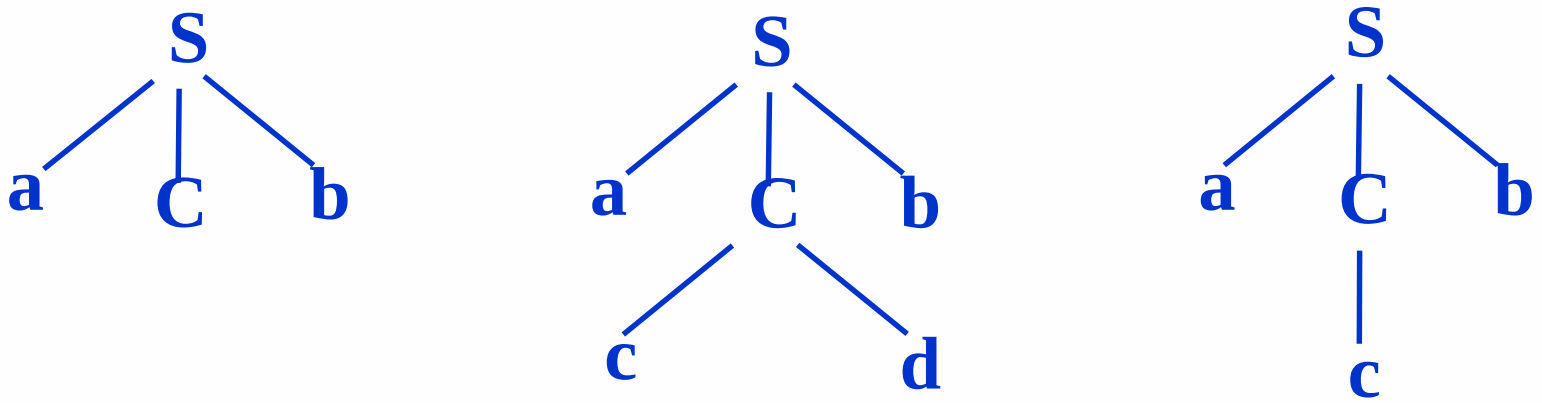


不能处理左递归、复杂的回溯技术、回溯导致语义工作推倒重来、难以报告出错的确切位置

4.3 自上而下分析

4.3.1 自上而下分析的一般方法

- 例 文法 $S \rightarrow aCb$ $C \rightarrow cd \mid c$
  为输入串$w = acb$建立分析树



不能处理左递归、复杂的回溯技术、回溯导致语义工作推倒重来、难以报告出错的确切位置、效率低

4.3 自上而下分析

4.3.2 LL(1)文法

- 对文法加什么样的限制可以保证没有回溯？

- 先定义两个和文法有关的函数
  - **FIRST(α) = {a | $\alpha \Rightarrow^* a\ldots$, $a \in V_T$}**
    特别是，$\alpha \Rightarrow^* \varepsilon$时，规定$\varepsilon \in$ **FIRST(α)**

对A的任何两个不同选择α_i 和α_j，希望有

$$\text{FIRST}(\alpha_i) \cap \text{FIRST}(\alpha_j) = \varnothing$$

若**FIRST(α_i)** 或 **FIRST(α_j)**含ε，还需增加条件

4.3.2 LL(1)文法

- 对文法加什么样的限制可以保证没有回溯？

- 先定义两个和文法有关的函数
  - FIRST(α) = {a | $\alpha \Rightarrow^* a\ldots$, $a \in V_T$}
    特别是，$\alpha \Rightarrow^* \varepsilon$时，规定$\varepsilon \in$ FIRST(α)
  - FOLLOW(A) = {a | $S \Rightarrow^* \ldots Aa\ldots$，$a \in V_T$}
    如果A是某个句型的最右符号，那么\$属于FOLLOW(A)

4.3 自上而下分析

- **LL(1)文法**

  任何两个产生式$A \to \alpha \mid \beta$都满足下列条件：

  - $FIRST(\alpha) \cap FIRST(\beta) = \varnothing$
  - 若$\beta \Rightarrow^* \varepsilon$，那么$FIRST(\alpha) \cap FOLLOW(A) = \varnothing$

  例如, 对于下面文法，面临a…时，第2步推导不知用哪个产生式

  $S \to A\ B$

  $A \to a\ b \mid \varepsilon$　　$a \in FIRST(ab) \cap FOLLOW(A)$

  $B \to a\ C$

  $C \to \ldots$

4.3 自上而下分析

- **LL(1)文法**
  任何两个产生式$A \to \alpha \mid \beta$都满足下列条件：
  - $\mathbf{FIRST}(\alpha) \cap \mathbf{FIRST}(\beta) = \varnothing$
  - 若$\beta \Rightarrow^* \varepsilon$，那么$\mathbf{FIRST}(\alpha) \cap \mathbf{FOLLOW}(A) = \varnothing$

- **LL(1)文法有一些明显的性质**
  - 没有公共左因子
  - 不是二义的
  - 不含左递归

4.3 自上而下分析

- 例 $E \rightarrow TE'$

  $E' \rightarrow +TE' \mid \varepsilon$

  $T \rightarrow FT'$

  $T' \rightarrow *FT' \mid \varepsilon$

  $F \rightarrow (E) \mid id$

FIRST(*E*) = FIRST(*T*) = FIRST(*F*) = { (, id }

FIRST(*E* ′) = {+, ε}

FRIST(*T* ′) = {*, ε}

FOLLOW(*E*) = FOLLOW(*E* ′) = {), $}

FOLLOW(*T*) = FOLLOW (*T* ′) = {+,), $}

FOLLOW(*F*) = {+, *,), $}

4.3.3 递归下降的预测分析

- 为每一个非终结符写一个分析过程
- 这些过程可能是递归的

❖ 例

type → ***simple***
| ↑ id
| array [***simple***] of ***type***

simple → integer
| char
| num dotdot num

4.3 自上而下分析

一个辅助过程

```
void match (terminal t) {
        if (lookahead == t) lookahead = nextToken( );
        else error( );
}
```

4.3 自上而下分析

```
void type( ) {
   if ( (lookahead == integer) || (lookahead == char) ||
                                   (lookahead == num) )
        simple( );
   else if ( lookahead == '↑' ) { match('↑'); match(id);}
   else if (lookahead == array) {
        match(array); match( '[' ); simple( );
        match( ']' ); match(of ); type( );
   }
   else error( );
}
```

type → simple
| ↑ id
| array [simple] of type

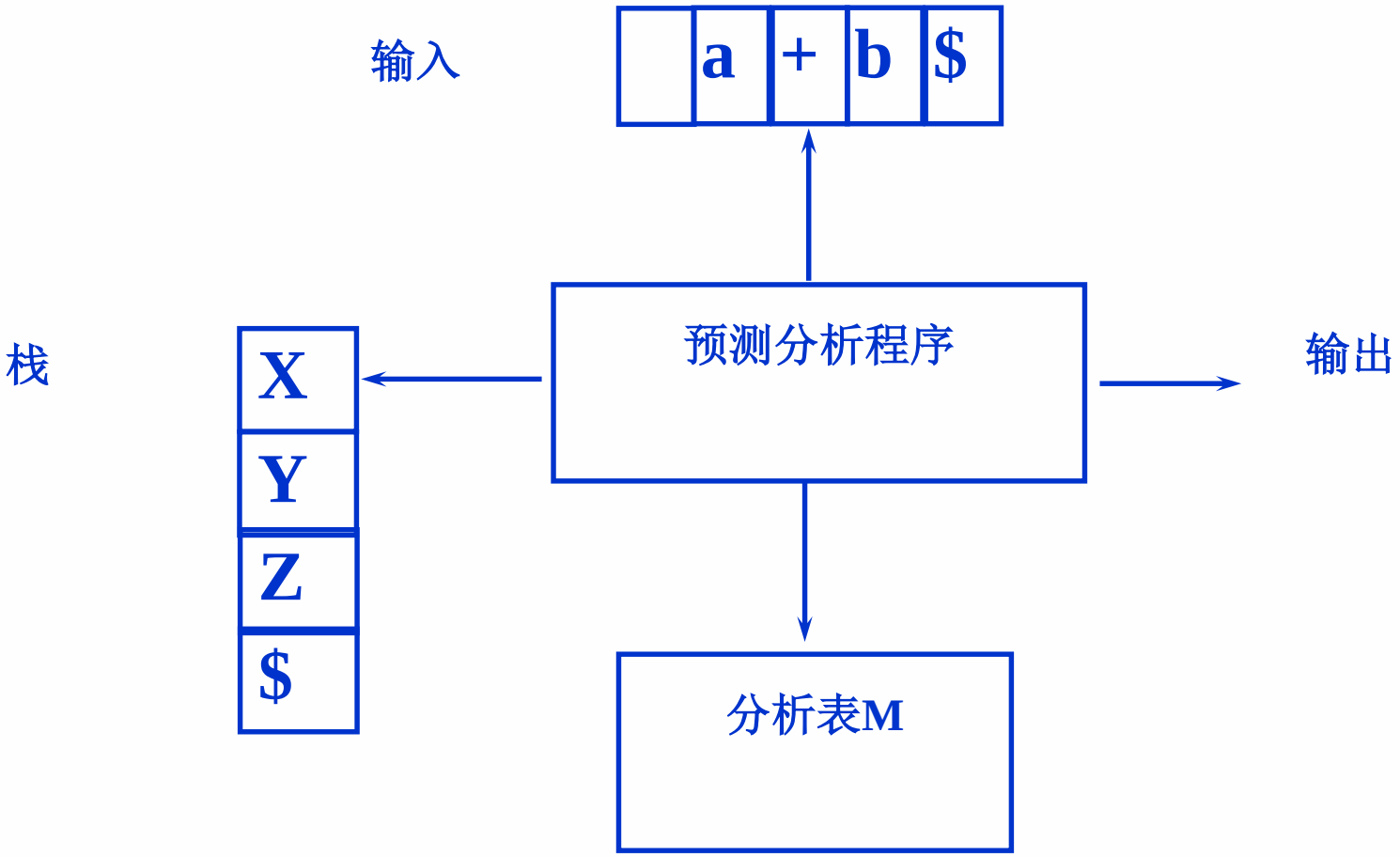
4.3 自上而下分析

```
void simple( ) {
   if ( lookahead == integer) match(integer);
   else if (lookahead == char) match(char);
   else if (lookahead == num) {
        match(num); match(dotdot); match(num);
   }
   else error( );
}
```

simple → integer
| char
| num dotdot num

4.3 自上而下分析

4.4.4 非递归的预测分析



4.3 自上而下分析

分析表的一部分

非终结符	输入符号			
	id	+	*	…
E	$E \to TE'$			
E'		$E' \to +TE'$		
T	$T \to FT'$			
T'		$T' \to \varepsilon$	$T' \to *FT'$	
F	$F \to \mathrm{id}$			

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$*E*	id * id + id$	

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E 'T	id * id + id$	$E \rightarrow TE'$

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E ′T	id * id + id$	$E \rightarrow TE'$
$E ′T ′F	id * id + id$	$T \rightarrow FT'$

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E ′T	id * id + id$	$E \rightarrow TE'$
$E ′T ′F	id * id + id$	$T \rightarrow FT'$
$E ′T ′ id	id * id + id$	$F \rightarrow$ id

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E 'T	id * id + id$	$E \rightarrow TE'$
$E 'T 'F	id * id + id$	$T \rightarrow FT'$
$E 'T ' id	id * id + id$	$F \rightarrow$ id
$E 'T '	* id + id$	

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E ′T	id * id + id$	$E \rightarrow TE'$
$E ′T ′F	id * id + id$	$T \rightarrow FT'$
$E ′T ′ id	id * id + id$	$F \rightarrow$ id
$E ′T ′	* id + id$	
$E ′T ′F *	* id + id$	$T' \rightarrow *FT'$

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E ′T	id * id + id$	$E \rightarrow TE'$
$E ′T ′F	id * id + id$	$T \rightarrow FT'$
$E ′T ′ id	id * id + id$	$F \rightarrow$ id
$E ′T ′	* id + id$	
$E ′T ′F *	* id + id$	$T' \rightarrow *FT'$
$E ′T ′F	id + id$	

4.3 自上而下分析

预测分析器接受输入id * id + id的前一部分动作

栈	输　入	输　出
$E	id * id + id$	
$E ′T	id * id + id$	$E \rightarrow TE'$
$E ′T ′F	id * id + id$	$T \rightarrow FT'$
$E ′T ′ id	id * id + id$	$F \rightarrow$ id
$E ′T ′	* id + id$	
$E ′T ′F *	* id + id$	$T' \rightarrow *FT'$
$E ′T ′F	id + id$	
$E ′T ′ id	id + id$	$F \rightarrow$ id

4.4.5 构造预测分析表

（1）对文法的每个产生式$A \to \alpha$，执行(2)和(3)

（2）对FIRST(α)的每个终结符a，

把$A \to \alpha$加入$M[A, a]$

（3）如果ε在FIRST(α)中，对FOLLOW(A)的每个终结符b（包括$），把$A \to \alpha$加入$M[A, b]$

（4）M中其它没有定义的条目都是error

4.3 自上而下分析

多重定义的条目

非终结符	输入符号			
	other	*b*	else	. . .
stmt	*stmt* → other			
e_part			*e_part*→ else *stmt* *e_part* → ε	
expr		*expr* → *b*		

4.3 自上而下分析

消去多重定义

非终结符	输入符号			
	other	*b*	else	. . .
stmt	*stmt* → other			
e_part			*e_part*→ else *stmt*	
expr		*expr* → *b*		

4.4.6 预测分析的错误恢复

1、先对编译器的错误处理做一个概述

- 词法错误，如标识符、关键字或算符的拼写错
- 语法错误，如算术表达式的括号不配对
- 语义错误，如算符作用于不相容的运算对象
- 逻辑错误，如无穷的递归调用

4.3 自上而下分析

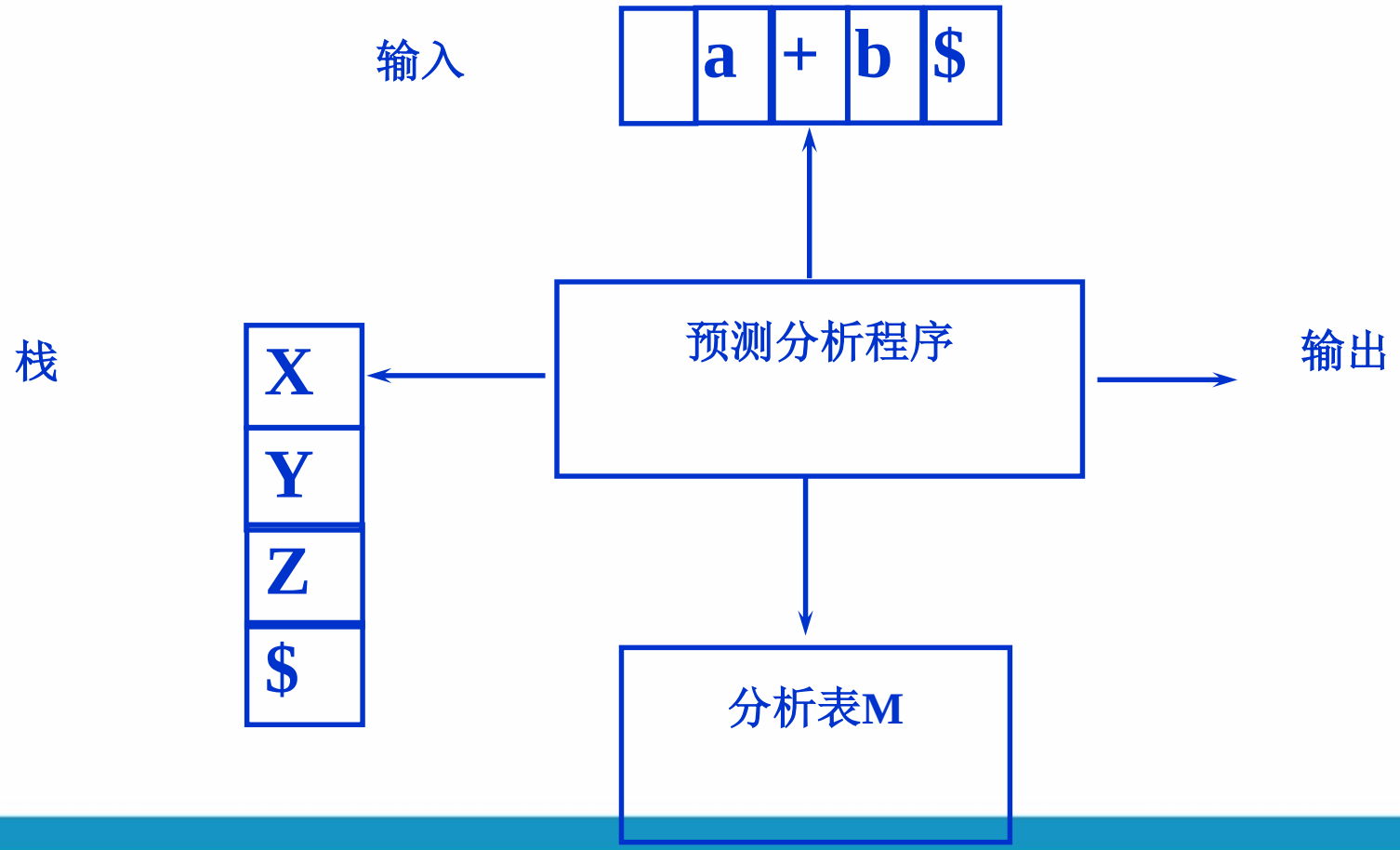
2、分析器对错误处理的基本目标

- 清楚而准确地报告错误的出现，并尽量少出现伪错误
- 迅速地从每个错误中恢复过来，以便诊断后面的错误
- 它不应该使正确程序的处理速度降低太多

4.3 自上而下分析

3、非递归预测分析在什么场合下发现错误

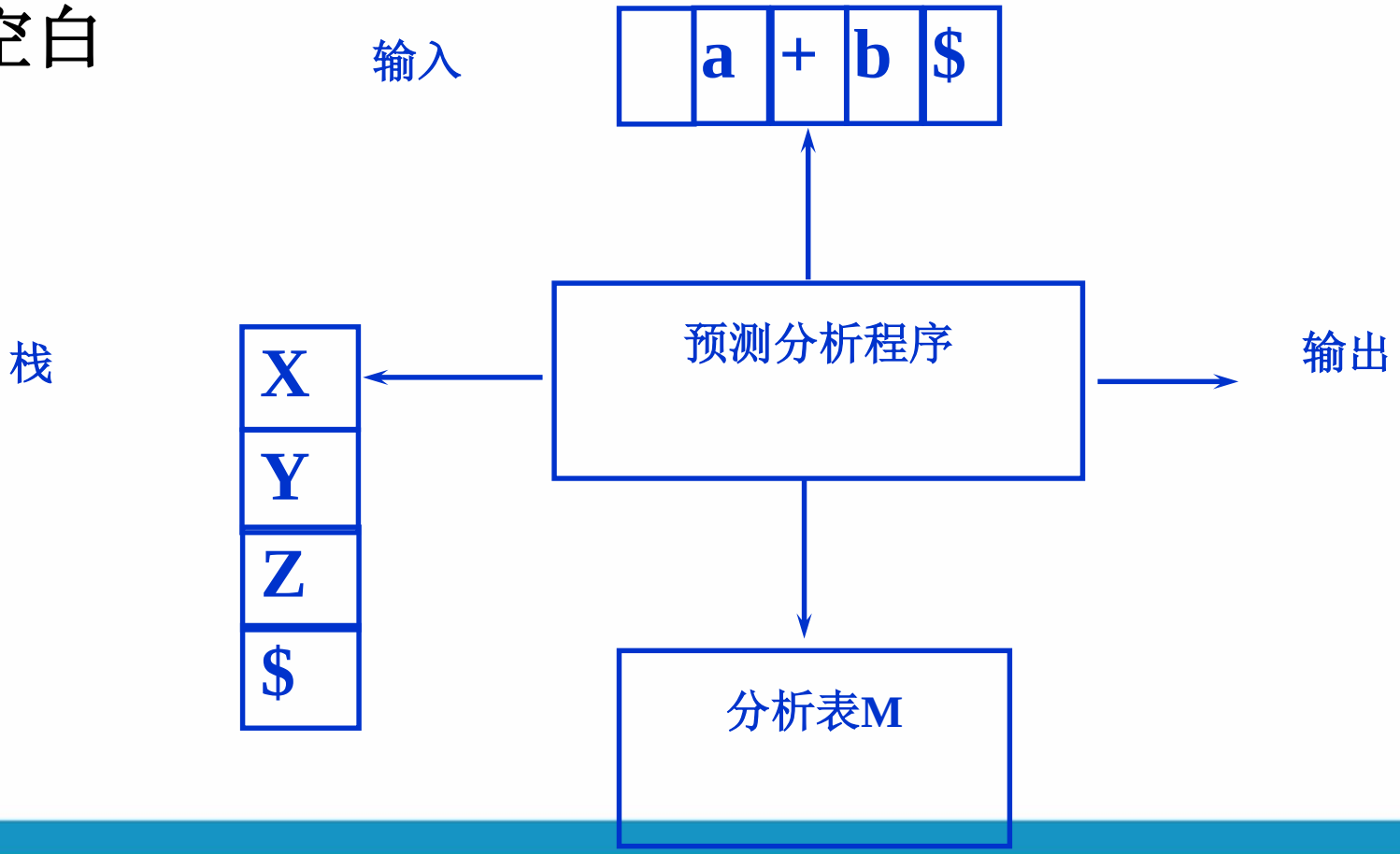
- 栈顶的终结符和下一个输入符号不匹配



4.3 自上而下分析

3、非递归预测分析在什么场合下发现错误

- 栈顶是非终结符*A*，输入符号是*a*，而*M*[*A* , *a*]是空白



4.3 自上而下分析

4、非递归预测分析：采用紧急方式的错误恢复

- 发现错误时，分析器每次抛弃一个输入记号，直到输入记号属于某个指定的同步记号集合为止

5、同步

- 词法分析器当前提供的记号流能够构成的语法构造，正是语法分析器所期望的
- 不同步的例子

  语法分析器期望剩余的前缀构成过程调用语句，而实际剩余的前缀形成赋值语句

6、同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合

6、同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合

if *expr* then *stmt*

（then和分号等记号是*expr*的同步记号）

6、同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合
- 把高层构造的开始符号加到低层构造的同步记号集合中

6、同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合
- 把高层构造的开始符号加到低层构造的同步记号集合中

***a* = *expr*; if …**

(语句的开始符号作为表达式的同步记号，以免表达式出错又遗漏分号时忽略if语句等一大段程序)

4.3 自上而下分析

6、同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合
- 把高层构造的开始符号加到低层构造的同步记号集合中
- 把FIRST(*A*)的终结符加入*A*的同步记号集合

***a* = *expr*; ， if …**

（语句的开始符号作为语句的同步符号，以免多出一个逗号时会把if语句忽略了）

4.3 自上而下分析

6、同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合
- 把高层构造的开始符号加到低层构造的同步记号集合中
- 把FIRST(*A*)的终结符加入*A*的同步记号集合
- 如果非终结符可以产生空串，若出错时栈顶是这样的非终结符，则可以使用推出空串的产生式

4.3 自上而下分析

例　　栈顶为T′，面临id时出错

非终结符	输入符号			
	id	+	*	…
E	$E\to TE'$			
E'		$E'\to +TE'$		
T	$T\to FT'$			
T'	出错	$T'\to \varepsilon$	$T'\to *FT'$	
…				

4.3 自上而下分析

T′可以产生空串，报错并用T′→ε

非终结符	输入符号			
	id	+	*	…
E	*E*→*TE*′			
E′		*E*′→+*TE*′		
T	*T*→*FT*′			
T′	出错，用*T*′→ε	*T*′→ε	*T*′→**FT*′	
…				

4.3 自上而下分析

同步记号集合的选择

- 把FOLLOW(*A*)的所有终结符放入非终结符*A*的同步记号集合
- 把高层结构的开始符号加到低层结构的同步记号集合中
- 把FIRST(*A*)的终结符加入*A*的同步记号集合
- 如果非终结符可以产生空串，若出错时栈顶是这样的非终结符，则可以使用推出空串的产生式
- 如果终结符在栈顶而不能匹配，弹出此终结符

4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

  $B \rightarrow d$

4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

  $B \rightarrow d$

abbcde（读入***ab***）

a b

4.4 自下而上分析

4.4.1 归约

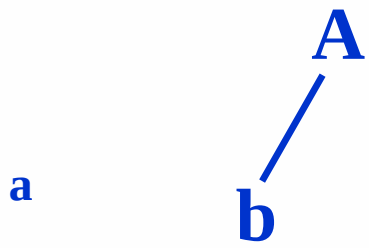
- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

  $B \rightarrow d$

abbcde

aAbcde（归约）



4.4 自下而上分析

4.4.1 归约

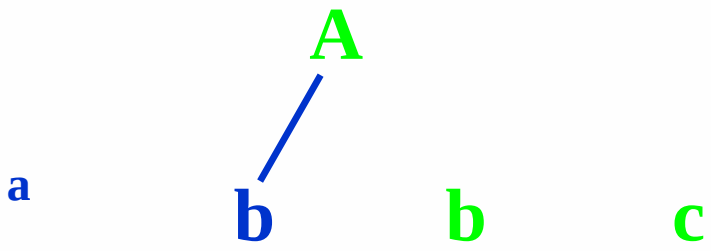
- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

  $B \rightarrow d$

abbcde

aAbcde（再读入***bc***）



4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

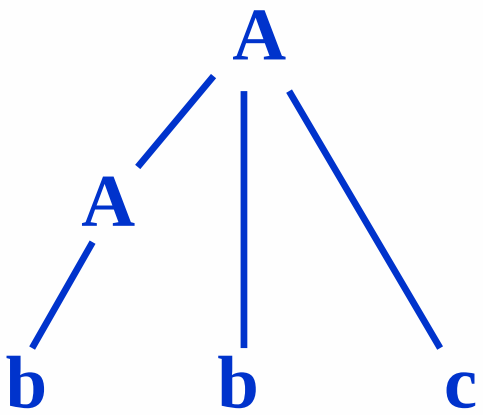
  $B \rightarrow d$

abbcde

aAbcde

aAde（归约）

a



4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

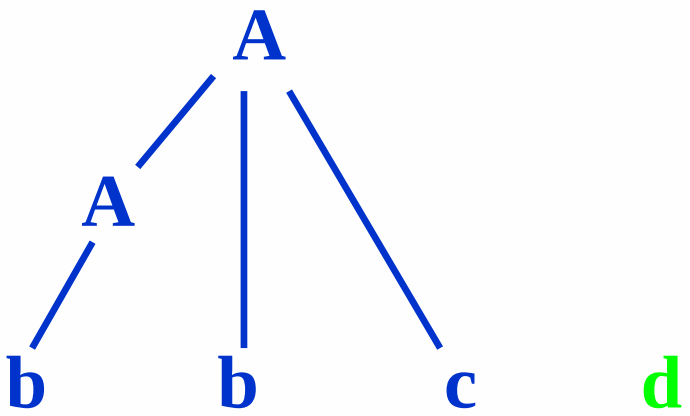
  $B \rightarrow d$

abbcde

aAbcde

aAde（再读入***d***）

a



4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

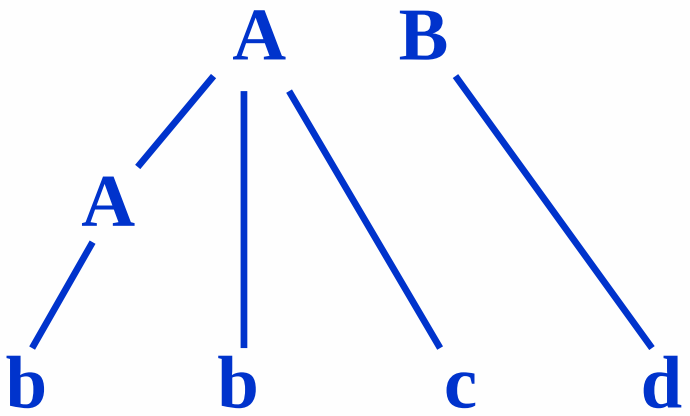
  $B \rightarrow d$

abbcde

aAbcde

aAde

aABe（归约）



4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

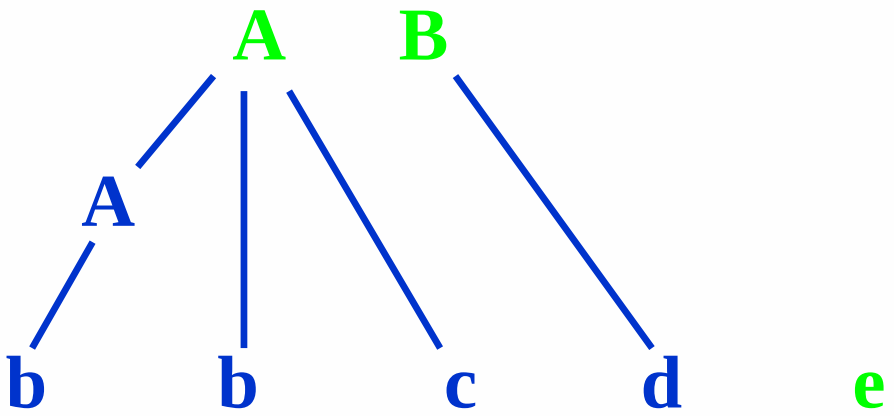
  $B \rightarrow d$

abbcde

aAbcde

aAde

aABe(再读入e)



4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

  $B \rightarrow d$

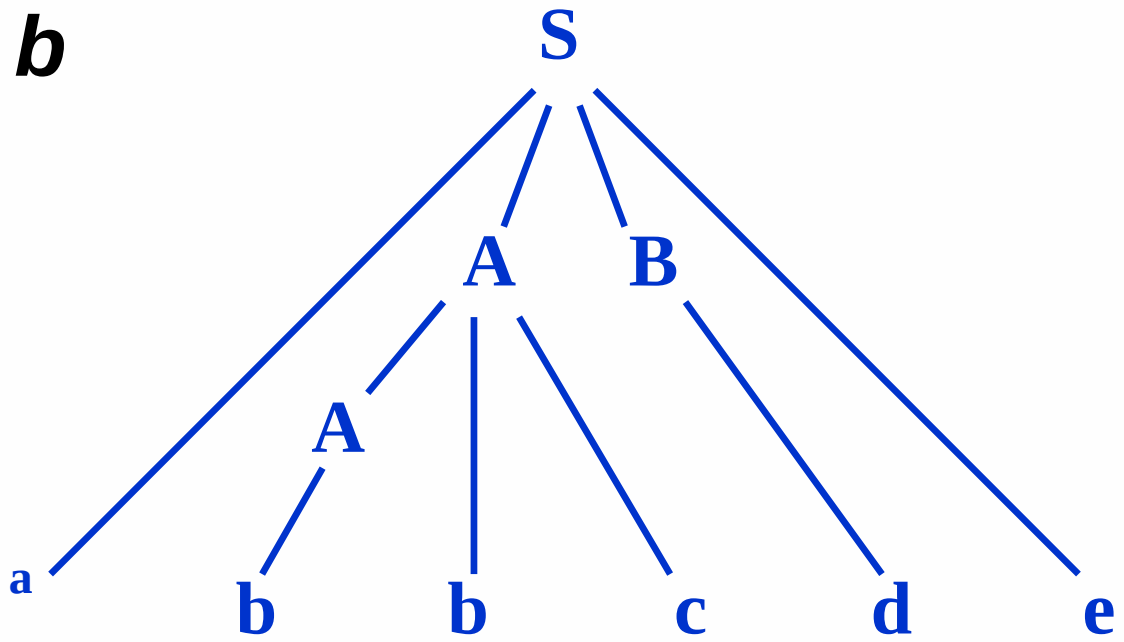
abbcde

aAbcde

aAde

aABe

S（归约）



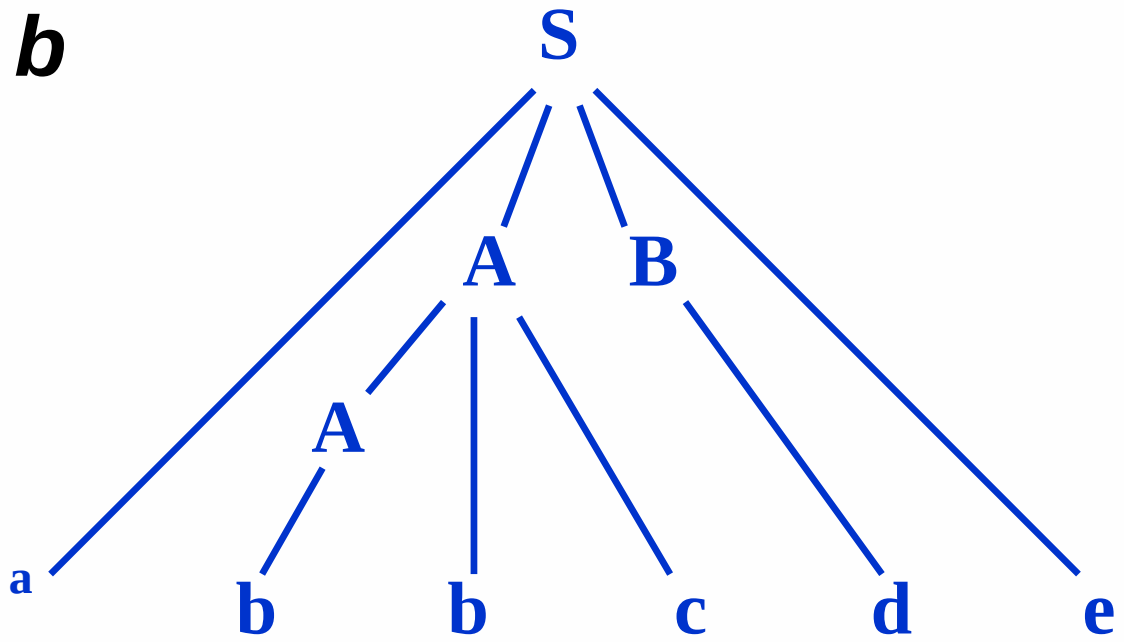
4.4 自下而上分析

4.4.1 归约

- 例 $S \rightarrow aABe$

  $A \rightarrow Abc \mid b$

  $B \rightarrow d$



abbcde

aAbcde

aAde

aABe

S

$S \Rightarrow_{rm} aABe \Rightarrow_{rm} aAde \Rightarrow_{rm} aAbcde \Rightarrow_{rm} abbcde$

4.4.2 句柄

句型的句柄是和某产生式右部匹配的子串，并且，把它归约成该产生式左部的非终结符代表了最右推导过程的逆过程的一步

$S \rightarrow aABe$

$A \rightarrow Abc \mid b$

$B \rightarrow d$

$S \Rightarrow_{rm} aABe \Rightarrow_{rm} aAde \Rightarrow_{rm} aAbcde \Rightarrow_{rm} abbcde$

句柄的右边仅含终结符

如果文法二义，那么句柄可能不唯一

4.4 自下而上分析

- 例 句柄不唯一

  $E \rightarrow E + E \mid E * E \mid (E) \mid \mathrm{id}$

4.4 自下而上分析

- 例 句柄不唯一

  $E \rightarrow E + E \mid E * E \mid (E) \mid \mathrm{id}$

$$
\begin{aligned}
E &\Rightarrow_{rm} E * E \\
&\Rightarrow_{rm} E * E + E \\
&\Rightarrow_{rm} E * E + \mathrm{id}_3 \\
&\Rightarrow_{rm} E * \mathrm{id}_2 + \mathrm{id}_3 \\
&\Rightarrow_{rm} \mathrm{id}_1 * \mathrm{id}_2 + \mathrm{id}_3
\end{aligned}
$$

4.4 自下而上分析

- 例 句柄不唯一

$E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

$E \Rightarrow_{rm} E * E$
$\Rightarrow_{rm} E * E + E$
$\Rightarrow_{rm} E * E + \mathrm{id}_3$
$\Rightarrow_{rm} E * \mathrm{id}_2 + \mathrm{id}_3$
$\Rightarrow_{rm} \mathrm{id}_1 * \mathrm{id}_2 + \mathrm{id}_3$

$E \Rightarrow_{rm} E + E$
$\Rightarrow_{rm} E + \mathrm{id}_3$
$\Rightarrow_{rm} E * E + \mathrm{id}_3$
$\Rightarrow_{rm} E * \mathrm{id}_2 + \mathrm{id}_3$
$\Rightarrow_{rm} \mathrm{id}_1 * \mathrm{id}_2 + \mathrm{id}_3$

在句型$E * E + \mathrm{id}_3$中，句柄不唯一

4.4.3 用栈实现移进–归约分析

先通过

移进–归约分析器在分析输入串$id_1 * id_2 + id_3$时的动作序列

来了解移进–归约分析的工作方式

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E$*	$id_2 + id_3$$	移进
$$E$*$id_2$	$+ id_3$$	按$E \rightarrow$ id归约

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E$*	$id_2 + id_3$$	移进
$$E$*$id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E$*$E$	$+ id_3$$	移进
$$E$*$E$+	$id_3$$	移进

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约
$$E*E+E$	$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约
$$E*E+E$	$$	按$E \rightarrow E+E$归约

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约
$$E*E+E$	$$	按$E \rightarrow E+E$归约
$$E*E$	$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约
$$E*E+E$	$$	按$E \rightarrow E+E$归约
$$E*E$	$$	按$E \rightarrow E*E$归约

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约
$$E*E+E$	$$	按$E \rightarrow E+E$归约
$$E*E$	$$	按$E \rightarrow E*E$归约
$$E$	$$	

4.4 自下而上分析

栈	输　入	动　作
$	$id_1 * id_2 + id_3$$	移进
$ id_1	$* id_2 + id_3$$	按$E \rightarrow$ id归约
$$E$	$* id_2 + id_3$$	移进
$$E*$	$id_2 + id_3$$	移进
$$E*id_2$	$+ id_3$$	按$E \rightarrow$ id归约
$$E*E$	$+ id_3$$	移进
$$E*E+$	$id_3$$	移进
$$E*E+id_3$	$$	按$E \rightarrow$ id归约
$$E*E+E$	$$	按$E \rightarrow E+E$归约
$$E*E$	$$	按$E \rightarrow E*E$归约
$$E$	$$	接受

4.4 自下而上分析

- 要想很好地使用移进–归约方式，尚需解决一些问题
  - 如何决策选择移进还是归约
  - 进行归约时，确定右句型中将要归约的子串
  - 进行归约时，如何确定选择哪一个产生式

4.4.4 移进–归约分析的冲突

1、移进–归约冲突

- 例

stmt → **if *expr* then *stmt***
| **if *expr* then *stmt* else *stmt***
| **other**

如果移进–归约分析器处于格局

栈	输入
… **if *expr* then *stmt***	**else … $**

4.4 自下而上分析

2、归约–归约冲突

stmt → id (*parameter_list*) | *expr* = *expr*

parameter_list → *parameter_list*, *parameter* | *parameter*

parameter → id

expr → id (*expr_list*) | id

expr_list → *expr_list*, *expr* | *expr*

由*A*(*I*, *J*)开始的语句

归约成expr还是parameter？

栈	输入
… id (id	, id)…

4.4 自下而上分析

2、归约–归约冲突

stmt → id (*parameter_list*) | *expr* = *expr*

parameter_list → *parameter_list*, *parameter* | *parameter*

parameter → id

expr → id (*expr_list*) | id

expr_list → *expr_list*, *expr* | *expr*

由*A*(*I*, *J*)开始的语句(词法分析查符号表, 区分第一个id)

栈	输入
… procid (id	, id)…

需要修改上面的文法

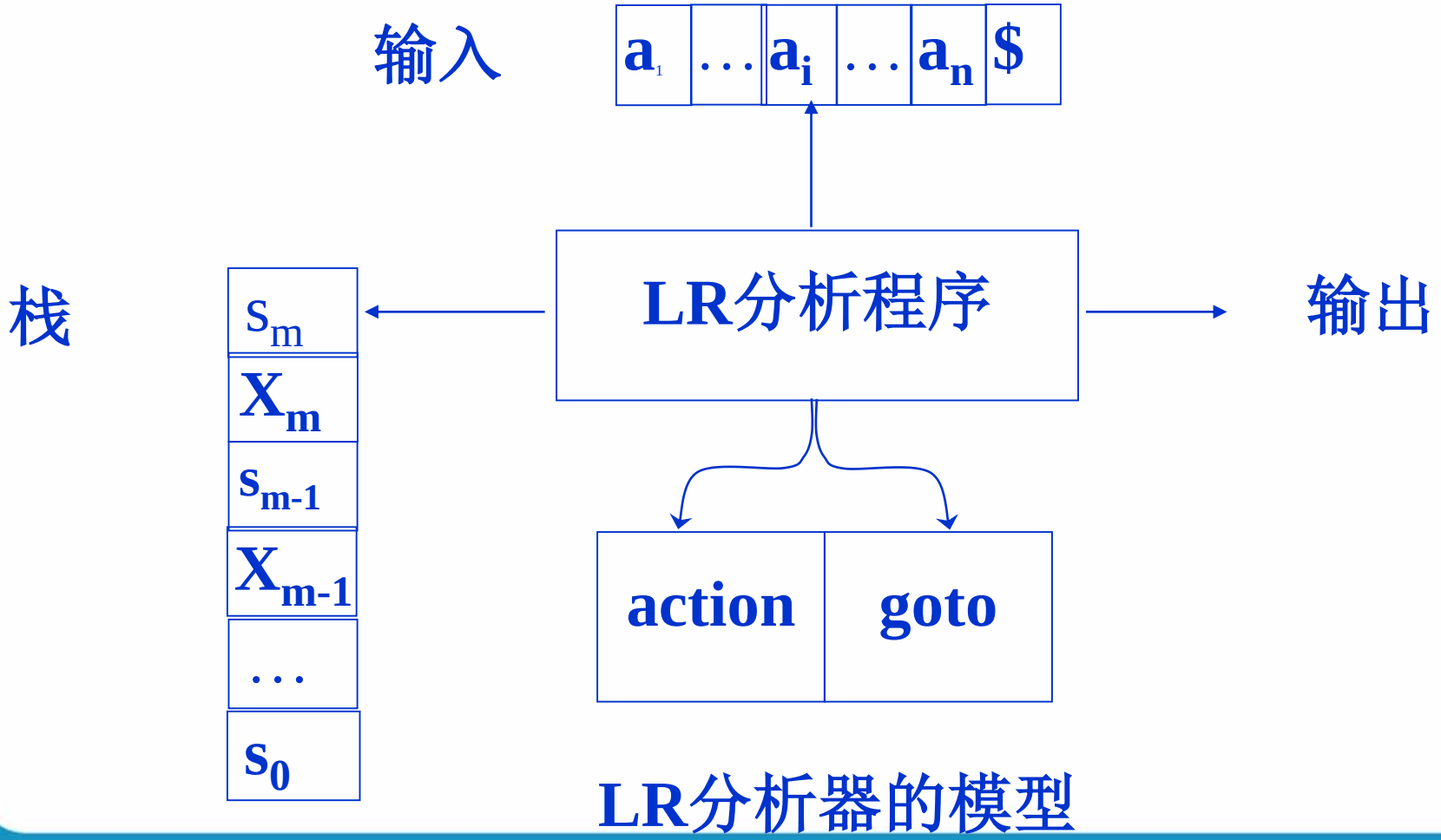
4.5 LR分析器

本节介绍LR(*k*)分析技术

- 特点
  - 适用于一大类上下文无关文法
  - 效率高
- 主要介绍构造LR分析表的三种技术
  - 简单的LR方法（简称SLR）
  - 规范的LR方法
  - 向前看的LR方法（简称LALR）

4.5 LR分析器

4.5.1 LR分析算法



LR分析器的模型

4.5 LR分析器

- 例 $E \to E + T \mid E \to T$
  $T \to T * F \mid T \to E$
  $F \to (E) \mid F \to \text{id}$

状态	动作						转移		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .
0 E 1	$	

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .
0 E 1	$	接受

4.5 LR分析器

4.5.2 LR文法和LR分析方法的特点

1、概念

活前缀：右句型的前缀，该前缀不超过最右句柄的右端

$S \Rightarrow^*_{rm} \gamma A w \Rightarrow_{rm} \gamma \beta w$

$\gamma\beta$的任何前缀（包括ε和$\gamma\beta$本身）都是活前缀

4.5.2 LR文法和LR分析方法的特点

1、概念

活前缀：右句型的前缀，该前缀不超过最右句柄的右端

2、定义

LR文法：我们能为之构造出所有条目都唯一的LR分析表

4.5 LR分析器

3、LR分析方法的特点

- 栈中的文法符号总是形成一个活前缀

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .
0 E 1	$	接受

4.5 LR分析器

3、LR分析方法的特点

- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA

4.5 LR分析器

- 例 $E \to E + T \mid E \to T$
  $T \to T * F \mid T \to E$
  $F \to (E) \mid F \to \text{id}$

下表绿色部分构成识别活前缀DFA的状态转换表

状态	动作						转移		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3

4.5 LR分析器

3、LR分析方法的特点

- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA
- 栈顶的状态符号包含了确定句柄所需要的一切信息

4.5 LR分析器

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .
0 E 1	$	接受

4.5 LR分析器

3、LR分析方法的特点

- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA
- 栈顶的状态符号包含了确定句柄所需要的一切信息
- 是已知的最一般的无回溯的移进–归约方法
- 能分析的文法类是预测分析法能分析的文法类的真超集
- 能及时发现语法错误

- 手工构造分析表的工作量太大

4、LR分析方法和LL分析方法的比较

	LR(1)方法	LL(1)方法
建立分析树的方式	自下而上	自上而下

4.5 LR分析器

4、LR分析方法和LL分析方法的比较

	LR(1)方法	LL(1)方法
建立分析树的方式	自下而上	自上而下
归约还是推导	规范归约	最左推导

4.5 LR分析器

4、LR分析方法和LL分析方法的比较

在下面的推导中，最后一步用的是$A \rightarrow I\beta$

LR(1)决定用该产生式的位置

$$S \Rightarrow_{rm} \cdots \Rightarrow_{rm} \gamma A b w \Rightarrow_{rm} \gamma I \beta b w$$

LL(1)决定用该产生式的位置

4.5 LR分析器

4、LR分析方法和LL分析方法的比较

	LR(1)方法	LL(1)方法
建立分析树的方式	自下而上	自上而下
归约还是推导	规范归约	最左推导
决定使用产生式的时机	看见产生式右部推出的整个终结符串后，才确定用哪个产生式进行归约	看见产生式右部推出的第一个终结符后，便要确定用哪个产生式推导

4.5 LR分析器

4、LR分析方法和LL分析方法的比较

	LR(1)方 法	LL（1)方 法
对文法的显式限制	对文法没有限制	无左递归、无公共左因子

4、LR分析方法和LL分析方法的比较

	LR(1)方法	LL(1)方法
对文法的显式限制	对文法没有限制	无左递归、无公共左因子
分析表比较	状态×文法符号 分析表大	非终结符×终结符 分析表小

4、LR分析方法和LL分析方法的比较

	LR(1)方法	LL(1)方法
对文法的显式限制	对文法没有限制	无左递归、无公共左因子
分析表比较	状态×文法符号 分析表大	非终结符×终结符 分析表小
分析栈比较	状态栈，通常状态比文法符号包含更多信息	文法符号栈

4.5 LR分析器

4、LR分析方法和LL分析方法的比较

	LR(1)方 法	LL(1)方 法
确定句柄	根据栈顶状态和下一个符号便可以确定句柄和归约所用产生式	无句柄概念

4.5 LR分析器

4、LR分析方法和LL分析方法的比较

	LR(1)方法	LL(1)方法
确定句柄	根据栈顶状态和下一个符号便可以确定句柄和归约所用产生式	无句柄概念
语法错误	决不会将出错点后的符号移入分析栈	和LR一样，决不会读过出错点而不报错

4.5.3 构造SLR分析表

- 术语：LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式

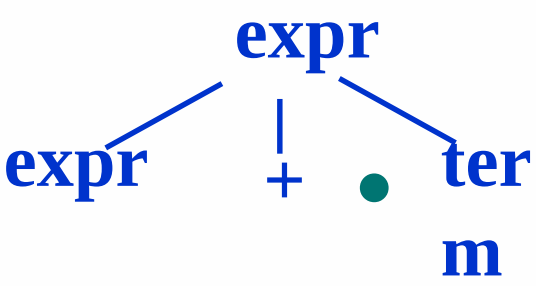
4.5.3 构造SLR分析表

- 术语：LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态

4.5 LR分析器

4.5.3 构造SLR分析表

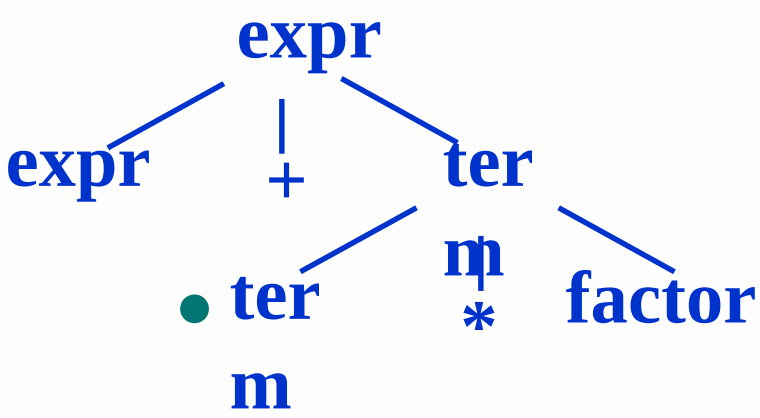
- 术语：LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态



4.5 LR分析器

4.5.3 构造SLR分析表

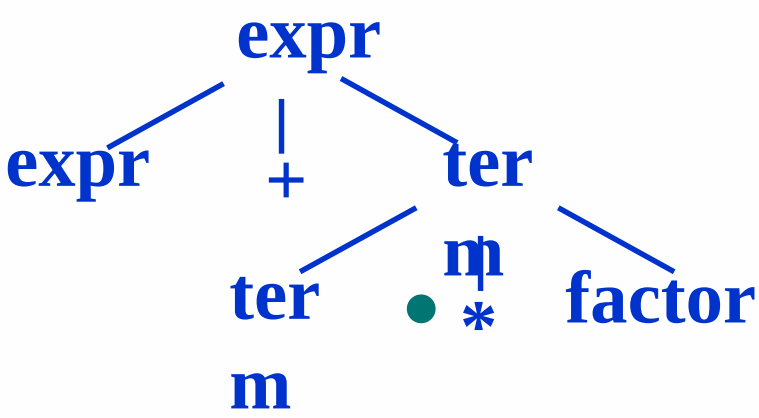
- 术语：LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态



4.5 LR分析器

4.5.3 构造SLR分析表

- 术语：LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态



4.5.3 构造SLR分析表

- 术语：LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态
- 例 $A \rightarrow XYZ$对应有四个项目

  $A \rightarrow \cdot XYZ$

  $A \rightarrow X \cdot YZ$

  $A \rightarrow XY \cdot Z$

  $A \rightarrow XYZ \cdot$
- 例 $A \rightarrow \varepsilon$只有一个项目和它对应

  $A \rightarrow \cdot$

4.5 LR分析器

构造SLR分析表的两大步骤

1、从文法构造识别活前缀的DFA
2、从上述DFA构造分析表

4.5 LR分析器

1、从文法构造识别活前缀的DFA

1）拓广文法

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid \mathrm{id}$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

1）拓广文法

$E' \rightarrow E$

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid \mathrm{id}$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$	（核心项目）
$E \rightarrow \cdot E + T$	
$E \rightarrow \cdot T$	（非核心项目，
$T \rightarrow \cdot T * F$	通过对核心项目求闭包
$T \rightarrow \cdot F$	而获得）
$F \rightarrow \cdot (E)$	
$F \rightarrow \cdot \mathrm{id}$	

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

$\xrightarrow{E}$

I_1:

$E' \to E \cdot$

$E \to E \cdot + T$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

$\xrightarrow{E}$

I_1:

$E' \to E \cdot$

$E \to E \cdot + T$

$I_1 := goto(I_0, E)$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

$\xrightarrow{E}$ I_1:

$E' \to E\cdot$

$E \to E\cdot + T$

$\xrightarrow{T}$ I_2:

$E \to T\cdot$

$T \to T\cdot * F$

4.5 LR分析器

1、从文法构造识别活前缀的DFA

2）构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

$\xrightarrow{E}$ I_1:

$E' \to E\cdot$

$E \to E\cdot + T$

$\xrightarrow{T}$ I_2:

$E \to T\cdot$

$T \to T\cdot * F$

$\xrightarrow{F}$ I_3:

$T \to F\cdot$

4.5 LR分析器

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

(

I_4:

$F \rightarrow (\cdot E\ )$

4.5 LR分析器

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

I_4:

$F \rightarrow (\cdot E)$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

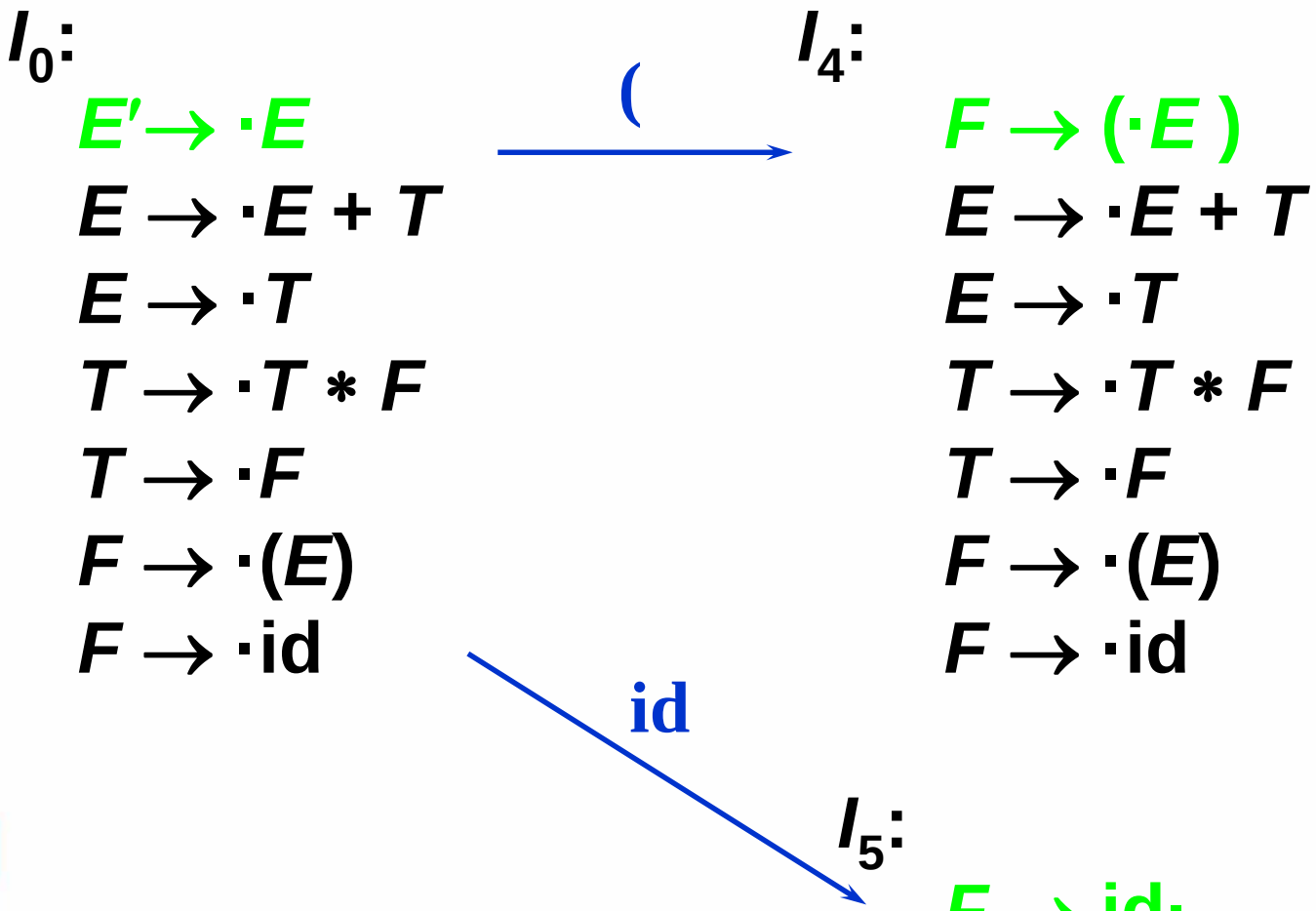
$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

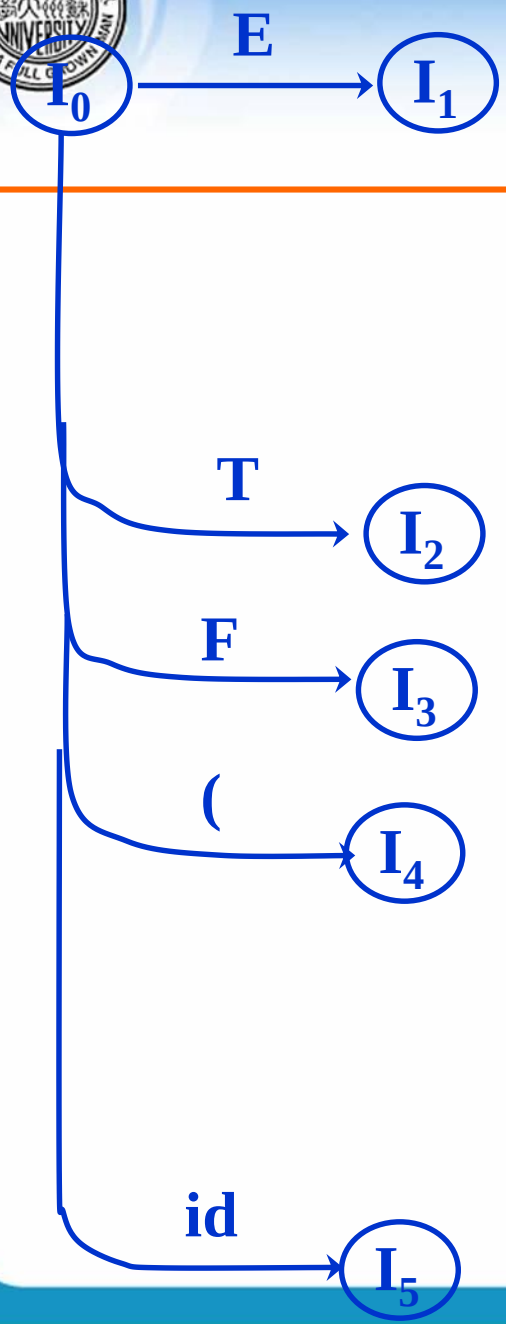
$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

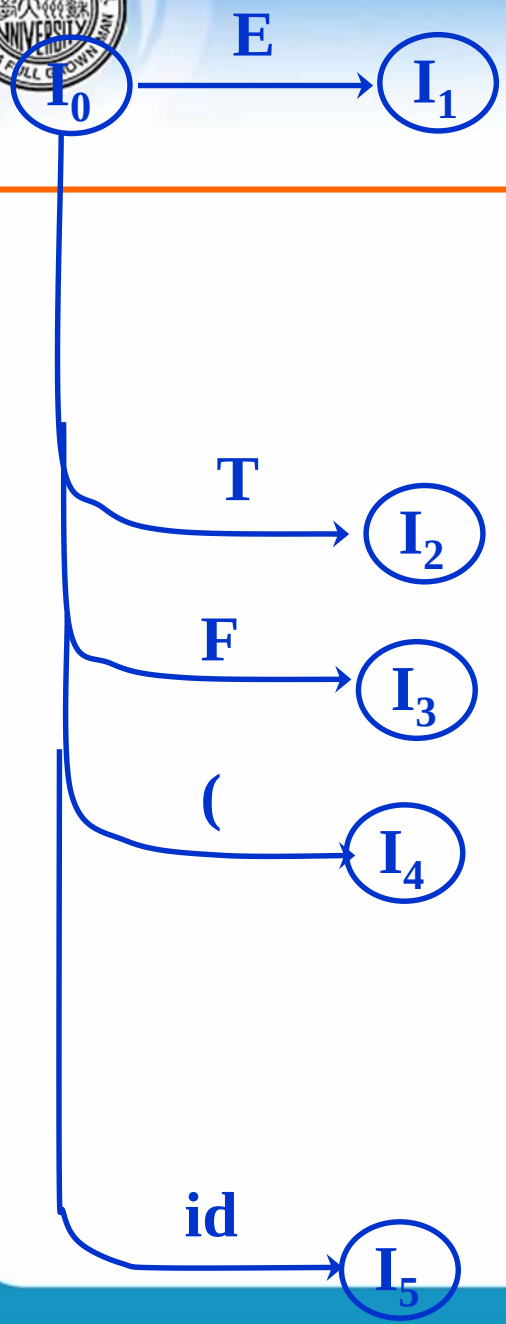
4.5 LR分析器


I_0:
$E' \to \cdot E$
$E \to \cdot E + T$
$E \to \cdot T$
$T \to \cdot T * F$
$T \to \cdot F$
$F \to \cdot(E)$
$F \to \cdot \mathrm{id}$
(
I_4:
$F \to (\cdot E)$
$E \to \cdot E + T$
$E \to \cdot T$
$T \to \cdot T * F$
$T \to \cdot F$
$F \to \cdot(E)$
$F \to \cdot \mathrm{id}$
id
I_5:
$F \to \mathrm{id}\cdot$

4.5 LR分析器



4.5 LR分析器

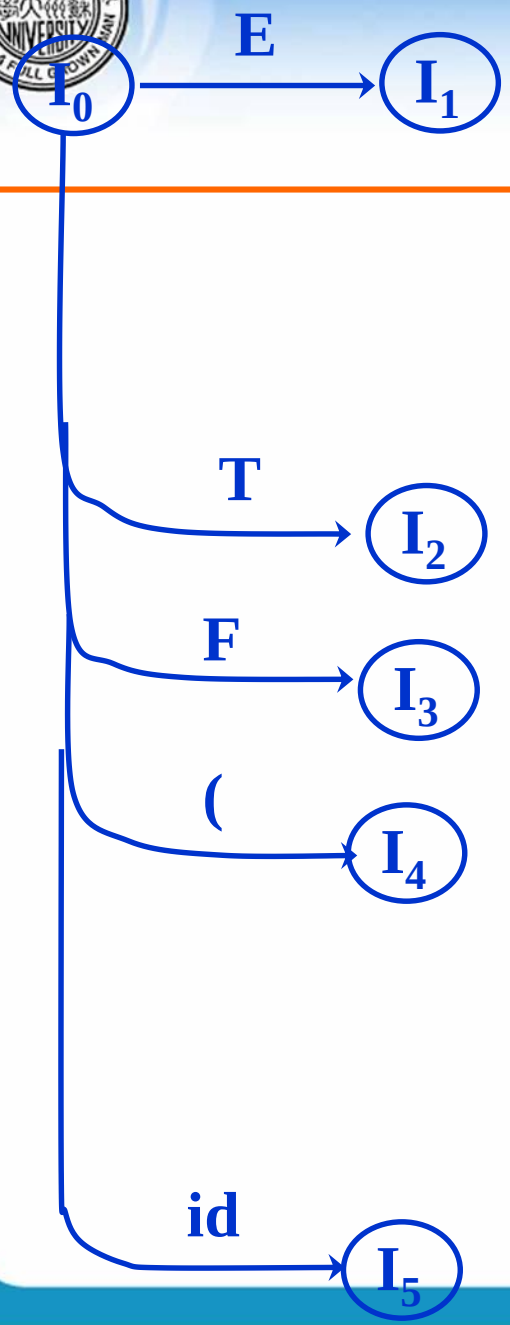


I_1:

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

4.5 LR分析器



I_1:

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

I_6 :

$E \rightarrow E + \cdot T$

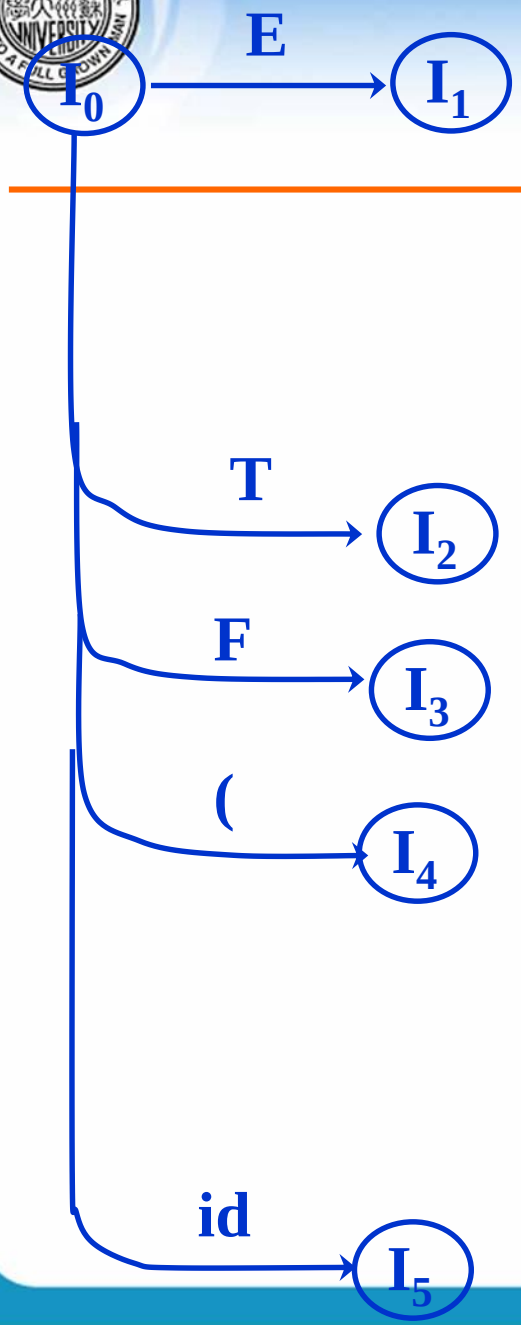
$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

4.5 LR分析器



I_1:

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

I_2:

$E \rightarrow T \cdot$

$T \rightarrow T \cdot * F$

I_6:

$E \rightarrow E + \cdot T$

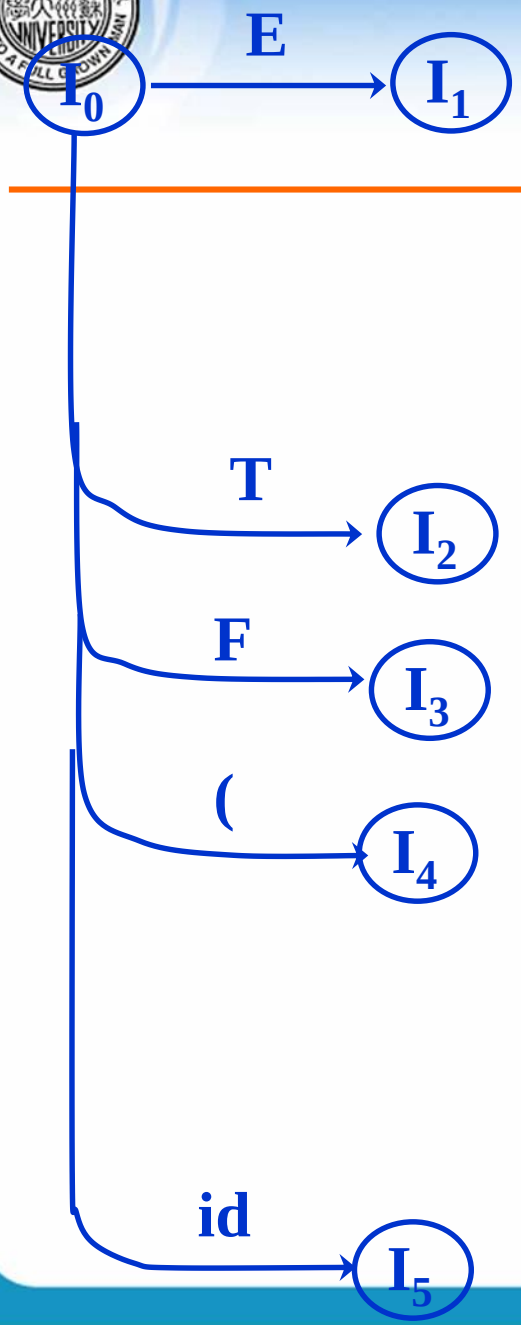
$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

4.5 LR分析器



I_1:

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

I_2:

$E \rightarrow T \cdot$

$T \rightarrow T \cdot * F$

I_6:

$E \rightarrow E + \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

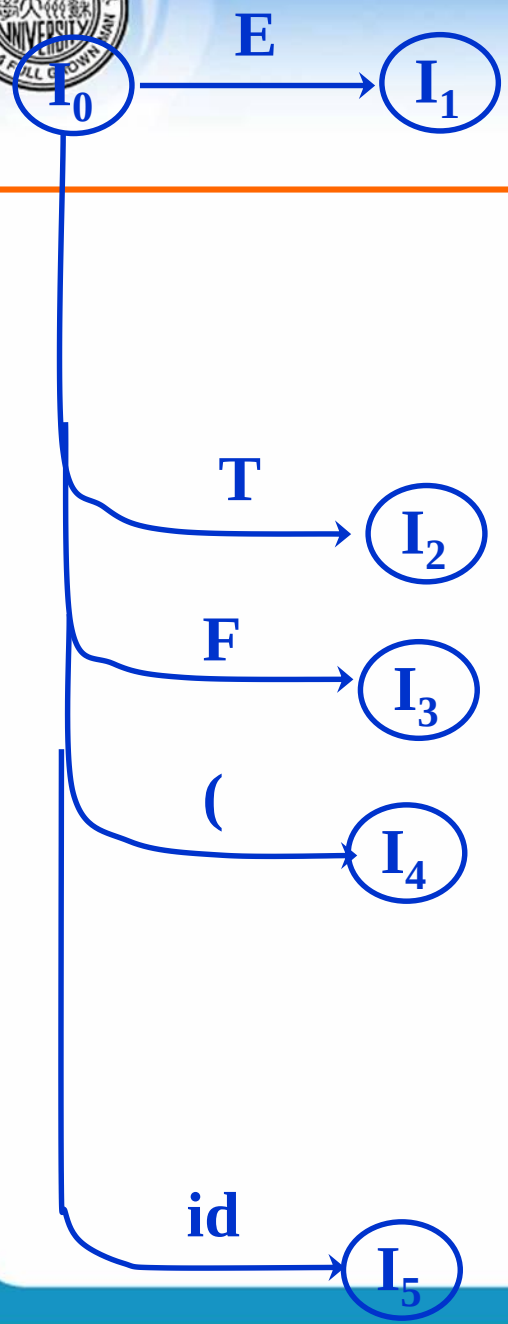
I_7:

$T \rightarrow T * \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

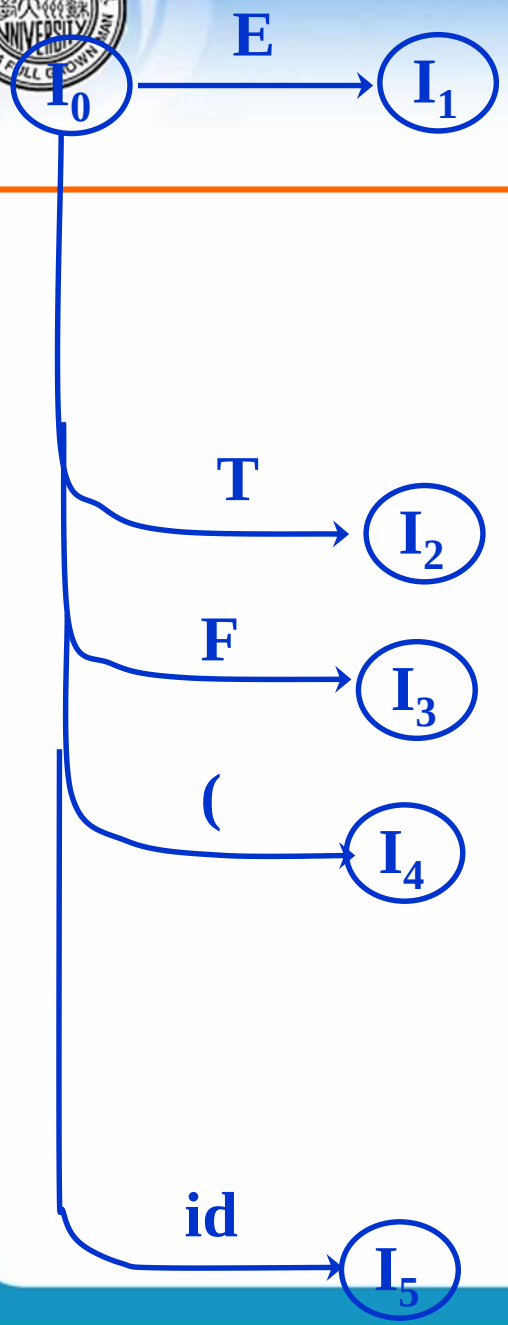
4.5 LR分析器



I_3:

$T \rightarrow F \cdot$ 无状态转换

4.5 LR分析器



I_4:

$F \rightarrow (\cdot E)$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

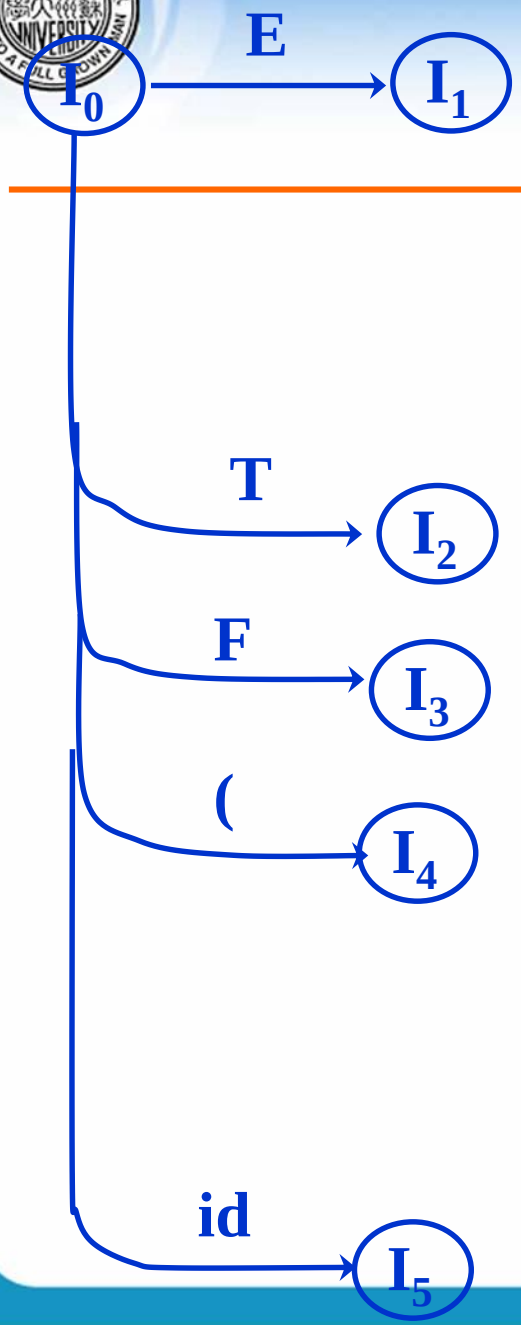
$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

4.5 LR分析器



I_4:

$F \rightarrow (\cdot E)$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

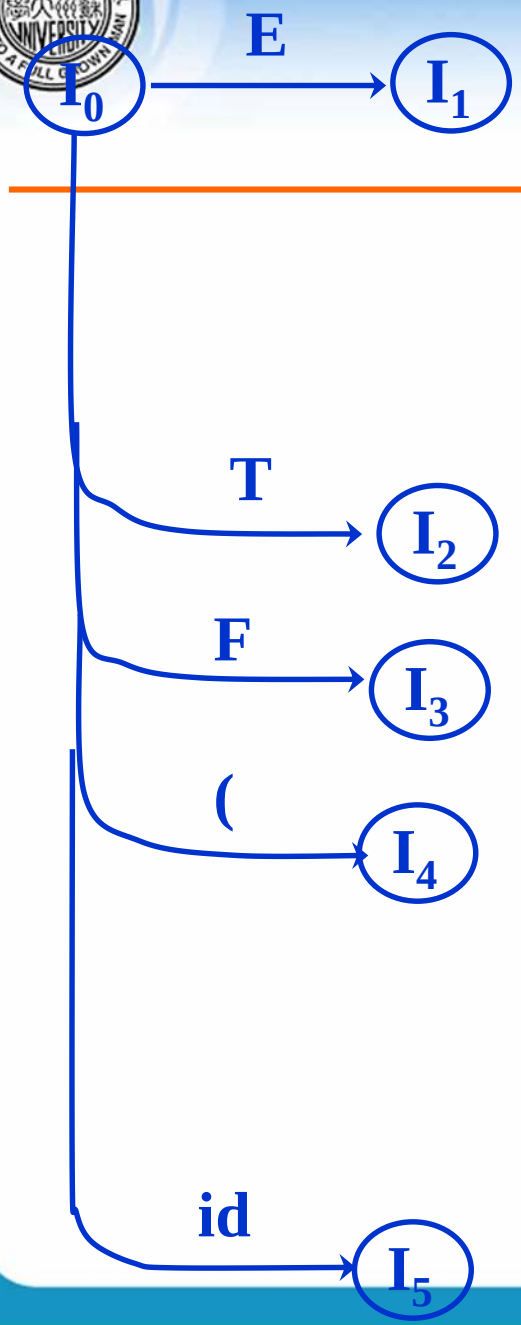
E

I_8:

$F \rightarrow (E \cdot)$

$E \rightarrow E \cdot + T$

4.5 LR分析器



I_4:

$F \rightarrow (\cdot E)$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

E

I_8:

$F \rightarrow (E \cdot)$

$E \rightarrow E \cdot + T$

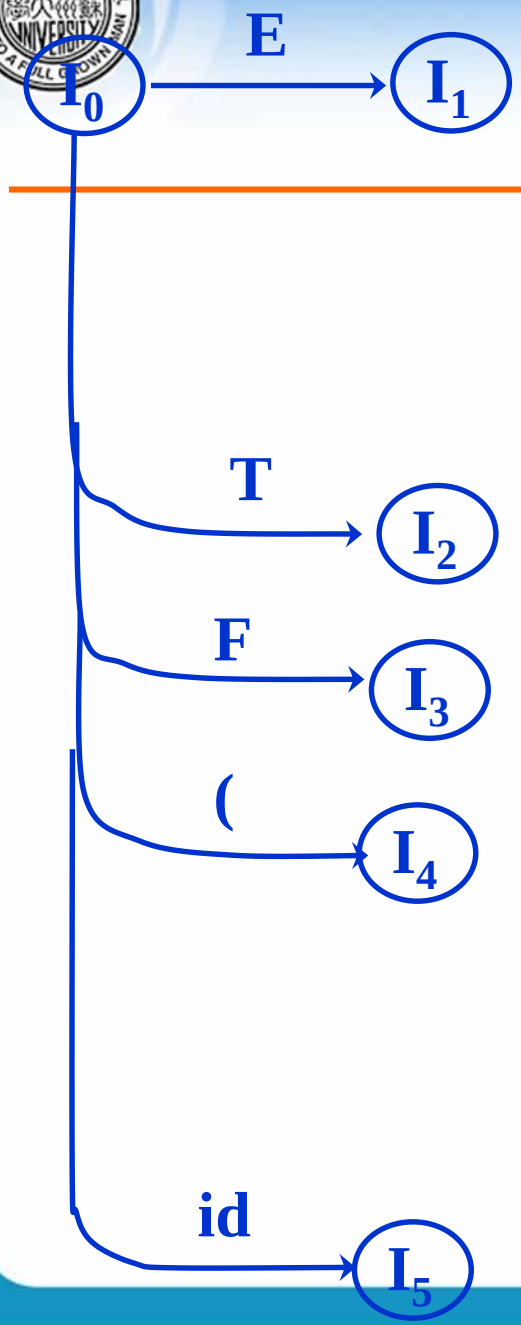
I_2:

$E \rightarrow T \cdot$

$T \rightarrow T \cdot * F$

T

4.5 LR分析器



I_4:

$F \to (\cdot E)$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot id$

I_8:

$F \to (E \cdot)$

$E \to E \cdot + T$

I_2:

$E \to T \cdot$

$T \to T \cdot * F$

I_3:

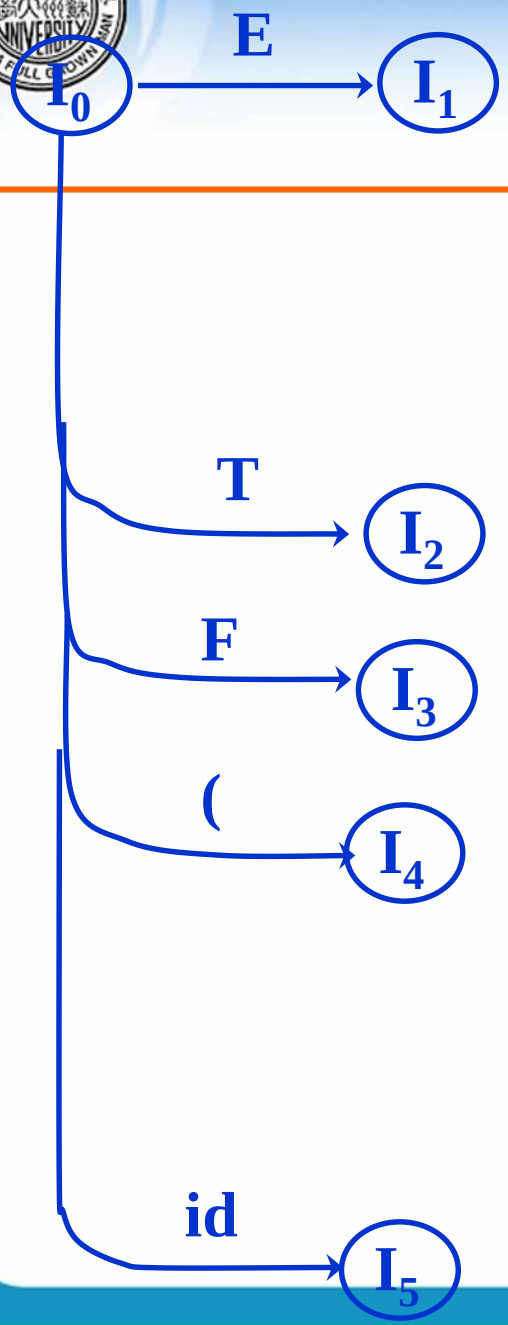
$T \to F \cdot$

E

T

F

4.5 LR分析器



I_4:

$F \to (\cdot E)$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot id$

I_8:

$F \to (E \cdot)$

$E \to E \cdot + T$

I_2:

$E \to T \cdot$

$T \to T \cdot * F$

I_3:

$T \to F \cdot$

E

T

F

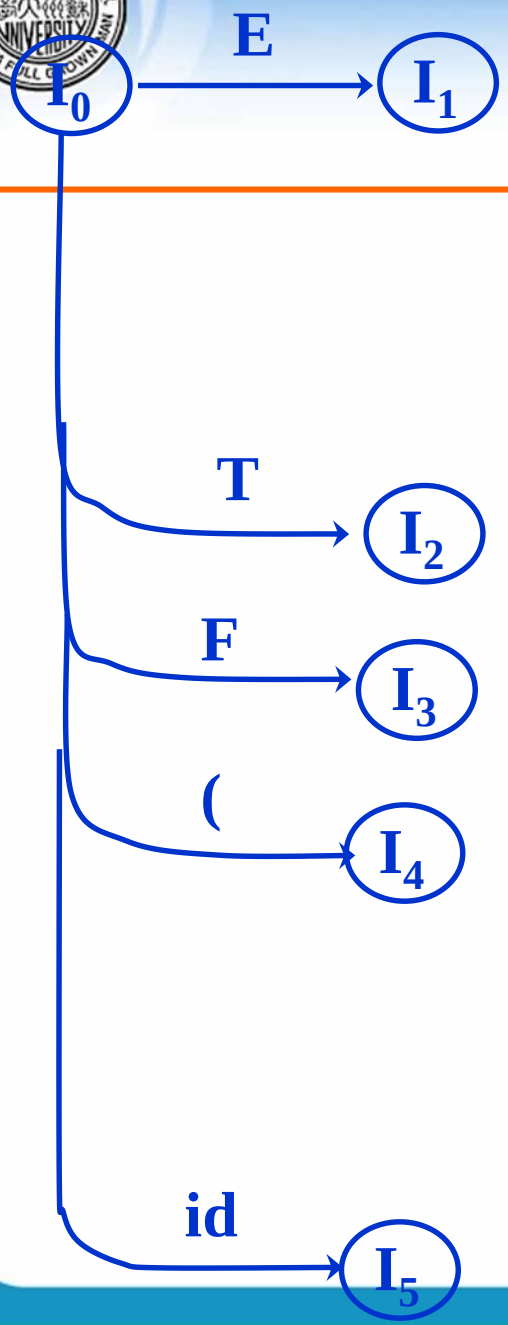
I_4:

$F \to (\cdot E)$

$F \to (\cdot E)$

...

4.5 LR分析器



I_4:

$F \to (\cdot E)$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot id$

I_8:

$F \to (E \cdot)$

$E \to E \cdot + T$

I_2:

$E \to T \cdot$

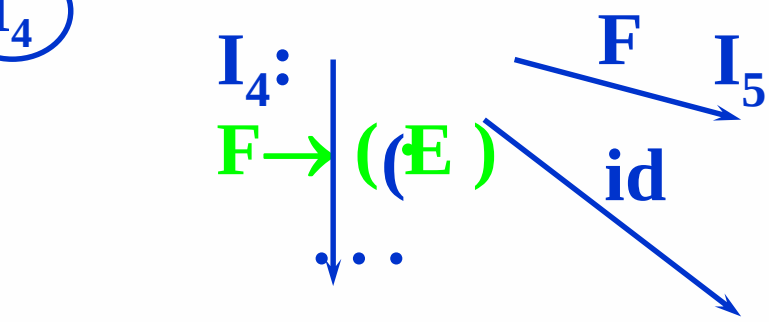
$T \to T \cdot * F$

I_3:

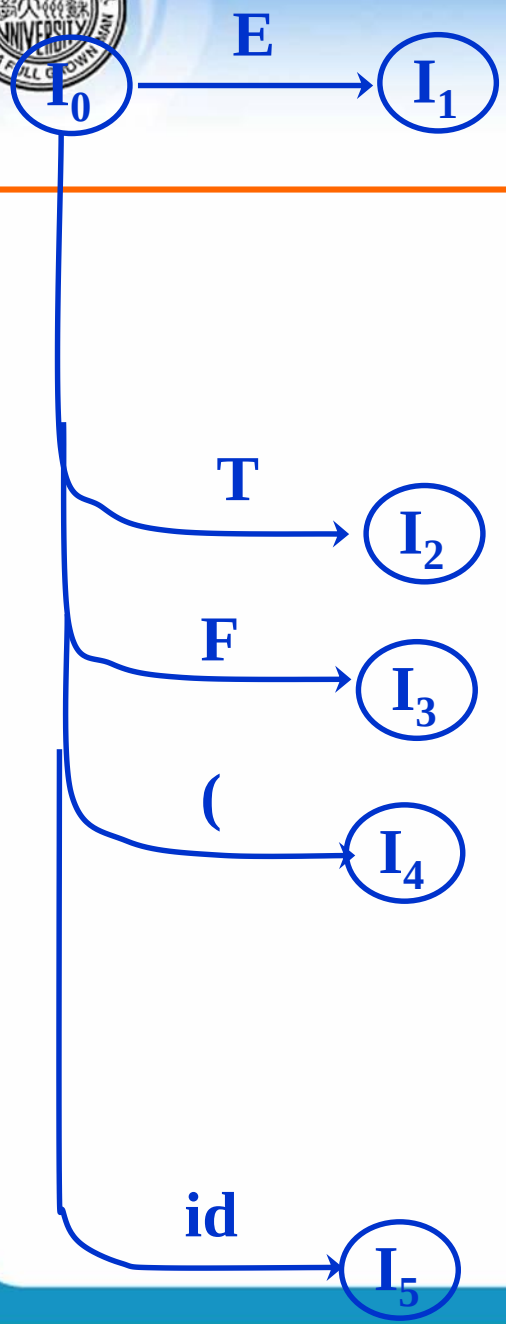
$T \to F \cdot$

I_5:

$F \to id \cdot$



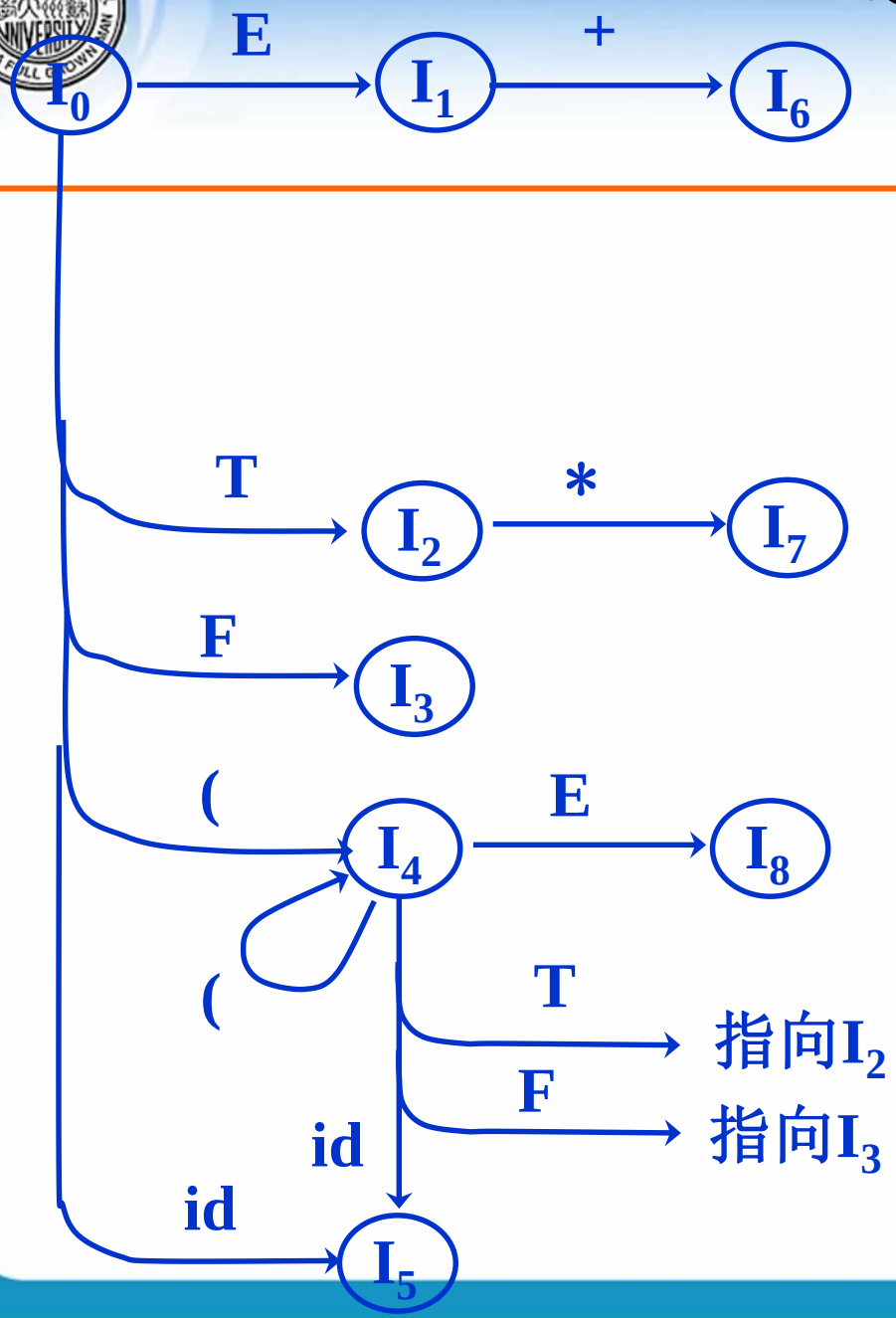
4.5 LR分析器



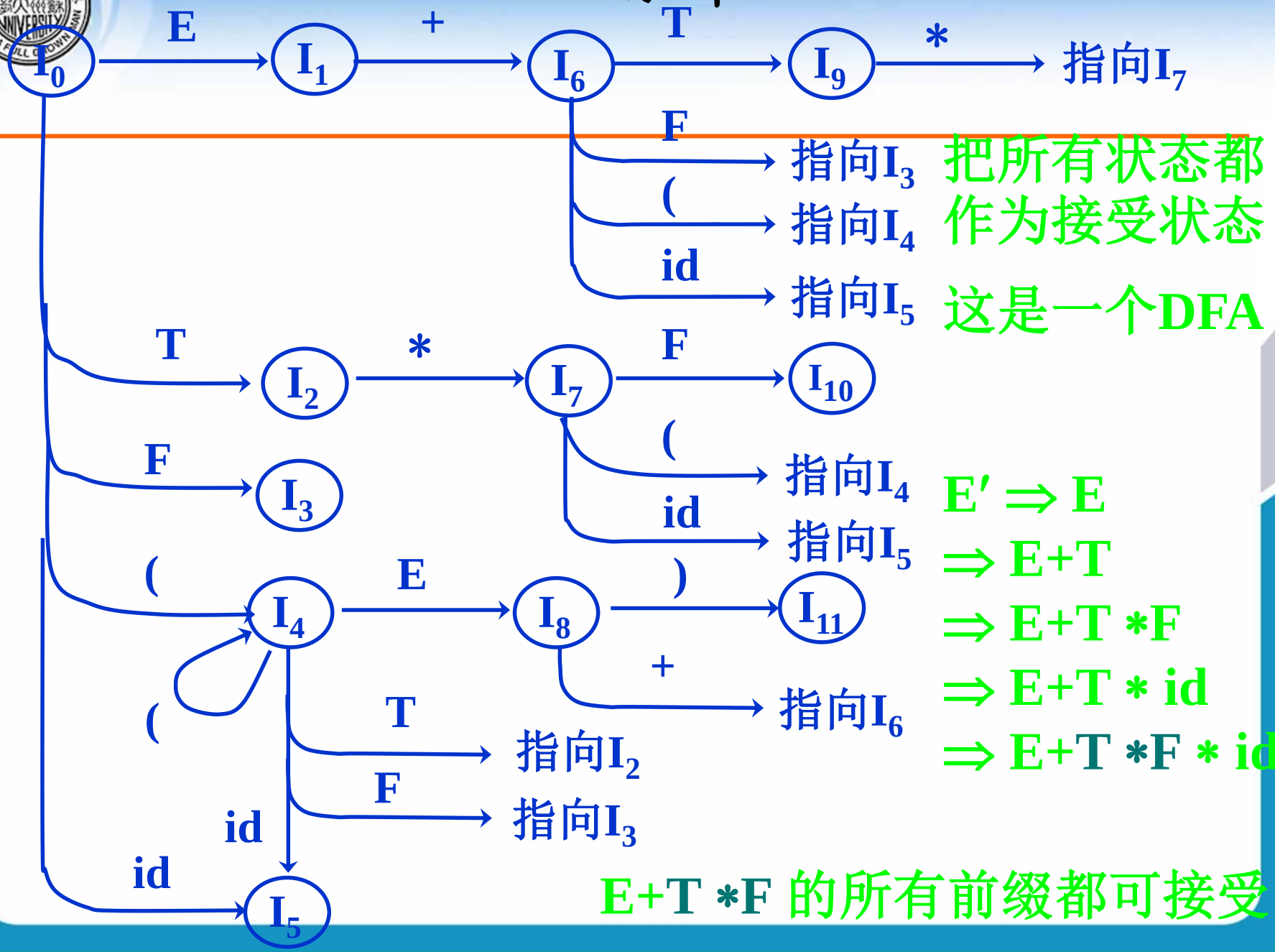
I_5:

$F \rightarrow id \cdot$ 无状态转换

4.5 LR分析器



4.5 LR分析器


E
+
T
*
指向I_7
F
指向I_3
(
指向I_4
id
指向I_5
把所有状态都
作为接受状态
这是一个DFA
T
*
F
I_{10}
F
I_3
(
指向I_4
id
指向I_5
E′ ⇒ E
⇒ E+T
⇒ E+T *F
⇒ E+T * id
⇒ E+T *F * id
(
E
)
I_{11}
+
指向I_6
(
T
指向I_2
F
指向I_3
id
id
I_5
E+T *F 的所有前缀都可接受

4.5 LR分析器

也可以构造一个识别活前缀的NFA N

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

$E' \rightarrow \cdot E$

4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

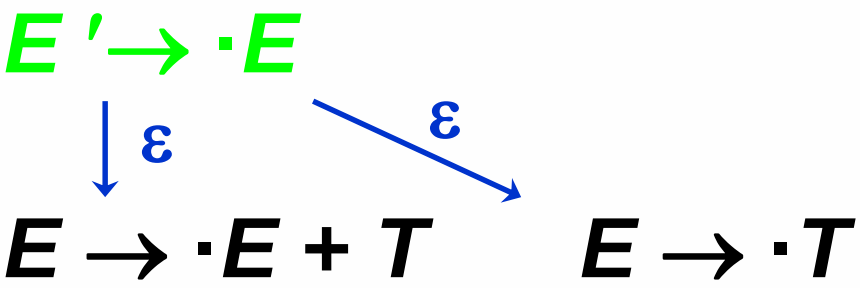
$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$



4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

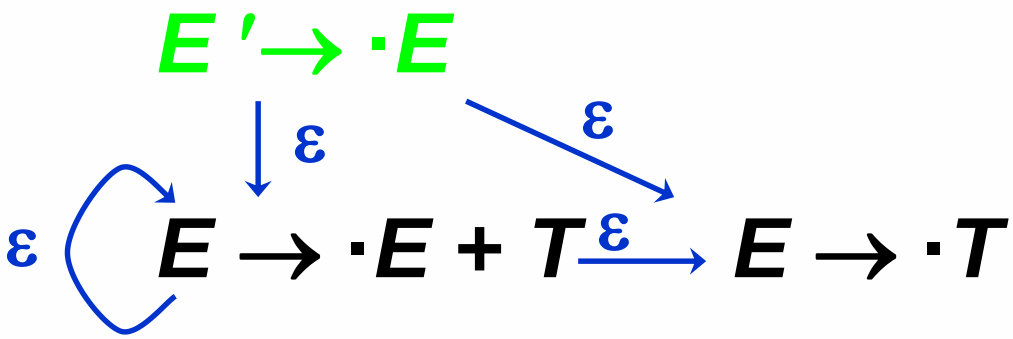
$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$



4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

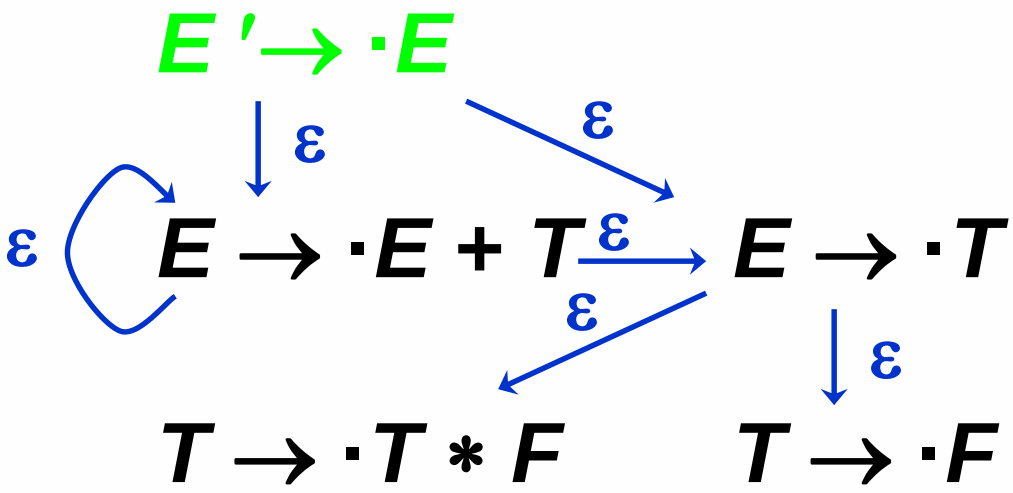
$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$



4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

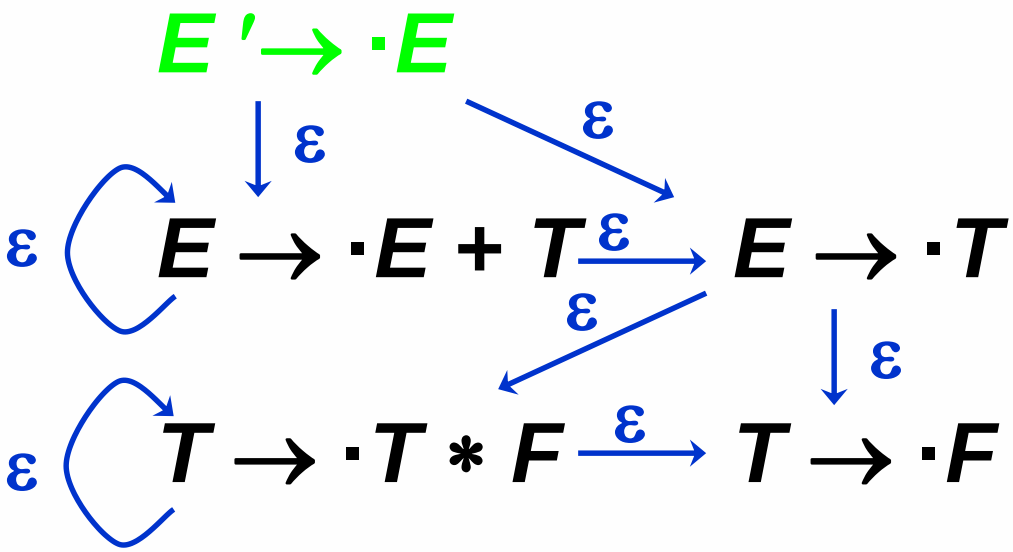
$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$



4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

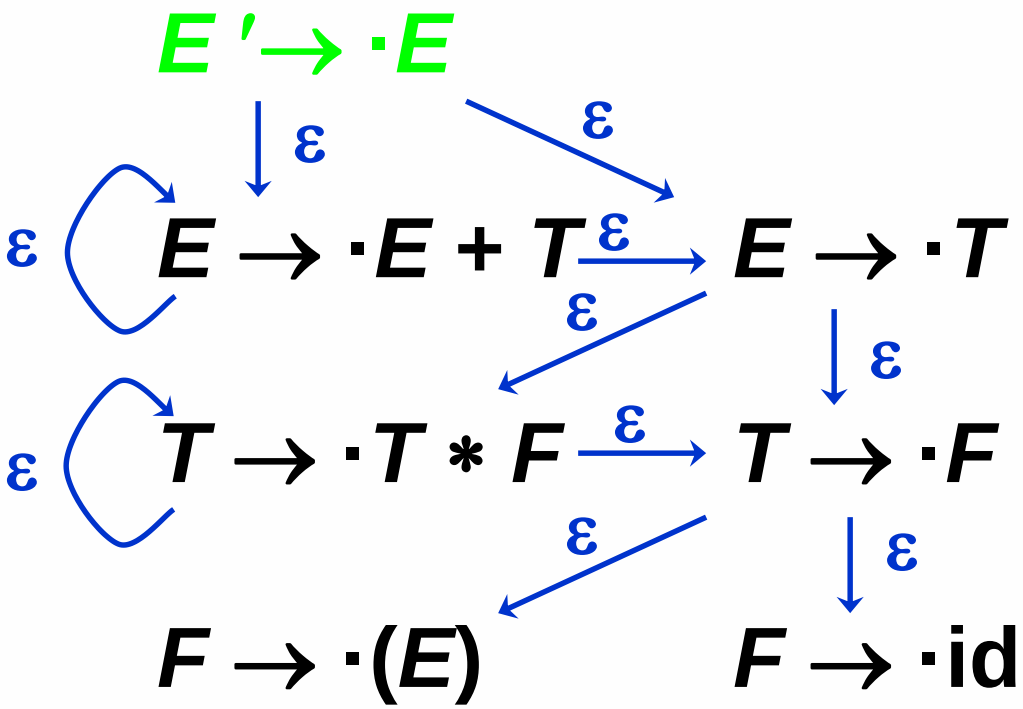
$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$



4.5 LR分析器

也可以构造一个识别活前缀的NFA N

每个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

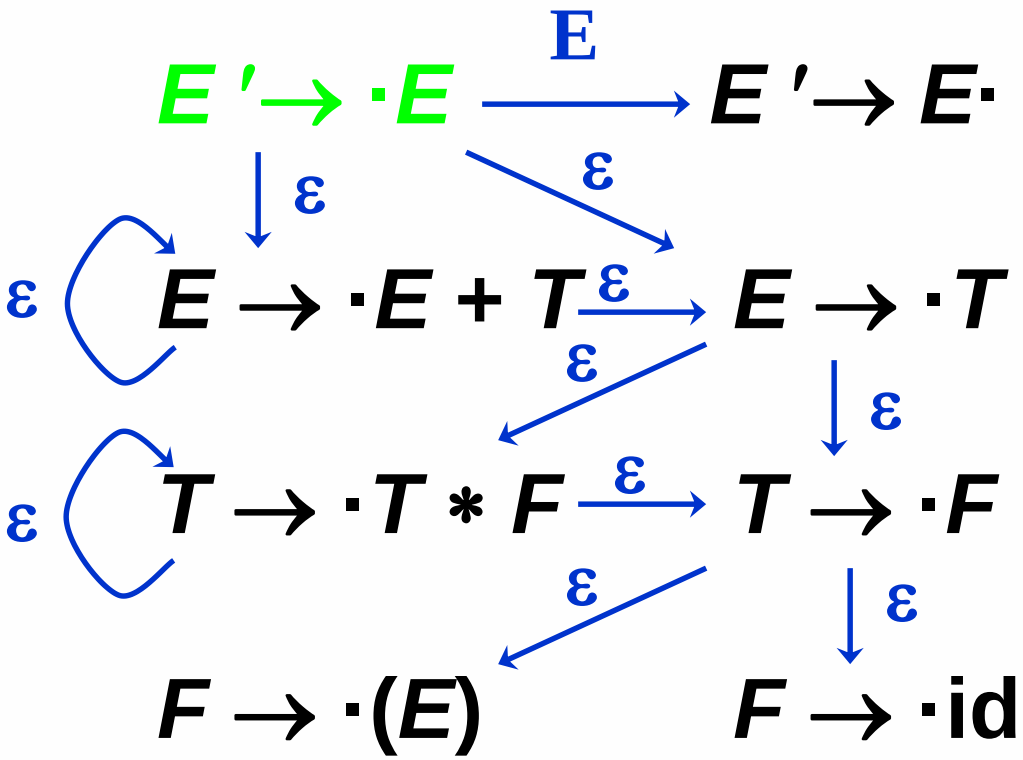
$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$



4.5 LR分析器

从文法构造的识别活前缀的DFA的一些特点

- 概念：有效项目

  如果$S' \Rightarrow^*_{rm} \alpha A w \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的

（1）一个项目可能对好几个活前缀都是有效的

$E \rightarrow \cdot E+T$对 ε 和 (这两个活前缀都有效

$E' \Rightarrow E \Rightarrow E+T$　　(α，β_1都为空)

$E' \Rightarrow E \Rightarrow (E) \Rightarrow (E+T)$　　(α ="(", β_1为空)

该DFA读过ε 和(后到达不同的状态，那么项目$E \rightarrow \cdot E+T$就出现在对应的不同项目集中

4.5 LR分析器

从文法构造的识别活前缀的DFA的一些特点

- 概念：有效项目

  如果$S' \Rightarrow^*_{rm} \alpha A w \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的

（1）一个项目可能对好几个活前缀都是有效的

从项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$有效这个事实可以知道

- 如果$\beta_2 \neq \varepsilon$，应该移进
- 如果$\beta_2 = \varepsilon$，应该用产生式$A \rightarrow \beta_1$归约

4.5 LR分析器

从文法构造的识别活前缀的DFA的一些特点

- 概念：有效项目

  如果$S' \Rightarrow^*_{rm} \alpha A w \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的

（1）一个项目可能对好几个活前缀都是有效的

（2）一个活前缀可能有多个有效项目

一个活前缀γ的有效项目集就是从这个DFA的初态出发，沿着标记为γ的路径到达的那个项目集（状态）

4.5 LR分析器

- 例 串$E + T *$是活前缀，读完它后，DFA处于状态I_7

I_7: $T \to T * \cdot F$, $F \to \cdot (E)$, $F \to \cdot \mathrm{id}$

$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T * F \Rightarrow E+T * \mathrm{id} \Rightarrow E+T * F * \mathrm{id}$

$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T * F \Rightarrow E+T * (E)$

$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T * F \Rightarrow E+T * \mathrm{id}$

4.5 LR分析器

构造SLR分析表的两大步骤

1、从文法构造识别活前缀的DFA
2、从上述DFA构造分析表

4.5 LR分析器

2、从DFA构造SLR分析表

- 状态i从I_i构造，它的**action**函数如下确定：
  - 如果$[A\to\alpha\cdot a\beta]$在I_i中，并且$goto(I_i, a) = I_j$，那么置$action[i, a]$为sj
  - 如果$[A\to\alpha\cdot]$在I_i中，那么对FOLLOW(A)中的所有a，置$action[i, a]$为rj, j是产生式 $A\to\alpha$的编号
  - 如果$[S'\to S\cdot]$在I_i中，那么置$action[i, \$]$为接受$acc$

如果出现动作冲突，那么该文法就不是SLR(1)的

4.5 LR分析器

2、从DFA构造SLR分析表

- 状态i从I_i构造，它的**action**函数如下确定：
  - . . .
- 使用下面规则构造状态i的**goto**函数：
  - 对所有的非终结符A，如果$goto(I_i, A) = I_j$，那么$goto[i, A] = j$

4.5 LR分析器

2、从DFA构造SLR分析表

- 状态*i*从I_i构造，它的*action*函数如下确定：
  - . . .
- 使用下面规则构造状态*i*的*goto*函数：
  - . . .
- 不能由上面两步定义的条目都置为error
- 分析器的初始状态是包含[$S' \rightarrow \cdot S$]的项目集对应的状态

4.5 LR分析器

- 例 I_2:

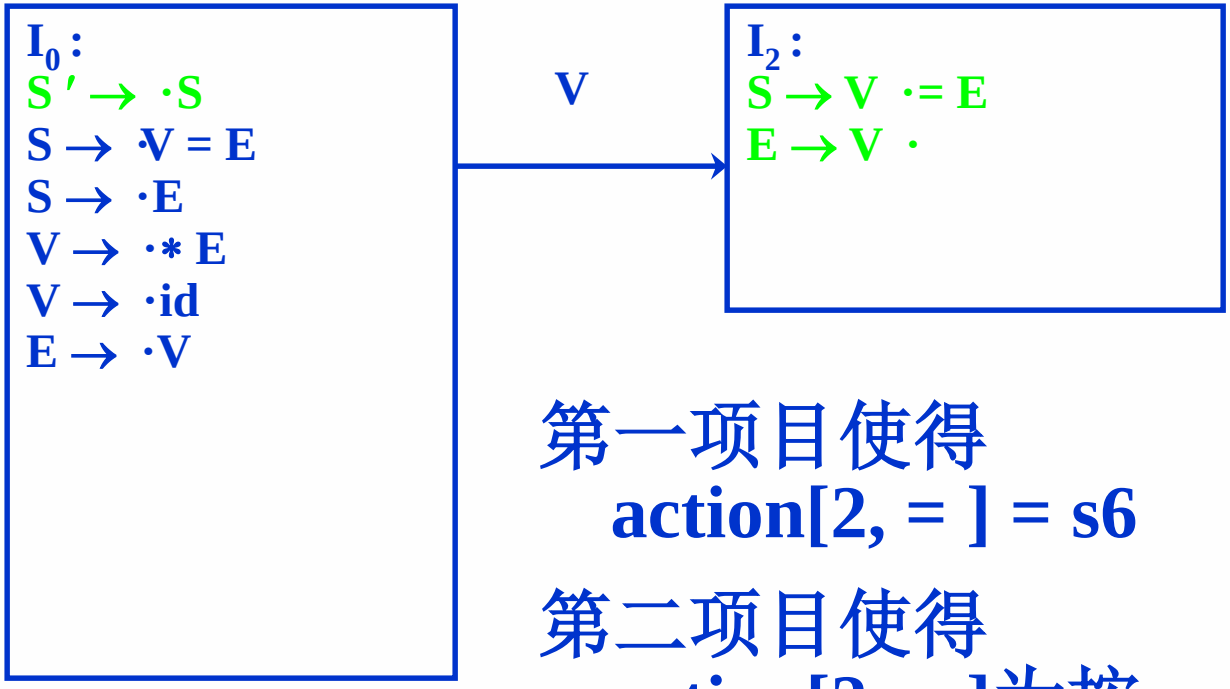
$$E \rightarrow T\cdot$$
$$T \rightarrow T\cdot * F$$

  - 因为FOLLOW(*E*) = {$, +,)}, 所以
    action[2, $]=*action*[2, +]=*action*[2,)]=*r*2
  - *action*[2, *] = *s*7

4.5 LR分析器

- 例 SLR(1)文法的描述能力有限

$S \to V = E$
$S \to E$
$V \to * E$
$V \to id$
$E \to V$



第一项目使得
action[2, =] = s6

第二项目使得
action[2, =]为按E→V归约，因为=是E的一个后继符

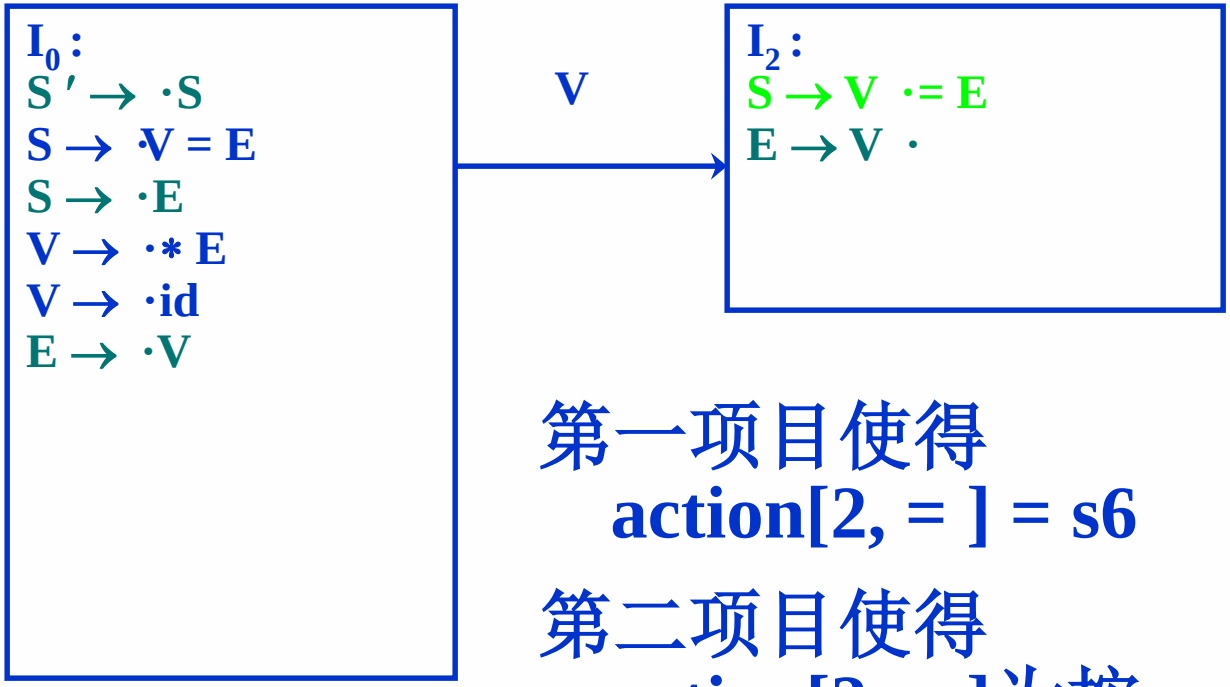
=是E的一个后继符：

$S \$\Rightarrow V = E \$ \Rightarrow * E = E \$$

4.5 LR分析器

- 例 SLR(1)文法的描述能力有限

$S \to V = E$
$S \to E$
$V \to * E$
$V \to id$
$E \to V$



第一项目使得
action[2, =] = s6

第二项目使得
action[2, =]为按E→V归约，因为=是E的一个后继符

在所关注场合E的后继是$:

$S \$ \Rightarrow V = E \$ \Rightarrow * E = E \$$

$S \$ \Rightarrow E \$ \Rightarrow V \$$

4.5 LR分析器

4.5.4 构造规范的LR分析表

- LR(1)项目：

  重新定义项目，让它带上搜索符，成为如下形式

  $[A \rightarrow \alpha \cdot \beta, a]$

- LR(1)项目$[A \rightarrow \alpha \cdot \beta, a]$对活前缀$\gamma$有效：

  如果存在着推导$S \Rightarrow^*_{rm} \delta Aw \Rightarrow_{rm} \delta\alpha\beta w$，其中：

  - $\gamma = \delta\alpha$;
  - a是w的第一个符号，或者w是ε且a是\$

4.5 LR分析器

- 例 $S \rightarrow BB$

  $B \rightarrow bB \mid a$

从最右推导$S \Rightarrow^*_{rm} bbBba \Rightarrow_{rm} bbbBba$看出：

$[B \rightarrow b \cdot B, b]$对活前缀$\gamma = bbb$是有效的

对于项目$[A \rightarrow \alpha \cdot \beta, a]$，当$\beta$为空时，是根据搜索符$a$来填表（归约项目），而不是根据$A$的后继符来填表

4.5 LR分析器

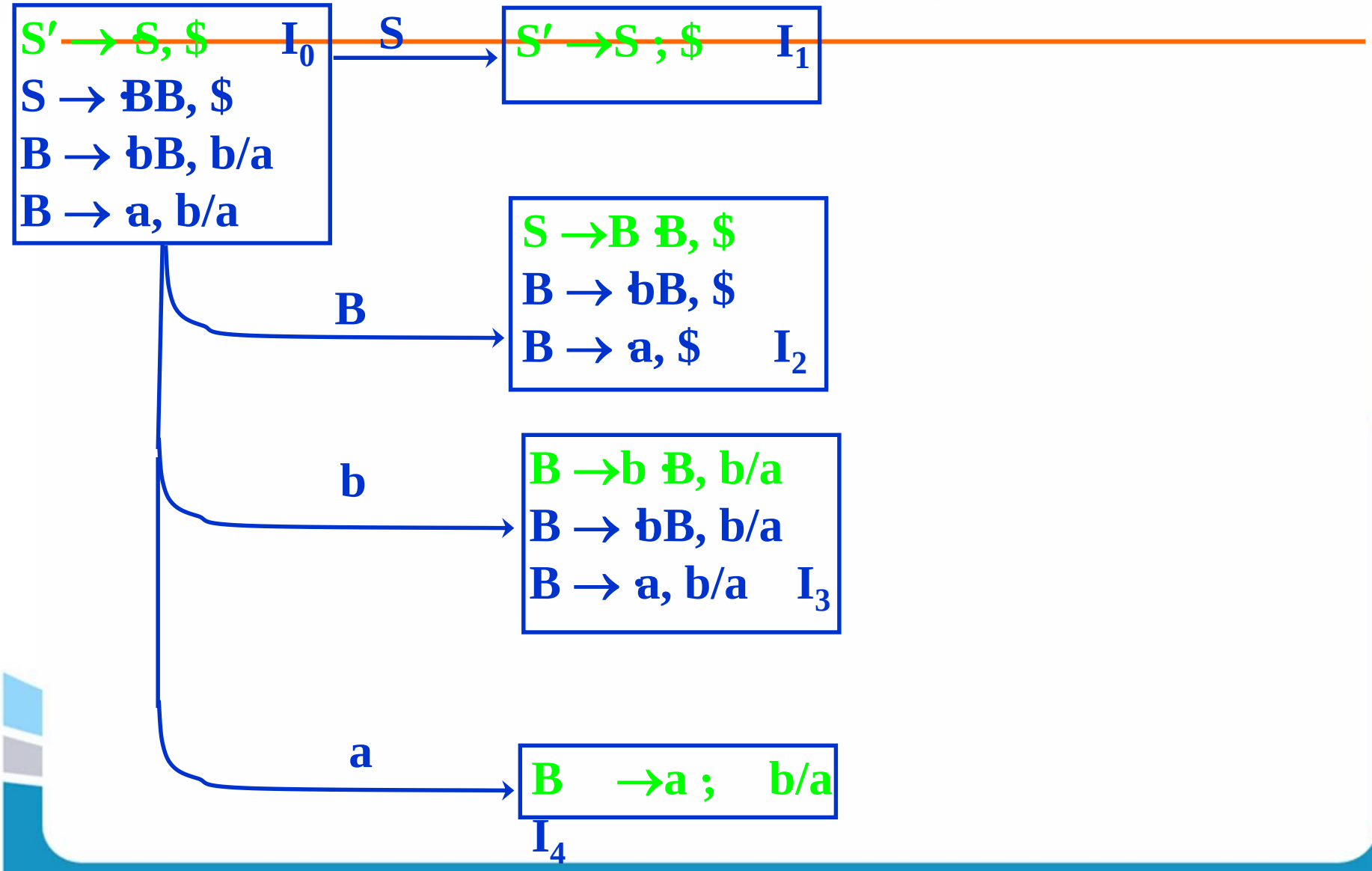
$S' \rightarrow \cdot S, \$$ I_0

$S \rightarrow \cdot BB, \$$

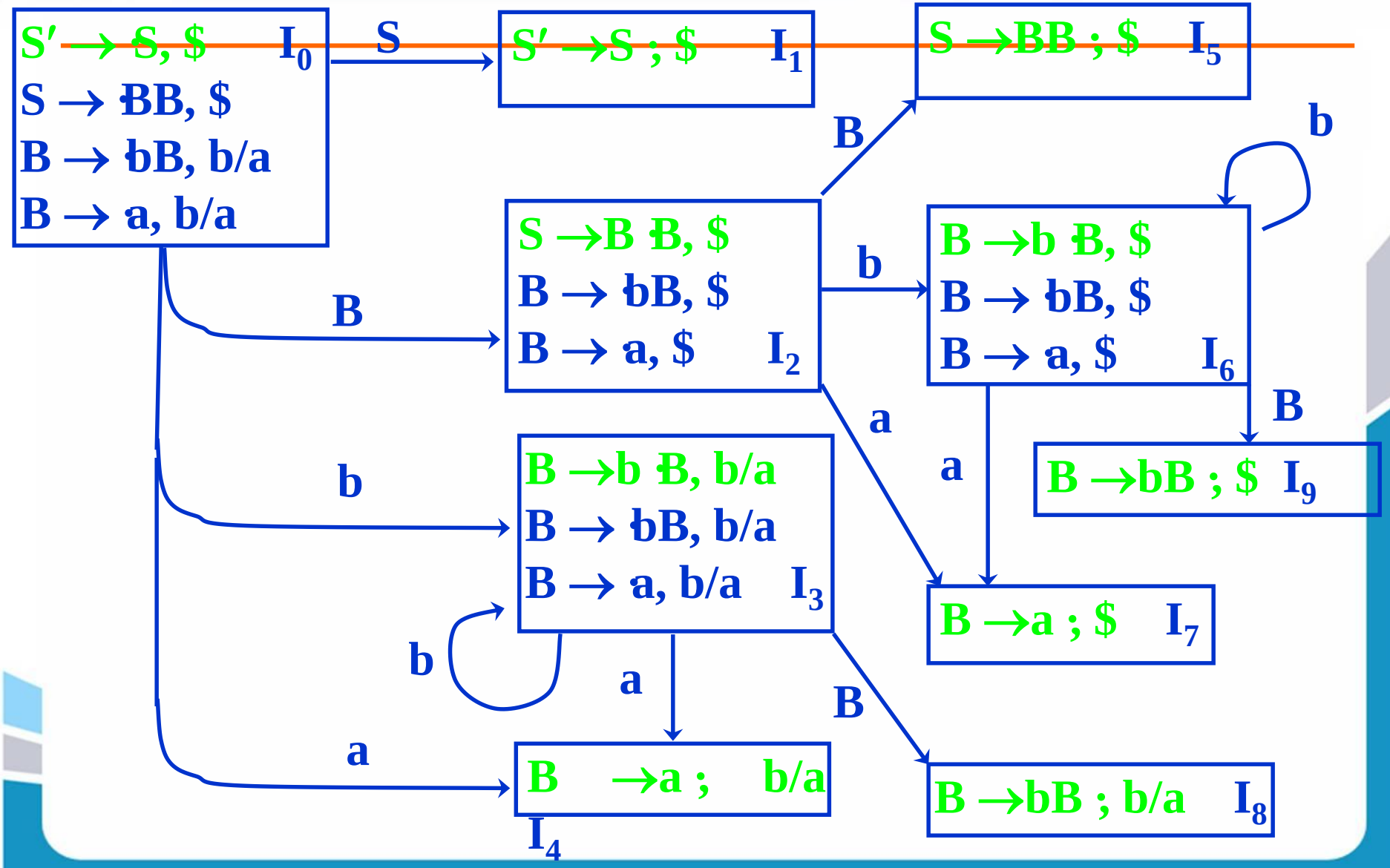
$B \rightarrow \cdot bB, b/a$

$B \rightarrow \cdot a, b/a$

4.5 LR分析器


S′ → ·S, $
S → ·BB, $
B → ·bB, b/a
B → ·a, b/a
I0
S
S′ → S ·, $
I1
B
S → B·B, $
B → ·bB, $
B → ·a, $
I2
b
B → b·B, b/a
B → ·bB, b/a
B → ·a, b/a
I3
a
B → a ·, b/a
I4

4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
b
B
S → B B, $
B → bB, $
B → a, $ I2
b
B → b B, $
B → bB, $
B → a, $ I6
a
a
B
B → bB ; $ I9
b
B → b B, b/a
B → bB, b/a
B → a, b/a I3
B → a ; $ I7
b
a
B
a
B → a ; b/a
I4
B → bB ; b/a I8

4.5 LR分析器

构造规范的LR分析表

(1) 基于LR(1)项目来构造识别G'活前缀的DFA

(2)从I_i构造分析器的状态i, 状态i的*action*函数如下确定

- 如果$[A \rightarrow \alpha \cdot a\beta, b]$在$I_i$中，且$goto(I_i, a) = I_j$，那么置$action[i, a]$为$sj$
- 如果$[A \rightarrow \alpha \cdot, a]$在$I_i$中，且$A \neq S'$，那么置$action[i, a]$为$rj$
- 如果$[S' \rightarrow S \cdot, \$]$在$I_i$中，那么置$action[i, \$] = acc$

如果用上面规则构造出现了冲突，那么文法就不是LR(1)的

4.5 LR分析器

构造规范的LR分析表

(1) 基于LR(1)项目来构造识别*G'*活前缀的DFA

(2)从I_i构造分析器的状态*i*, 状态*i*的*action*函数如下确定

- . . .

(3) 状态*i*的*goto*函数如下确定：

如果$goto(I_i, A) = I_j$, 那么$goto[i, A] = j$

4.5 LR分析器

构造规范的LR分析表

(1) 基于LR(1)项目来构造识别G'活前缀的DFA

(2)从I_i构造分析器的状态i, 状态i的*action*函数如下确定

- . . .

(3) 状态i的*goto*函数如下确定：

. . .

(4) 用上面规则未能定义的所有条目都置为error

4.5 LR分析器

构造规范的LR分析表

(1) 基于LR(1)项目来构造识别*G′*活前缀的DFA

(2)从I_i构造分析器的状态*i*, 状态*i*的*action*函数如下确定

- . . .

(3) 状态*i*的*goto*函数如下确定：

. . .

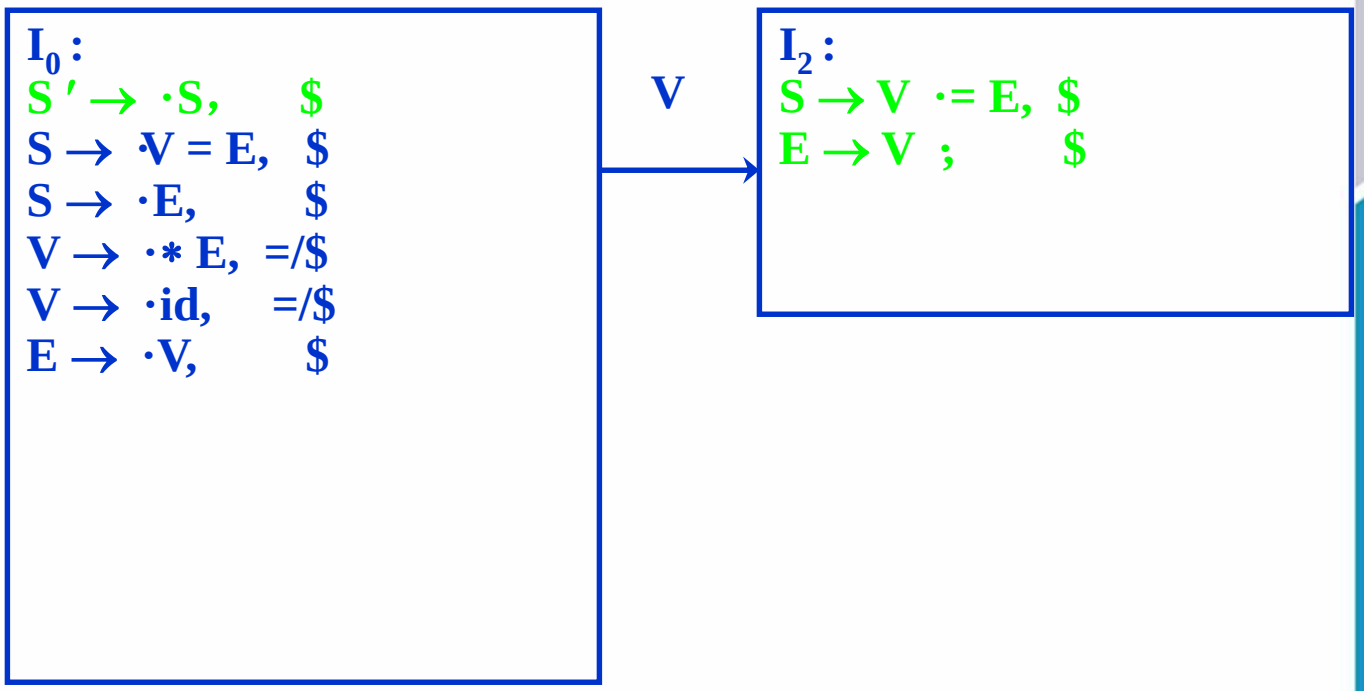
(4) 用上面规则未能定义的所有条目都置为error

(5) 分析器的初始状态是包含[*S′*→·*S*, $]的项目集对应的状态

4.5 LR分析器

- 先前基于SLR(1)有移进–归约冲突的例子，在基于规范LR(1)时无冲突

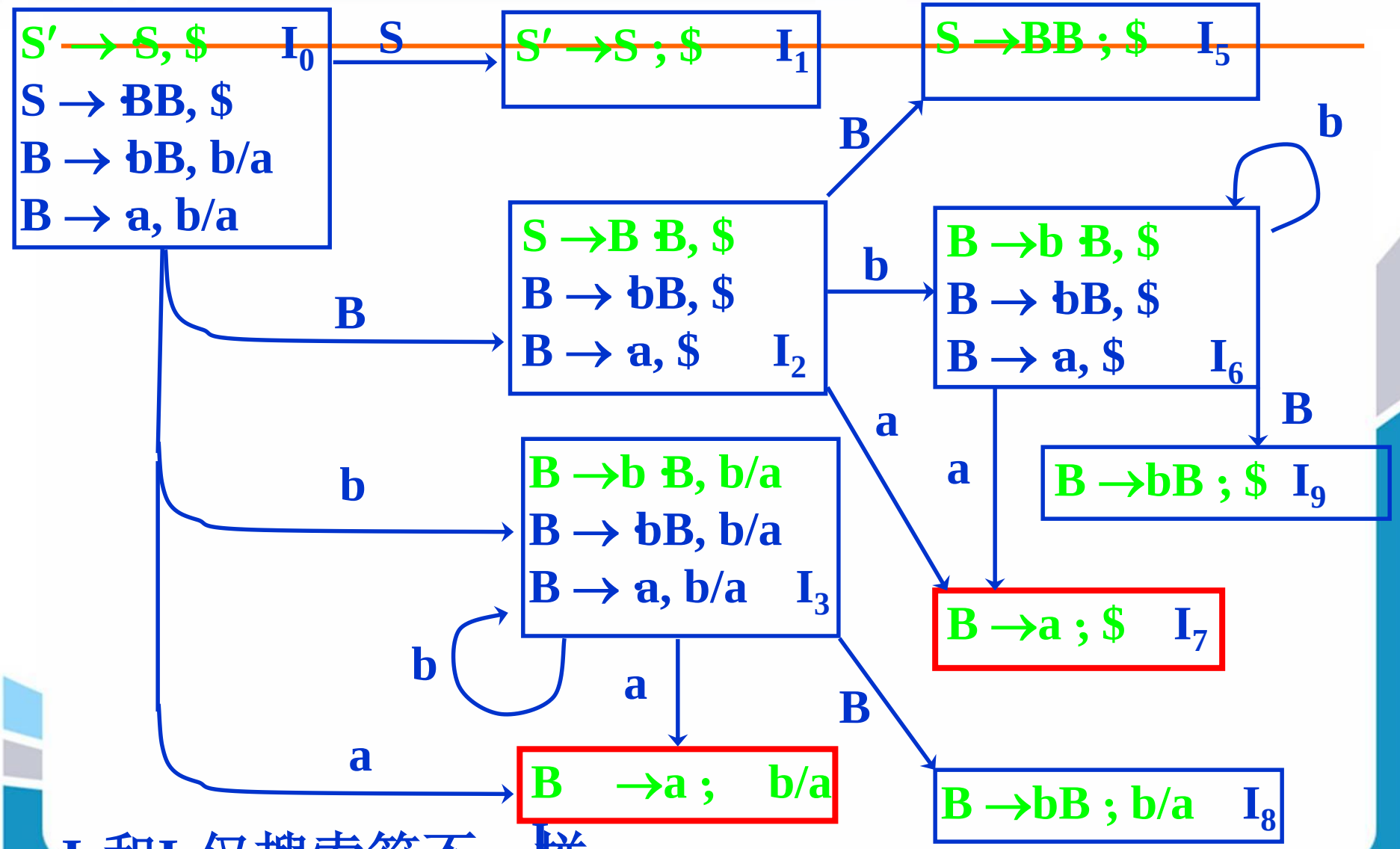
$S \rightarrow V = E$
$S \rightarrow E$
$V \rightarrow * E$
$V \rightarrow id$
$E \rightarrow V$



4.5.5 构造LALR分析表

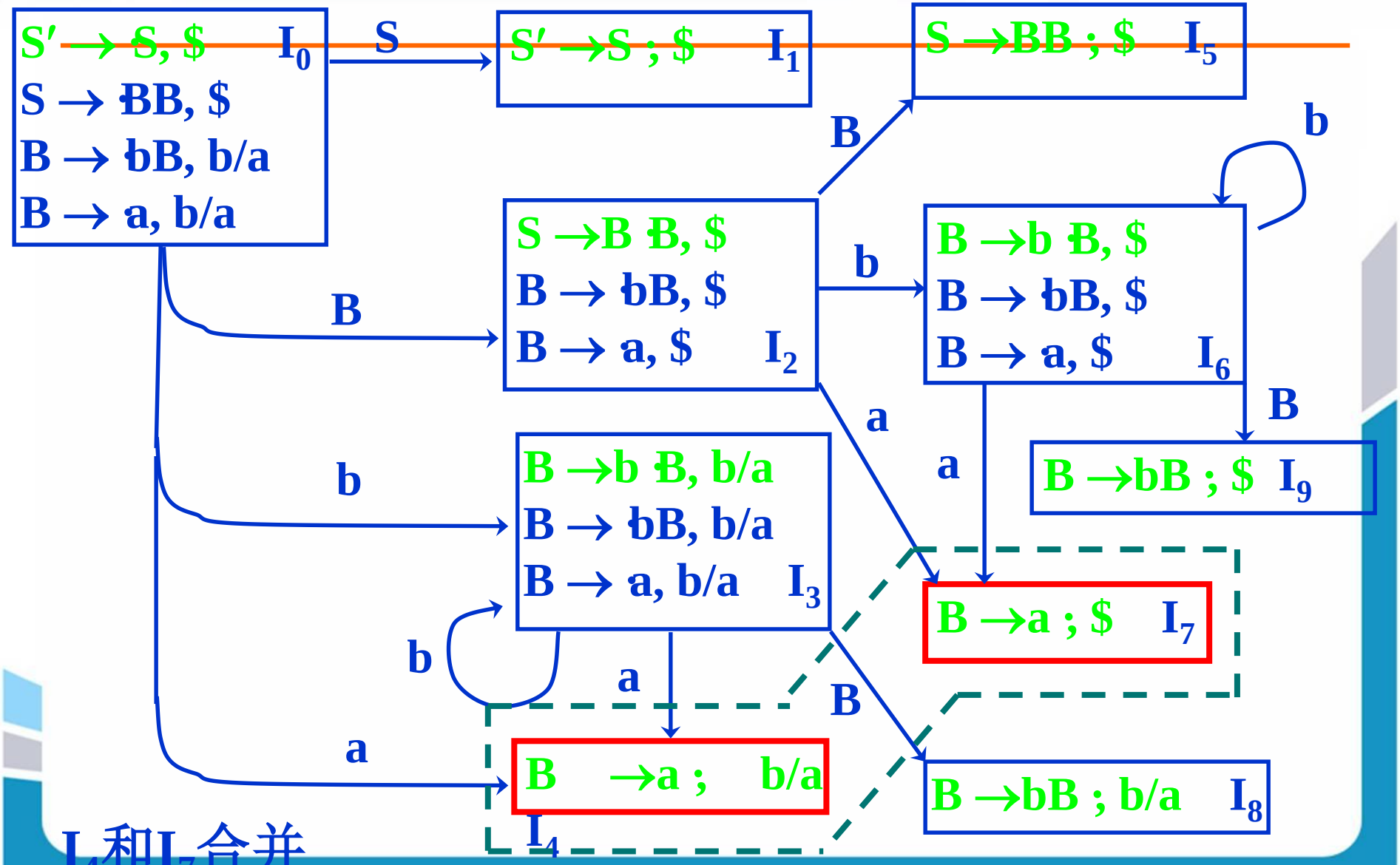
- 研究LALR的原因
  规范LR分析表的状态数偏多
- LALR特点
  - LALR和SLR的分析表有同样多的状态，比规范LR分析表要小得多
  - LALR的能力介于SLR和规范LR之间
  - LALR的能力在很多情况下已经够用
- LALR分析表构造方法
  - 通过合并规范LR(1)项目集来得到

4.5 LR分析器



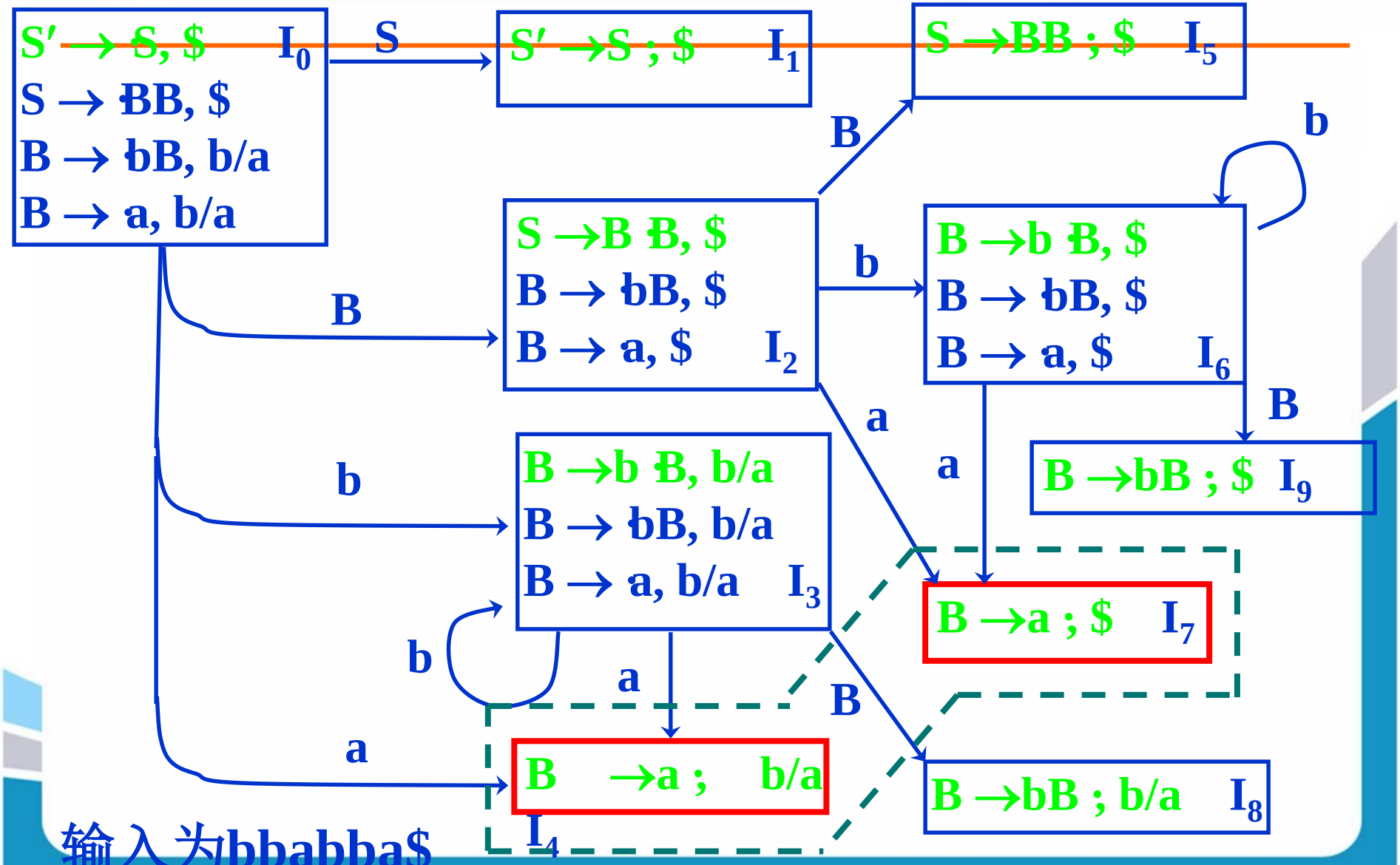
I_4和I_7仅搜索符不一样

4.5 LR分析器



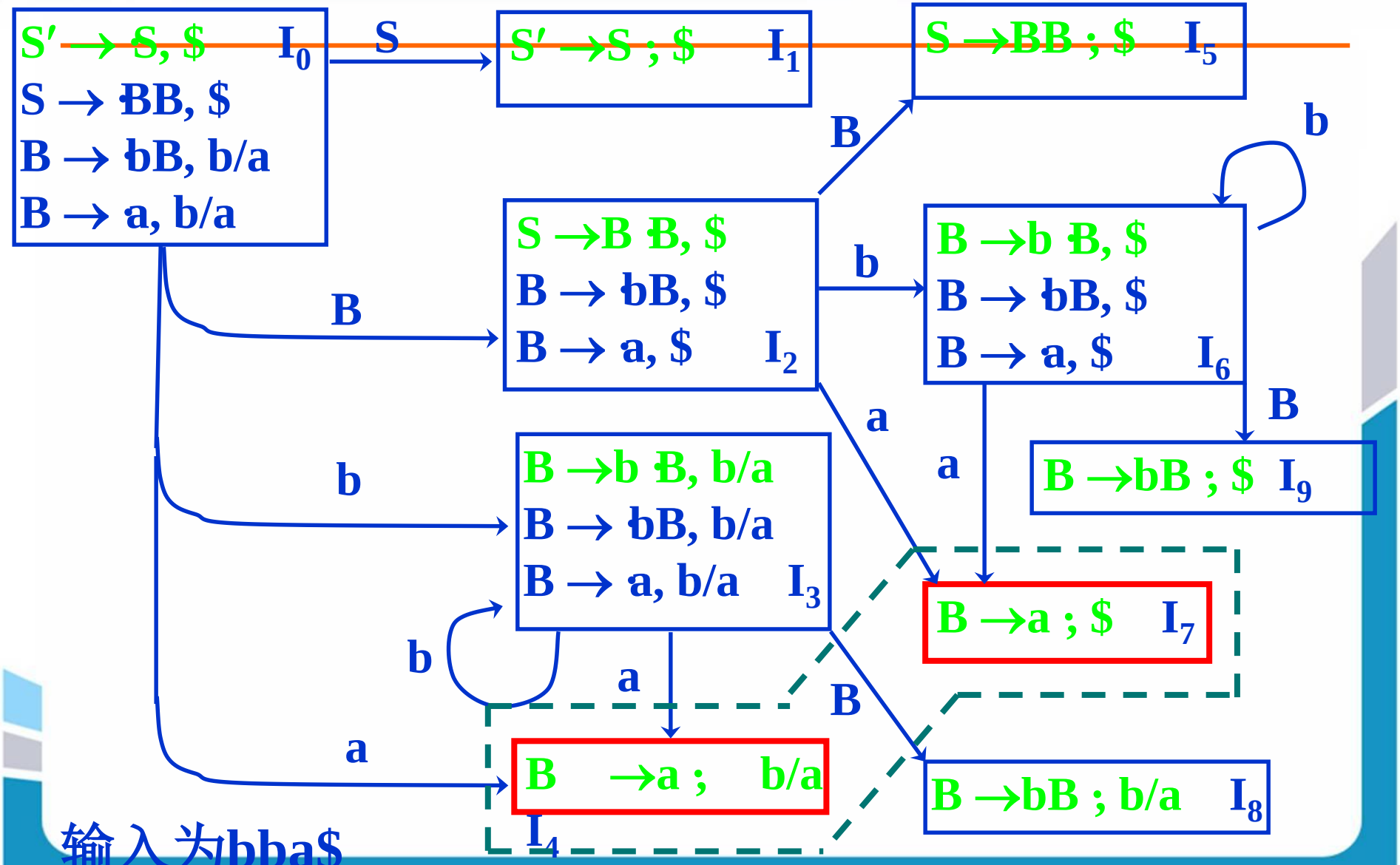
I_4和I_7合并

4.5 LR分析器



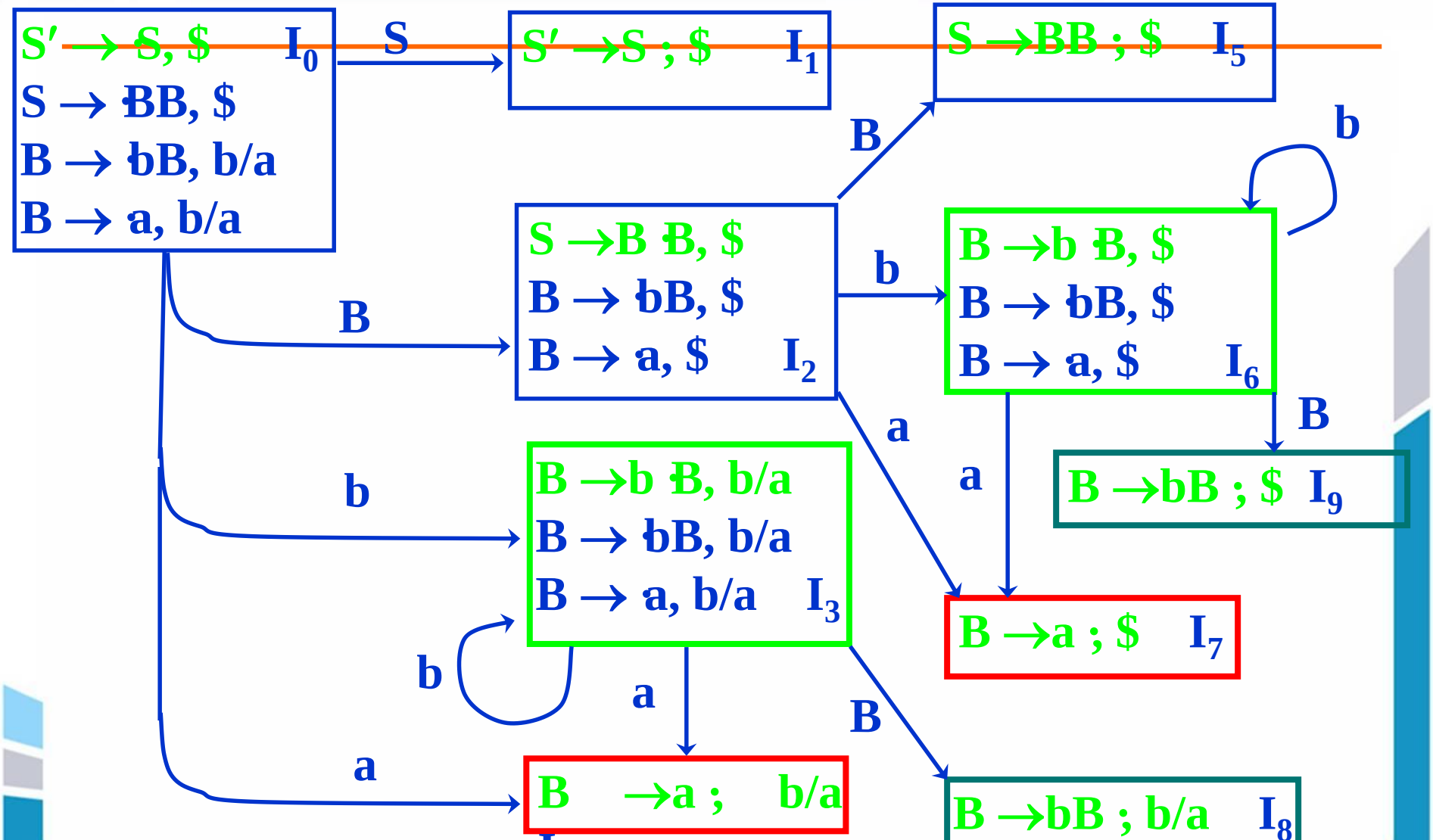
输入为bbabba$

4.5 LR分析器



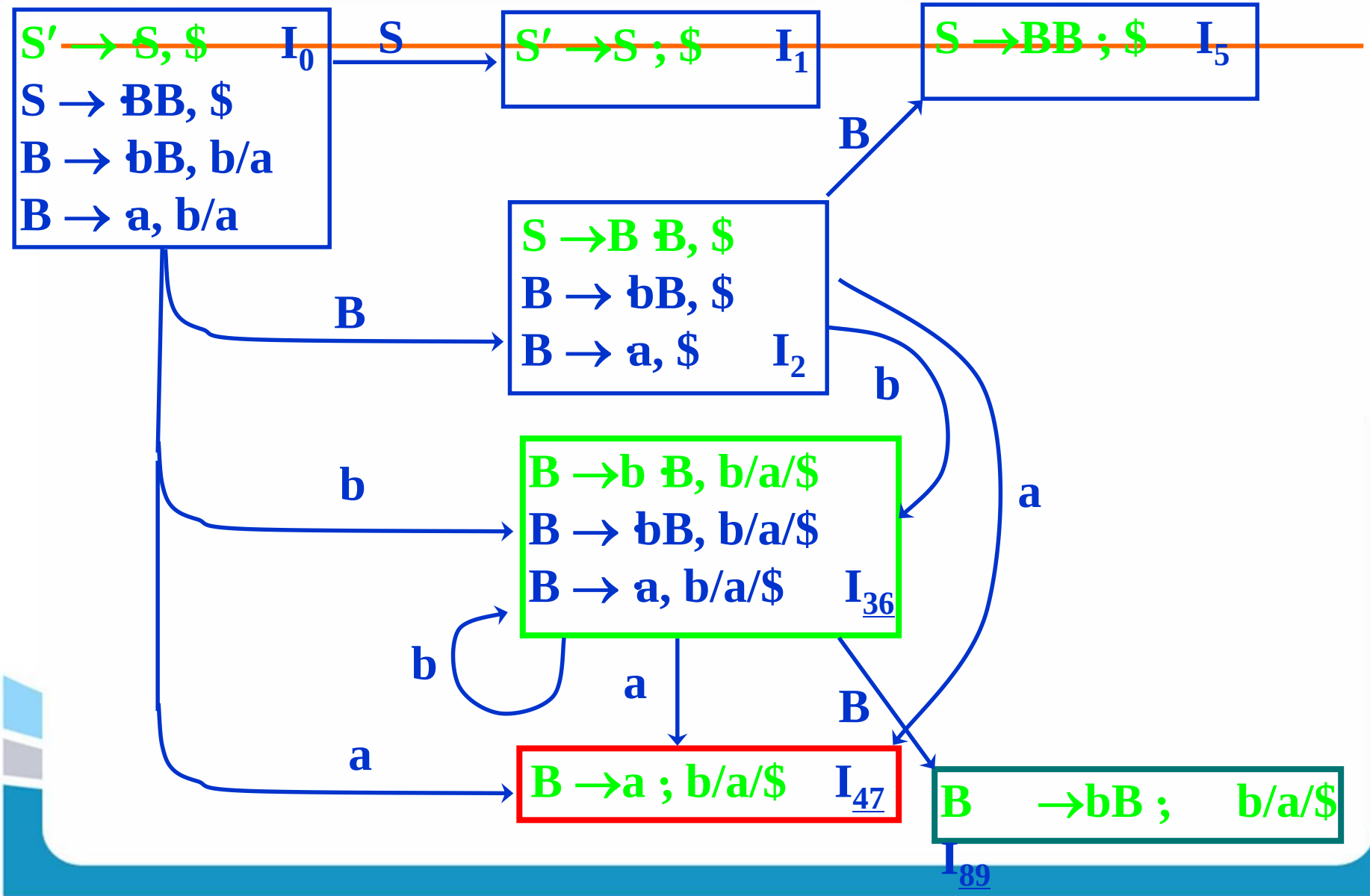
输入为bba$

4.5 LR分析器

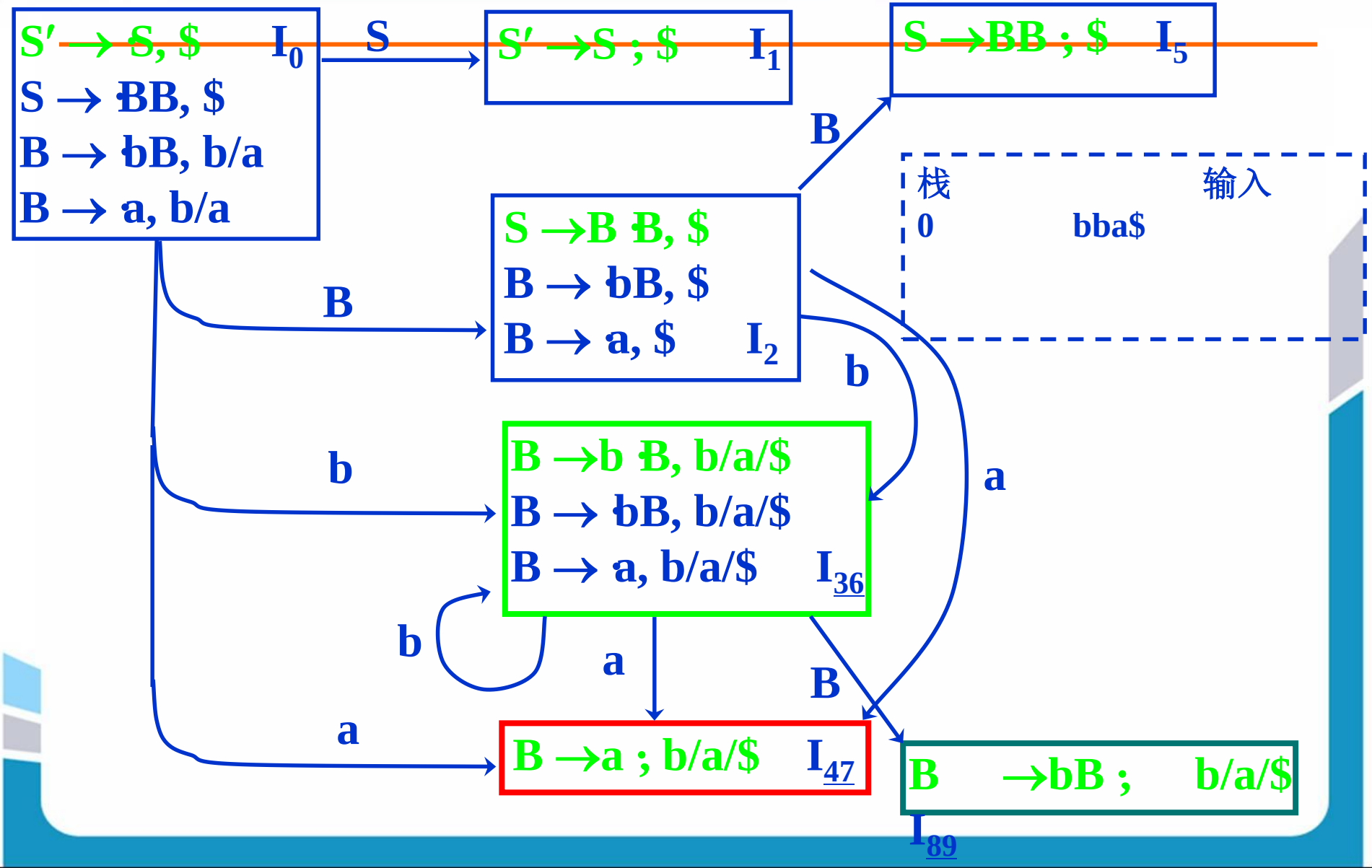


有三组同心集，都合并

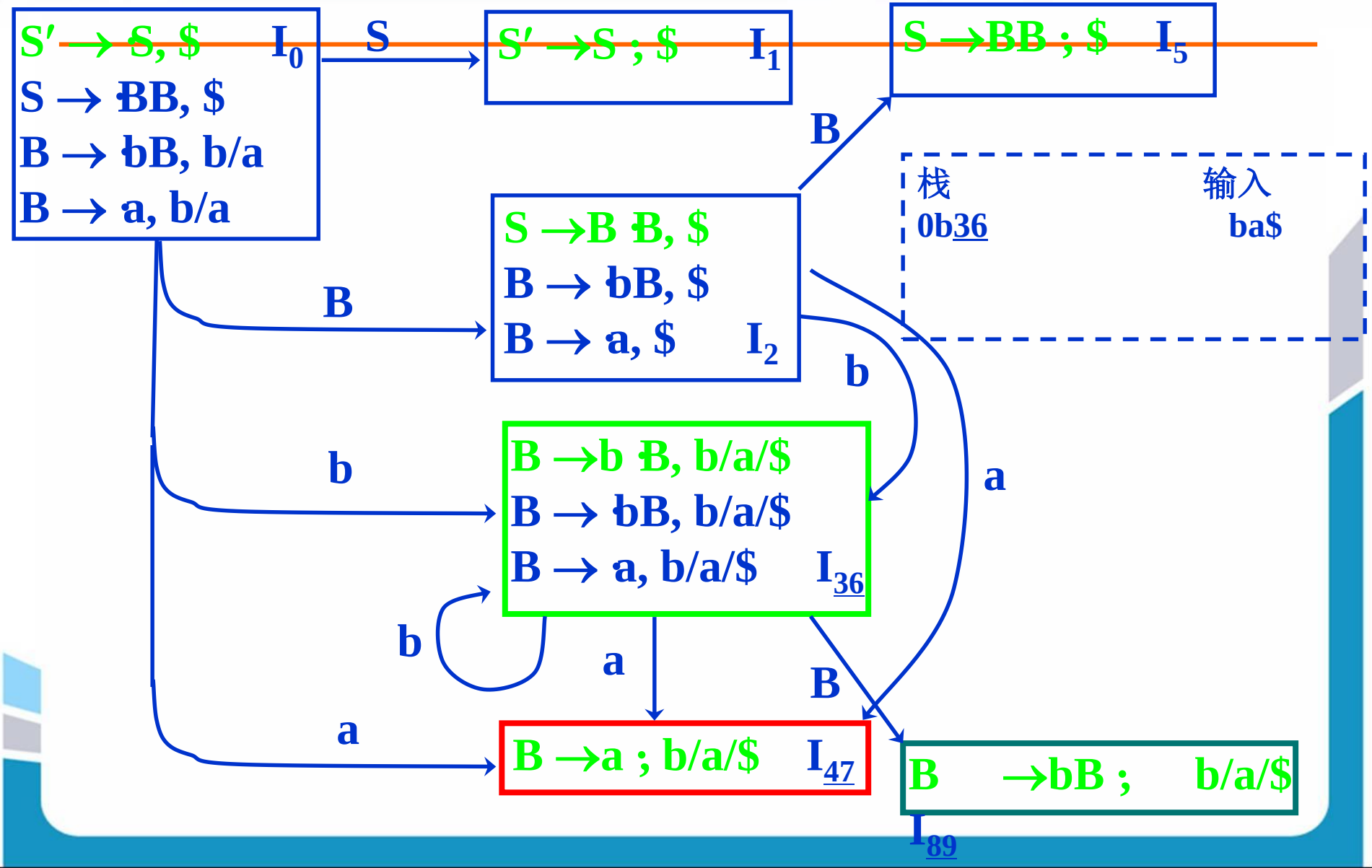
4.5 LR分析器


S′ → S, $
S → BB, $
B → bB, b/a
B → a, b/a
I0
S
S′ → S ; $
I1
S → BB ; $
I5
B
B
S → B B, $
B → bB, $
B → a, $
I2
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$
I36
a
b
a
B
a
B → a ; b/a/$
I47
B → bB ; b/a/$
I89

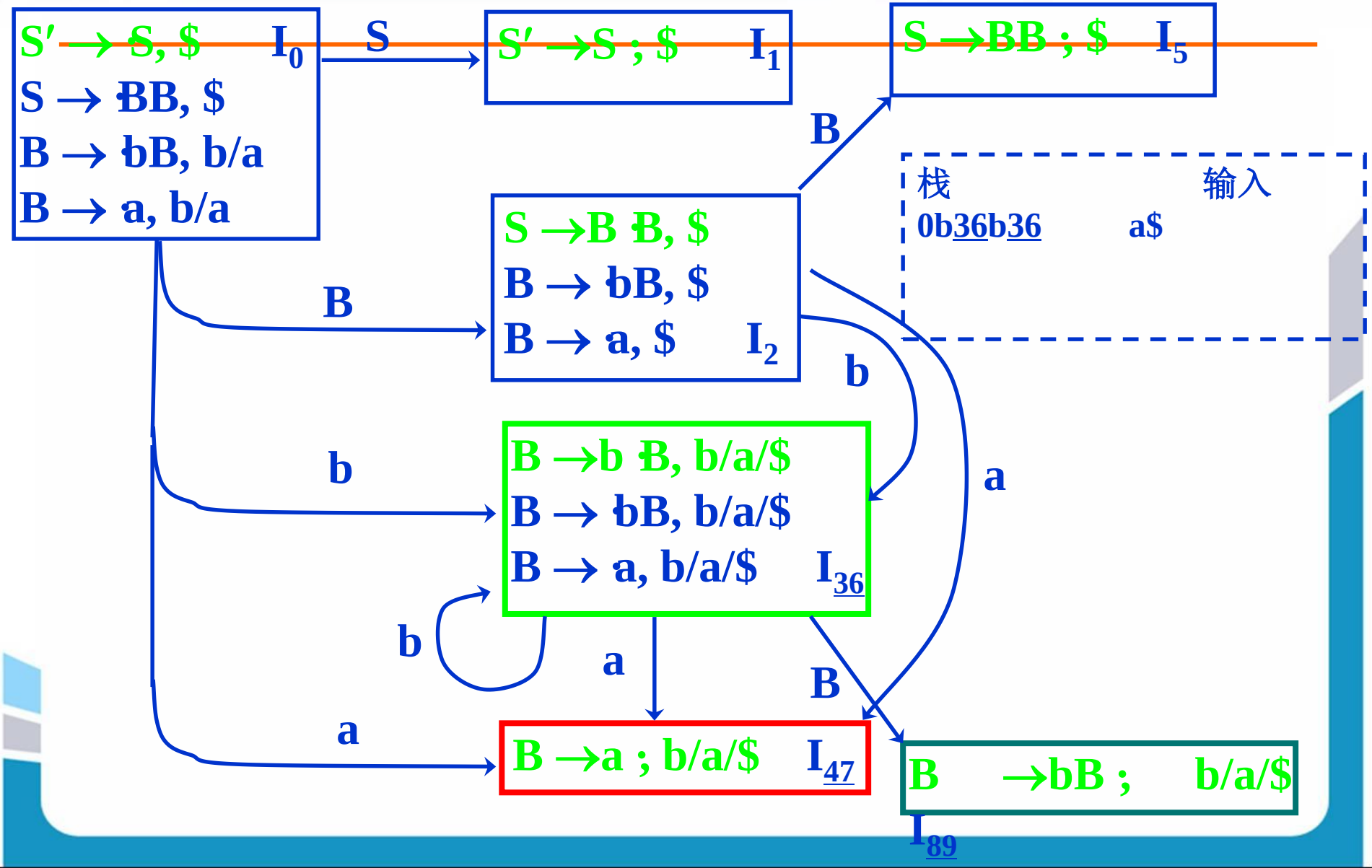
4.5 LR分析器


S′ → ·S, $ I0
S → ·BB, $
B → ·bB, b/a
B → ·a, b/a
S
S′ →S ·; $ I1
S →BB ·; $ I5
B
B
S →B ·B, $
B → ·bB, $
B → ·a, $ I2
b
b
B →b ·B, b/a/$
B → ·bB, b/a/$
B → ·a, b/a/$ I36
a
b
a
B
a
B →a ·; b/a/$ I47
B →bB ·; b/a/$
I89
栈 输入
0 bba$

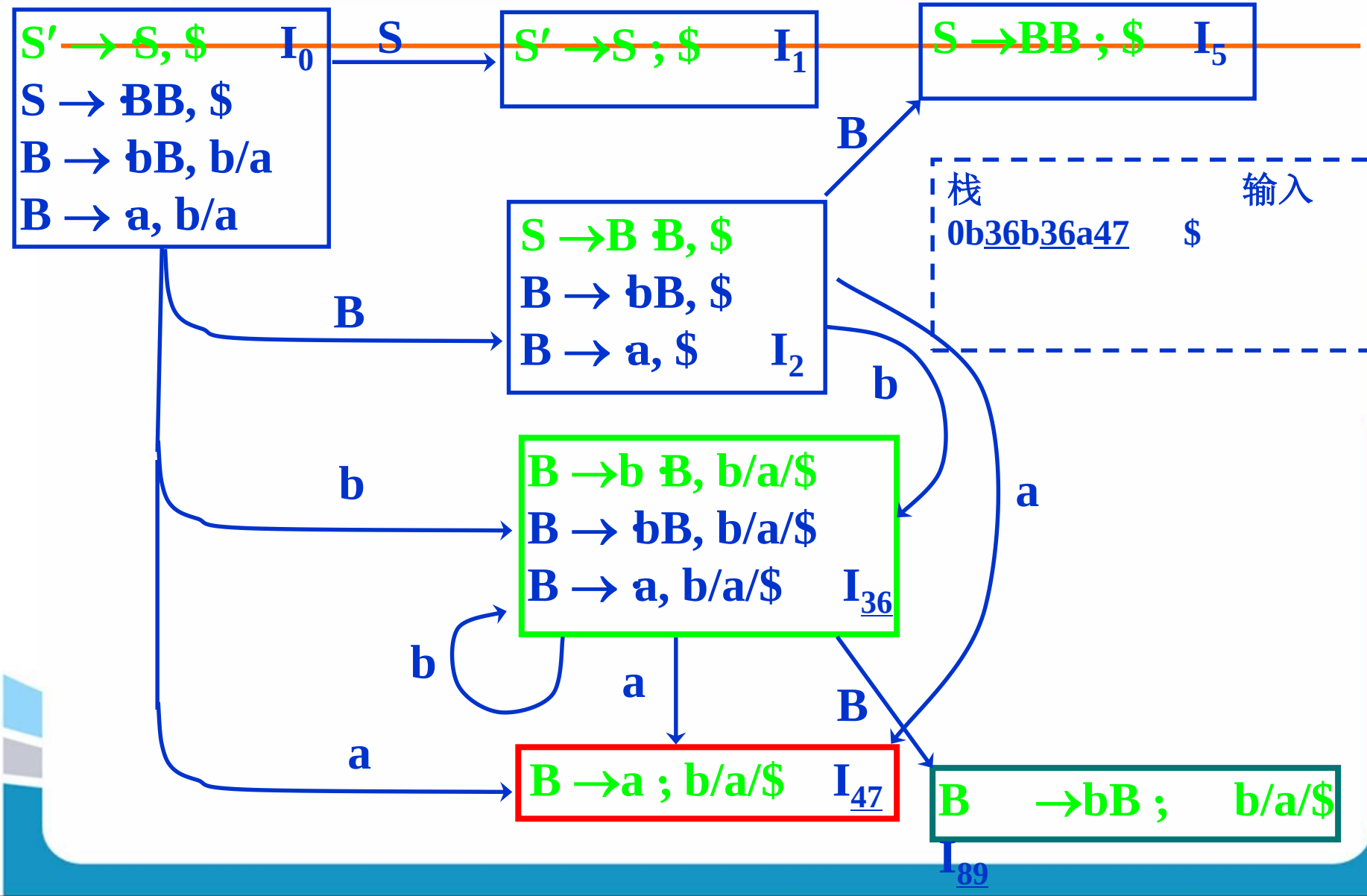
4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
B
S → B B, $
B → bB, $
B → a, $ I2
栈
0b36
输入
ba$
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$ I36
a
b
a
B
a
B → a ; b/a/$ I47
B → bB ; b/a/$
I89

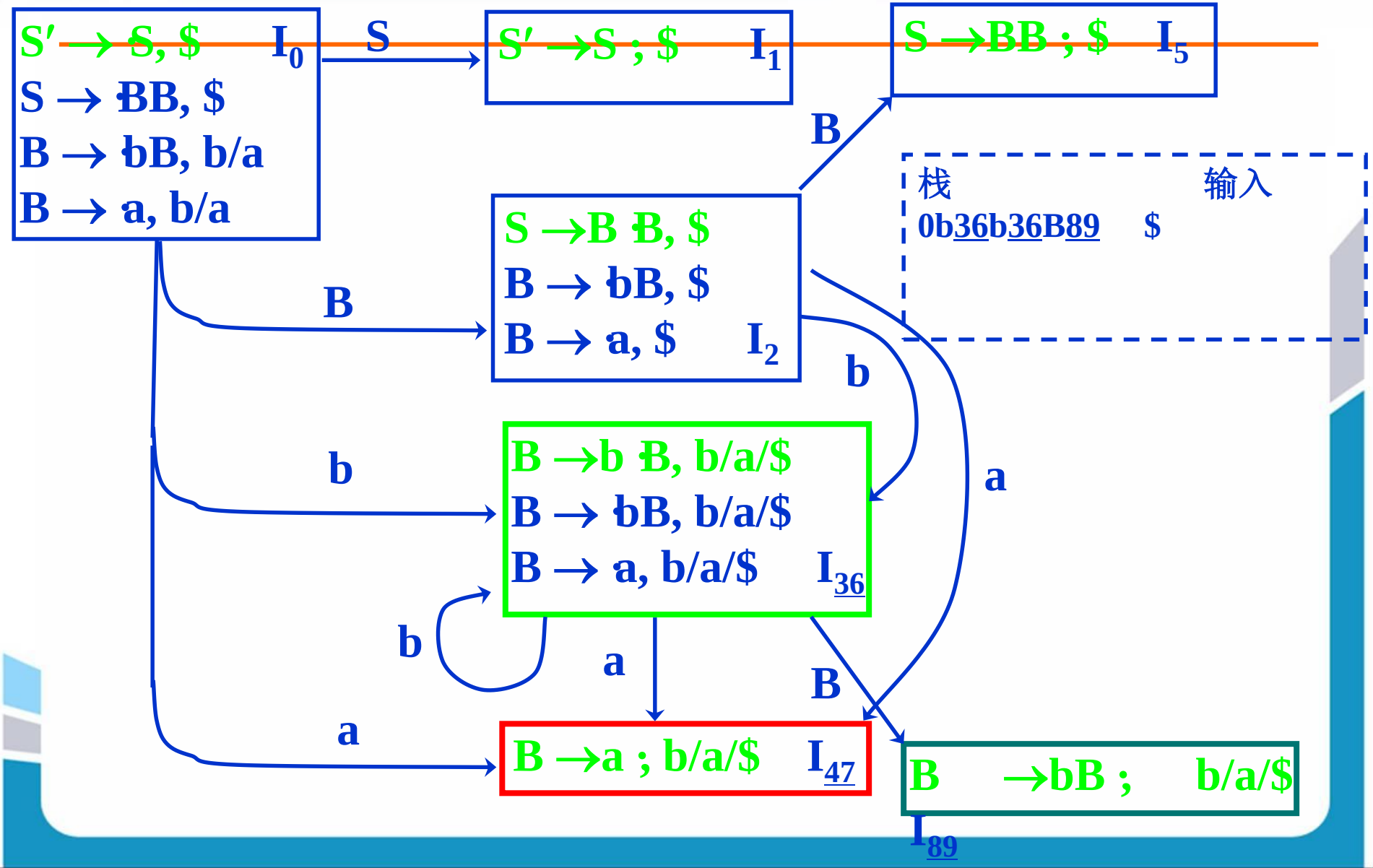
4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
B
S → B B, $
B → bB, $
B → a, $ I2
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$ I36
a
b
a
B
a
B → a ; b/a/$ I47
B → bB ; b/a/$
I89
栈 输入
0b36b36 a$

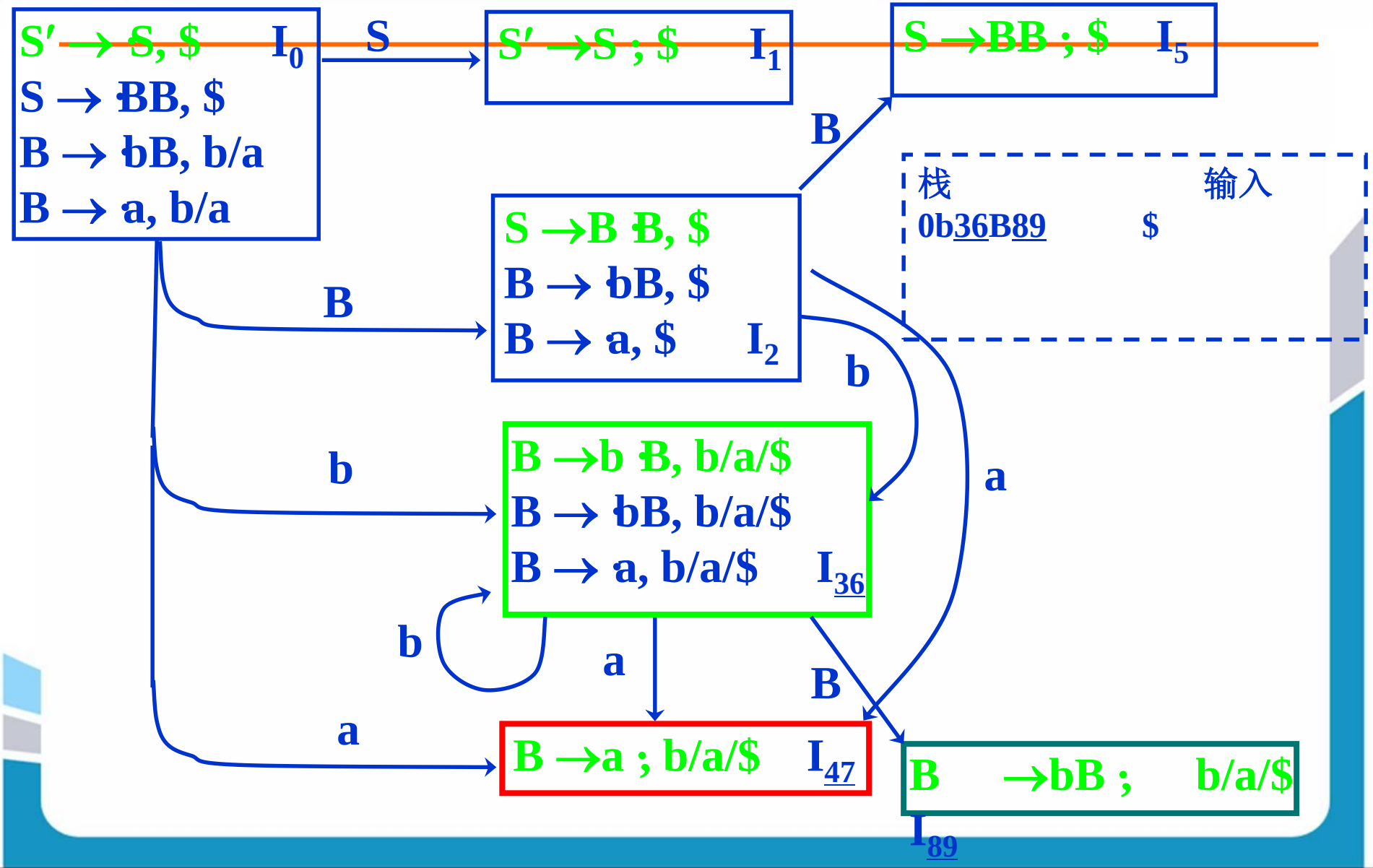
4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
B
S → B B, $
B → bB, $
B → a, $ I2
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$ I36
a
b
a
B
a
B → a ; b/a/$ I47
B → bB ; b/a/$
I89
栈 输入
0b36b36a47 $

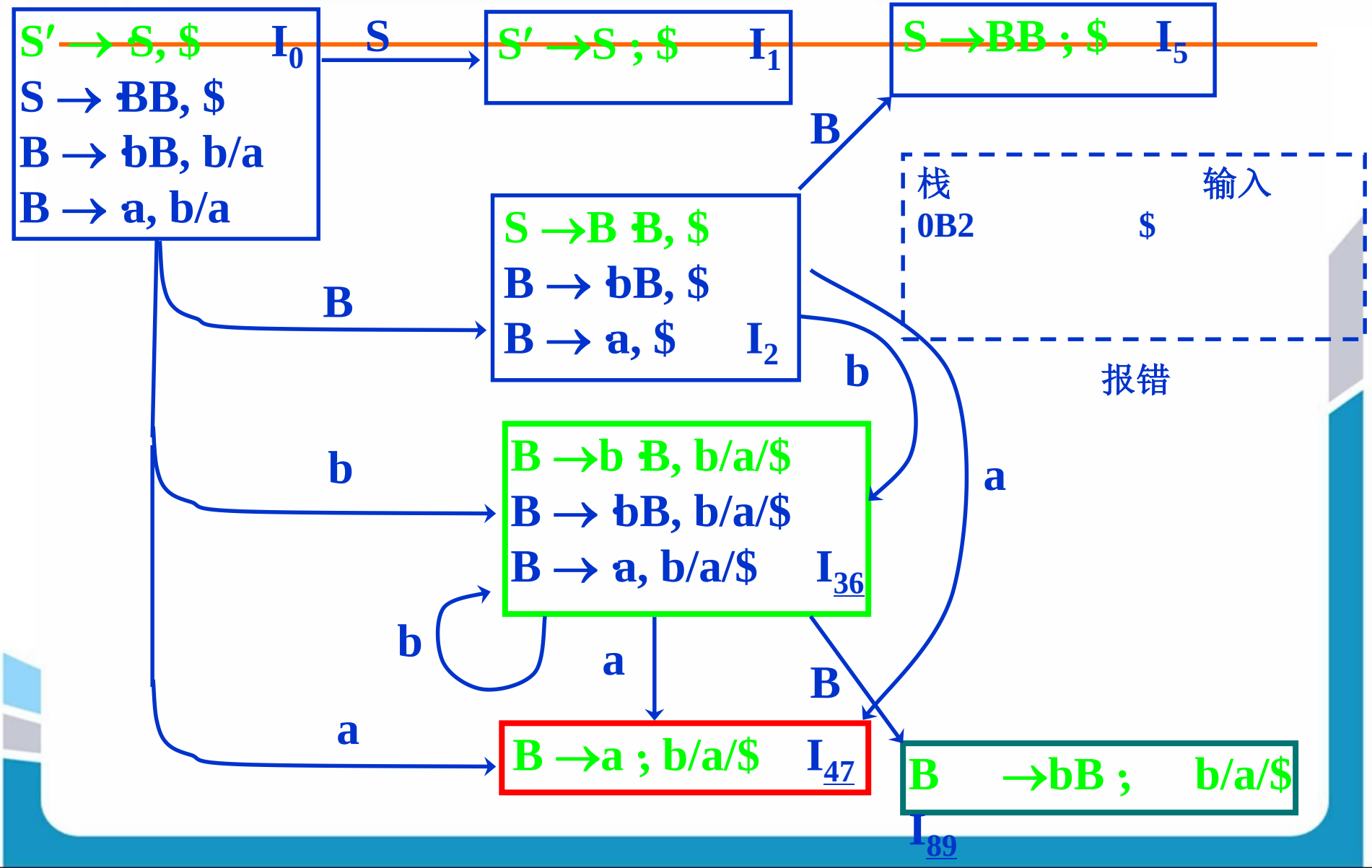
4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
B
S → B B, $
B → bB, $
B → a, $ I2
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$ I36
a
b
a
B
a
B → a ; b/a/$ I47
B → bB ; b/a/$
I89
栈 输入
0b36b36B89 $

4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
B
S → B B, $
B → bB, $
B → a, $ I2
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$ I36
a
b
a
B
a
B → a ; b/a/$ I47
B → bB ; b/a/$
I89
栈 输入
0b36B89 $

4.5 LR分析器


S′ → S, $ I0
S → BB, $
B → bB, b/a
B → a, b/a
S
S′ → S ; $ I1
S → BB ; $ I5
B
B
S → B B, $
B → bB, $
B → a, $ I2
b
b
B → b B, b/a/$
B → bB, b/a/$
B → a, b/a/$ I36
a
b
a
B
a
B → a ; b/a/$ I47
B → bB ; b/a/$
I89
栈 输入
0B2 $
报错

4.5 LR分析器

1、合并同心项目集可能会引起冲突

同心的LR(1)项目集：

略去搜索符后它们是相同的集合

4.5 LR分析器

1、合并同心项目集可能会引起冲突

- 同心集的合并不会引起新的移进–归约冲突

项目集1

$[A \rightarrow \alpha \cdot, a]$

. . .

项目集2

$[B \rightarrow \beta \cdot a\gamma, b]$

. . .

若合并后有冲突

4.5 LR分析器

1、合并同心项目集可能会引起冲突

- 同心集的合并不会引起新的移进–归约冲突

项目集1	项目集2
$[A\rightarrow\alpha\cdot, a]$	$[B\rightarrow\beta\cdot a\gamma, b]$
$[B\rightarrow\beta\cdot a\gamma, c]$	$[A\rightarrow\alpha\cdot, d]$
...	...

则合并前就有冲突

4.5 LR分析器

1、合并同心项目集可能会引起冲突

- 同心集的合并不会引起新的移进–归约冲突
- 同心集的合并有可能产生新的归约–归约冲突

$S' \rightarrow S$

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

$A \rightarrow c$

$B \rightarrow c$

对ac有效的项目集

$A \rightarrow c \ ; d$

$B \rightarrow c \ ; e$

对bc有效的项目集

$A \rightarrow c \ ; e$

$B \rightarrow c \ ; d$

合并同心集后

$A \rightarrow c \ ; d/e$

$B \rightarrow c \ ; d/e$

该文法是LR(1)的，但不是LALR(1)的

4.5 LR分析器

2、构造LALR(1)分析表

(1) 构造LR(1)项目集规范族$C = \{I_0, I_1, \ldots, I_n\}$

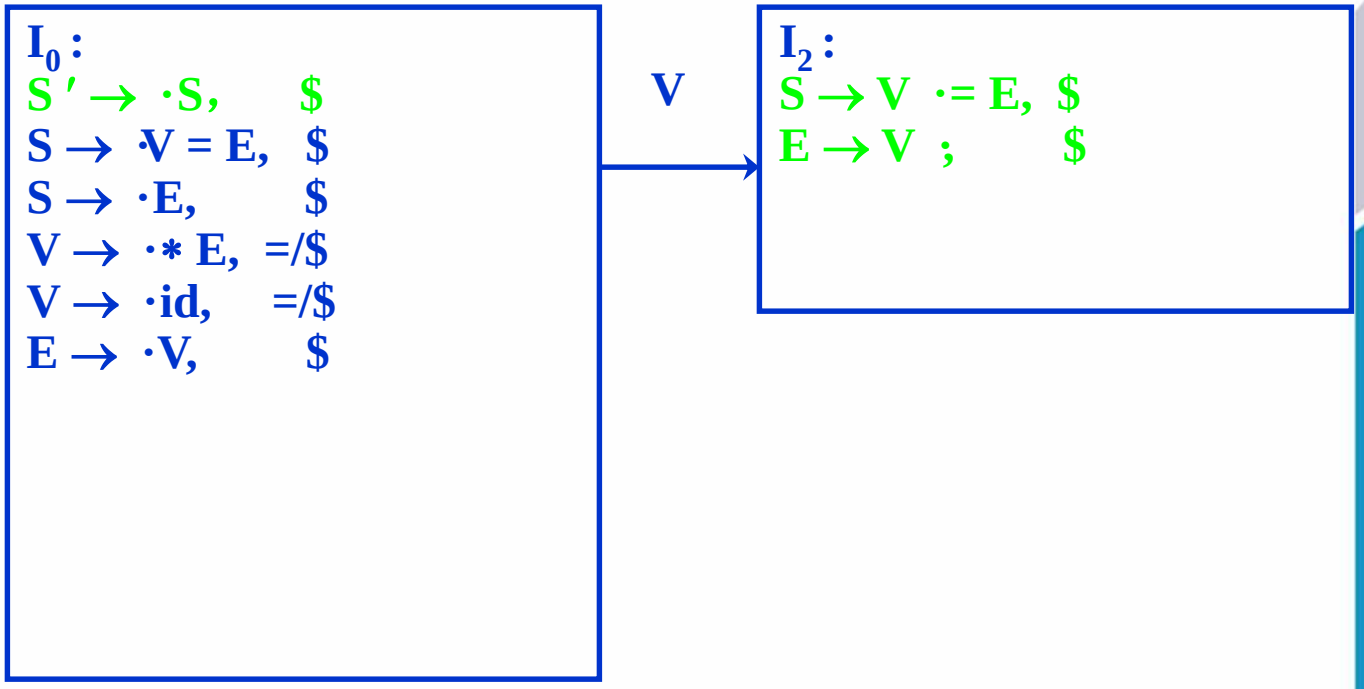
(2) 寻找LR(1)项目集规范族中同心的项目集，用它们的并集代替它们

(3) 按构造规范LR(1)分析表的方式构造分析表

4.5 LR分析器

- 下面文法是LALR(1)的，因无同心集可合并
- 但不是SLR(1)的

$S \rightarrow V = E$
$S \rightarrow E$
$V \rightarrow * E$
$V \rightarrow id$
$E \rightarrow V$



4.5 LR分析器

4.5.6 非LR的上下文无关结构

若自左向右扫描的移进–归约分析器能及时识别出现在栈顶的句柄，那么相应的文法就是LR的

语言$L = \{ww^R \mid w \in (a \mid b)^*\}$的文法

$S \rightarrow aSa \mid bSb \mid \varepsilon$

不是LR的　　*ababbbbaba*

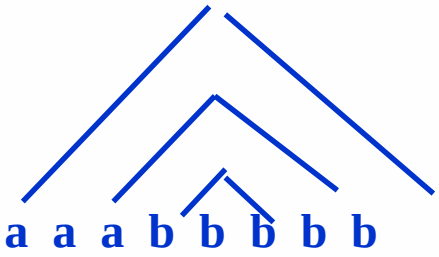
语言$L = \{wcw^R \mid w \in (a \mid b)^*\}$的文法

$S \rightarrow aSa \mid bSb \mid c$

是LR的　　*ababbcbbaba*

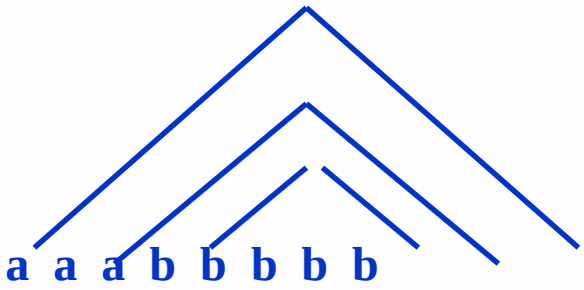
4.5 LR分析器

- 例 为语言$L = \{ a^m b^n \mid n > m \geq 0 \}$写三个文法, 它们分别是LR(1)的、二义的和非二义且非LR(1)的
- LR(1)文法：$S \rightarrow AB$　　$A \rightarrow aAb \mid \varepsilon$　　$B \rightarrow Bb \mid b$



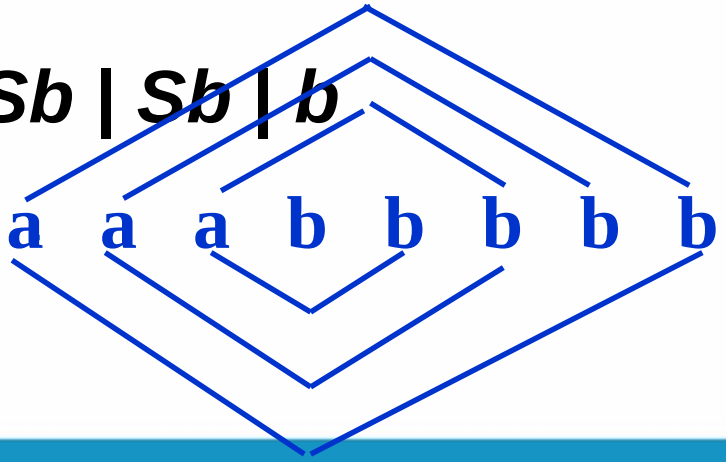
4.5 LR分析器

- 例 为语言$L = \{ a^m b^n \mid n > m \geq 0 \}$写三个文法, 它们分别是LR(1)的、二义的和非二义且非LR(1)的
- LR(1)文法：$S \rightarrow AB$　　$A \rightarrow aAb \mid \varepsilon$　　$B \rightarrow Bb \mid b$
- 非二义且非LR(1)的文法：$S \rightarrow aSb \mid B$　　$B \rightarrow Bb \mid b$



4.5 LR分析器

- 例 为语言$L = \{ a^m b^n \mid n > m \geq 0 \}$写三个文法, 它们分别是LR(1)的、二义的和非二义且非LR(1)的
- LR(1)文法：$S \rightarrow AB \quad A \rightarrow aAb \mid \varepsilon \quad B \rightarrow Bb \mid b$
- 非二义且非LR(1)的文法：$S \rightarrow aSb \mid B \quad B \rightarrow Bb \mid b$
- 二义的文法：$S \rightarrow aSb \mid Sb \mid b$



4.6 二义文法的应用

二义文法的特点：

- 二义文法决不是LR文法
- 简洁、自然
- 例 二义文法 $E \rightarrow E + E \mid E * E \mid (E) \mid \mathrm{id}$

  非二义的文法：

  $E \rightarrow E + T \mid T$

  $T \rightarrow T * F \mid F$

  $F \rightarrow (E) \mid \mathrm{id}$

  该文法有单个非终结符为右部的产生式

4.6 二义文法的应用

二义文法的特点：

- 二义文法决不是LR文法
- 简洁、自然
- 可以用文法以外的信息来消除二义
- 语法分析的效率高（基于消除二义后得到的分析表）

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \rightarrow E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_7

$E \to E + E\cdot$

$E \to E\cdot + E$

$E \to E\cdot * E$

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_7

$E \to E + E \cdot$

$E \to E \cdot + E$

$E \to E \cdot * E$

id + id + id

面临+，归约

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_7

$E \to E + E\cdot$

$E \to E\cdot + E$ id + id + id

$E \to E\cdot * E$ id + id * id

面临+，归约 面临*，移进

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \text{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_7

$E \to E + E\cdot$

$E \to E\cdot + E$　　id + id　　+ id

$E \to E\cdot * E$　　id + id　　* id

面临+，归约　　面临*，移进

面临) 和$，归约

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_8

$E \to E * E \cdot$

$E \to E \cdot + E$

$E \to E \cdot * E$

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_8

$E \to E * E \cdot$

$E \to E \cdot + E$

$E \to E \cdot * E$

id * id + id

面临+，归约

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_8

$E \to E * E \cdot$

$E \to E \cdot + E$ 　　id * id 　　+ id

$E \to E \cdot * E$ 　　id * id 　　* id

面临+，归约 　　面临*，归约

4.6 二义文法的应用

4.6.1 使用文法以外信息来解决分析动作冲突

- 例 二义文法 $E \to E + E \mid E * E \mid (E) \mid \mathrm{id}$

  规定： *优先级高于+，两者都是左结合

LR(0)项目集I_8

$E \to E * E \cdot$

$E \to E \cdot + E$ 　　id * id 　　+ id

$E \to E \cdot * E$ 　　id * id 　　* id

面临+，归约 　　面临*，归约

面临) 和$，归约

4.6 二义文法的应用

4.6.2 特殊情况产生式引起的二义性

$E \rightarrow E$ sub E sup E

$E \rightarrow E$ sub E

$E \rightarrow E$ sup E

$E \rightarrow \{E\}$

$E \rightarrow c$

4.6 二义文法的应用

4.6.2 特殊情况产生式引起的二义性

$E \rightarrow E$ sub E sup E

$E \rightarrow E$ sub E

$E \rightarrow E$ sup E

$E \rightarrow \{E\}$

$E \rightarrow c$

从定义形式语言的角度说，第一个产生式是多余的

4.6 二义文法的应用

4.6.2 特殊情况产生式引起的二义性

$E \rightarrow E$ sub E sup E

$E \rightarrow E$ sub E

$E \rightarrow E$ sup E

$E \rightarrow \{E\}$

$E \rightarrow c$

联系到语义处理，第一个产生式是必要的

4.6 二义文法的应用

4.6.2 特殊情况产生式引起的二义性

$E \rightarrow E$ sub E sup E

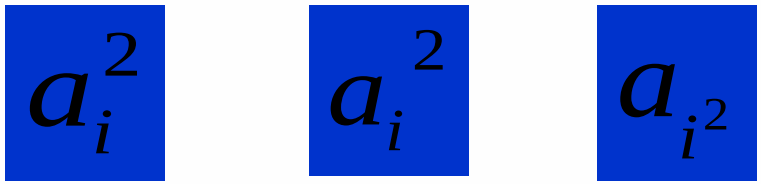
$E \rightarrow E$ sub E

$E \rightarrow E$ sup E

$E \rightarrow \{E\}$

$E \rightarrow c$

对*a* sub *i* sup 2，需要下面第一种输出



4.6 二义文法的应用

4.6.2 特殊情况产生式引起的二义性

$E \rightarrow E$ sub E sup E

$E \rightarrow E$ sub E

$E \rightarrow E$ sup E

$E \rightarrow \{E\}$

$E \rightarrow c$

I_{11}:

$E \rightarrow E$ sub E sup

$E \rightarrow E$ sub E sup $E\cdot$

. . .

按前面一个产生式归约

4.6 二义文法的应用

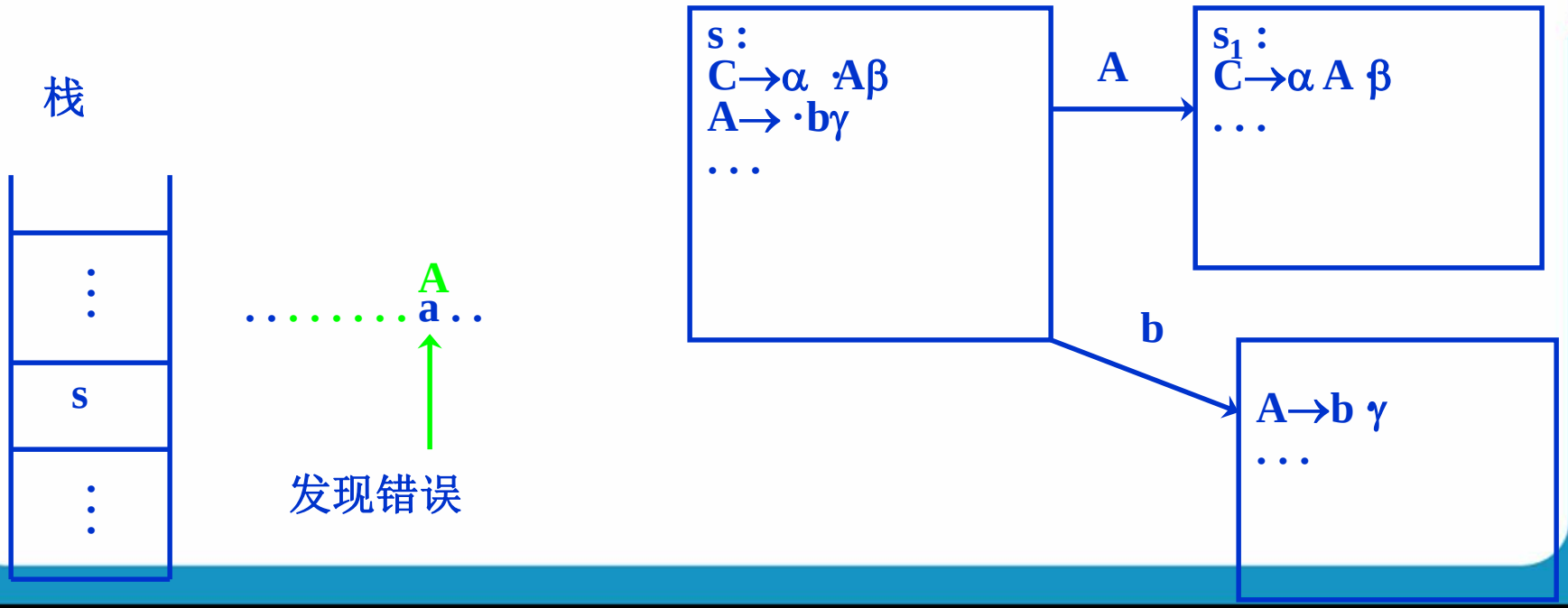
4.6.3 LR分析的错误恢复

1、LR分析器在什么情况下发现错误

- 访问动作表时若遇到出错条目
- 访问转移表时它决不会遇到出错条目
- 决不会把不正确的后继移进栈
- 规范的LR分析器甚至在报告错误之前决不做任何无效归约

4.6 二义文法的应用

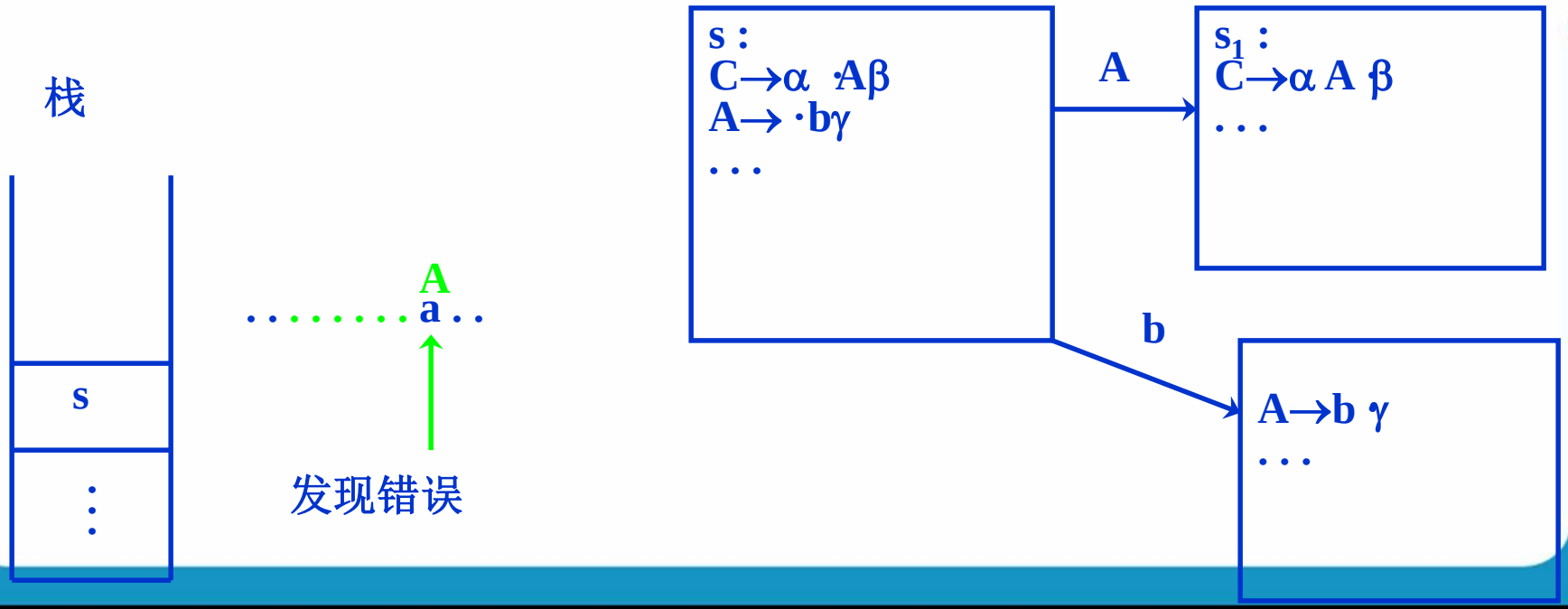
2、紧急方式错误恢复



4.6 二义文法的应用

2、紧急方式错误恢复

(1) 退栈，直至出现状态s，它有预先确定的*A*的转移

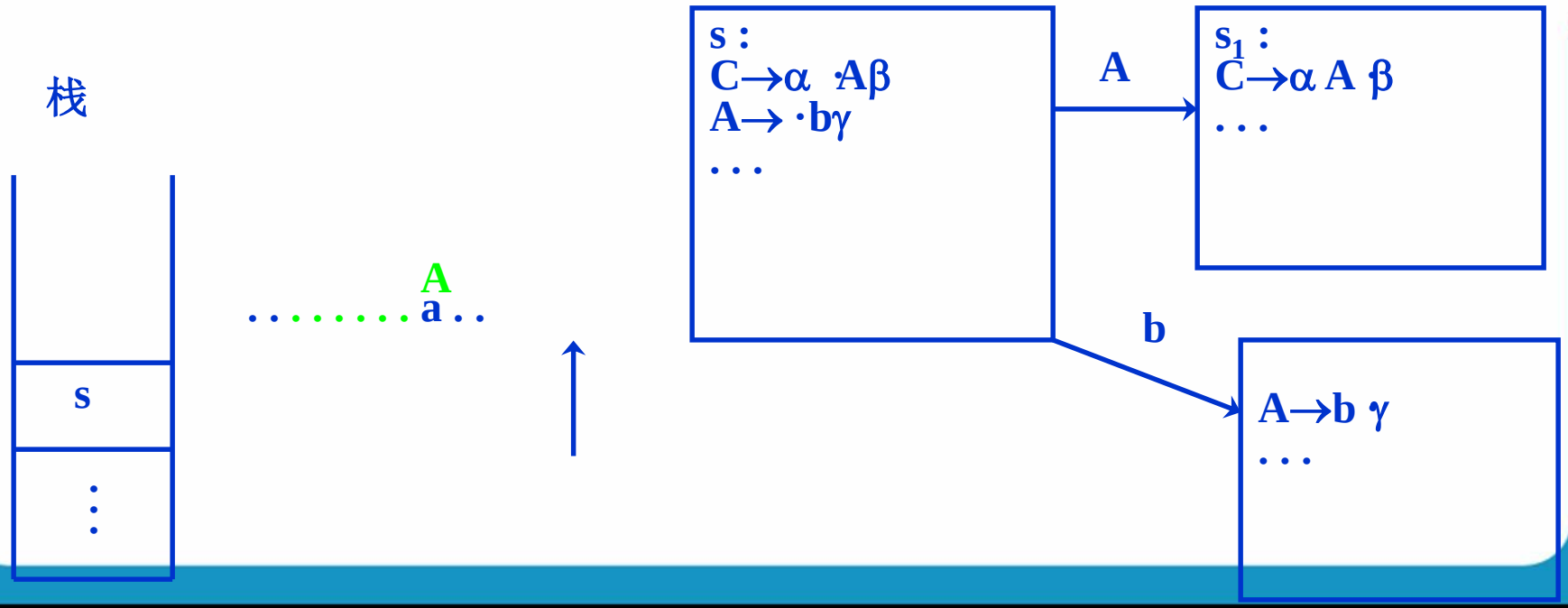


4.6 二义文法的应用

2、紧急方式错误恢复

(1) 退栈，直至出现状态s，它有预先确定的*A*的转移

(2)抛弃若干输入符号，直至找到*a*，它是*A*的合法后继



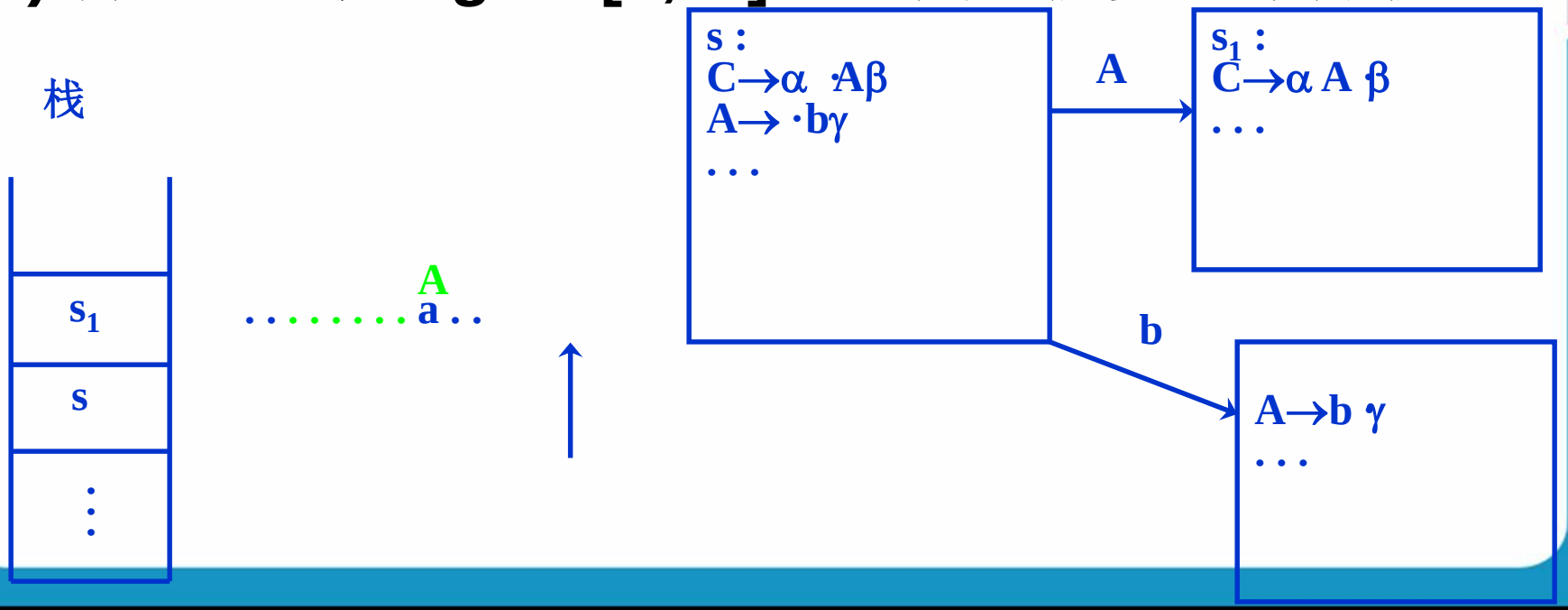
4.6 二义文法的应用

2、紧急方式错误恢复

(1) 退栈，直至出现状态**s**，它有预先确定的***A***的转移

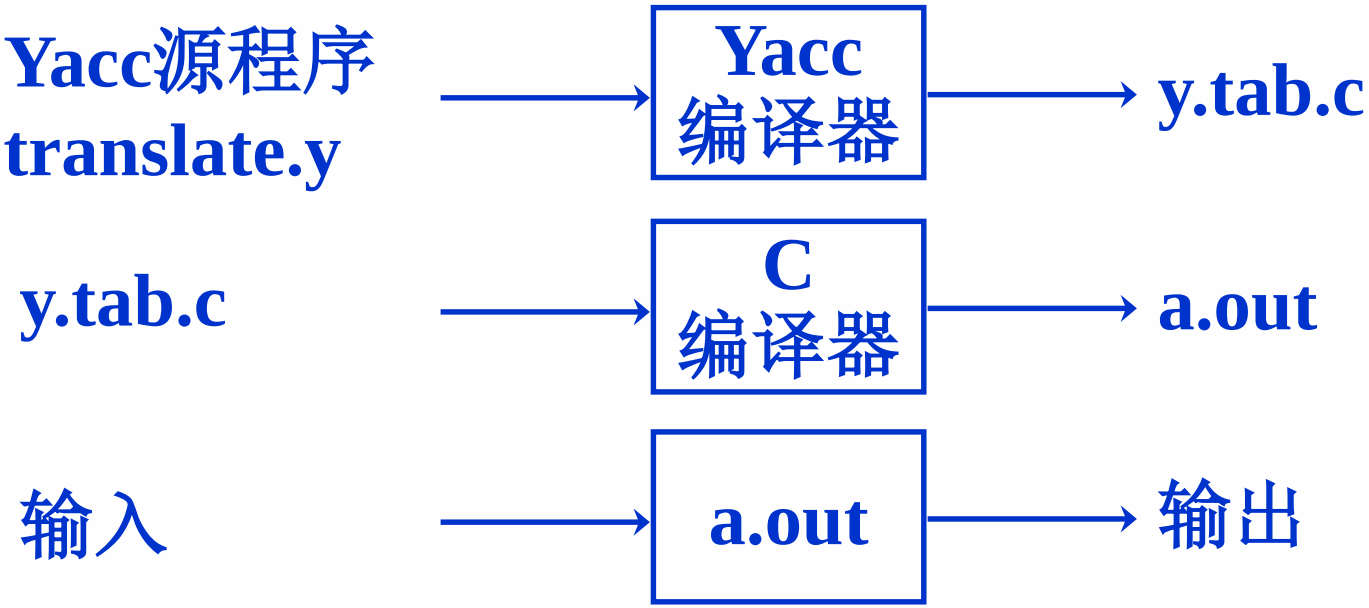
(2)抛弃若干输入符号，直至找到***a***，它是***A***的合法后继

(3) 再把***A***和状态***goto*[*s*, *A*]**压进栈，恢复正常分析



3.7 分析器的生成器

3.7.1 分析器的生成器Yacc



3.7.2 用Yacc处理二义文法

- 例 简单计算器
  - 输入一个表达式并回车，显示计算结果
  - 也可以输入一个空白行

3.7 分析器的生成器

```
%{
# include  <ctype .h>
# include  <stdio.h >
#  define  YYSTYPE  double  /*将栈定义为double类
   型 */
%}

%token  NUMBER
%left  '+'  '–'
%left  '*'  '/ '
%right  UMINUS
%%
```

3.7 分析器的生成器

```
lines       : lines expr '\n'   {printf ( "%g \n", $2 )
   }
            | lines '\n'
            |  /* ε */
            ;
expr        : expr '+' expr     {$$ = $1 + $3; }
            | expr '–' expr     {$$ = $1 – $3; }
            | expr '*' expr     {$$ = $1 * $3; }
            | expr '/ ' expr    {$$ = $1 / $3; }
            | '(' expr ')'      {$$ = $2; }
            | '–' expr  %prec UMINUS   {$$ = –$2; }
            | NUMBER
            ;

%%
```

3.7 分析器的生成器

```
lines     : lines expr '\n'    {printf ( "%g \n", $2 ) }
          | lines '\n'
          |  /* ε */
          ;
expr      : expr '+' expr      {$$ = $1 + $3; }
          | expr '–' expr      {$$ = $1 – $3; }
          | expr '*' expr      {$$ = $1 * $3; }
          | expr '/ ' expr     {$$ = $1 / $3; }
          | '(' expr ')'       {$$ = $2; }
          | '–' expr  %prec UMINUS    {$$ = –$2; }
          | NUMBER
          ;
%%
```

-5+10看成是-(5+10), 还是(-5)+10? 取后者

3.7 分析器的生成器

```
yylex ( ) {
        int c;
        while ( ( c = getchar ( ) ) == ‘  ’ );
        if ( ( c == ‘.’ ) | | (isdigit (c) ) ) {
                ungetc (c, stdin);
                scanf ( “% lf ”,  &yylval);
                return  NUMBER;
        }
        return c;
}
```

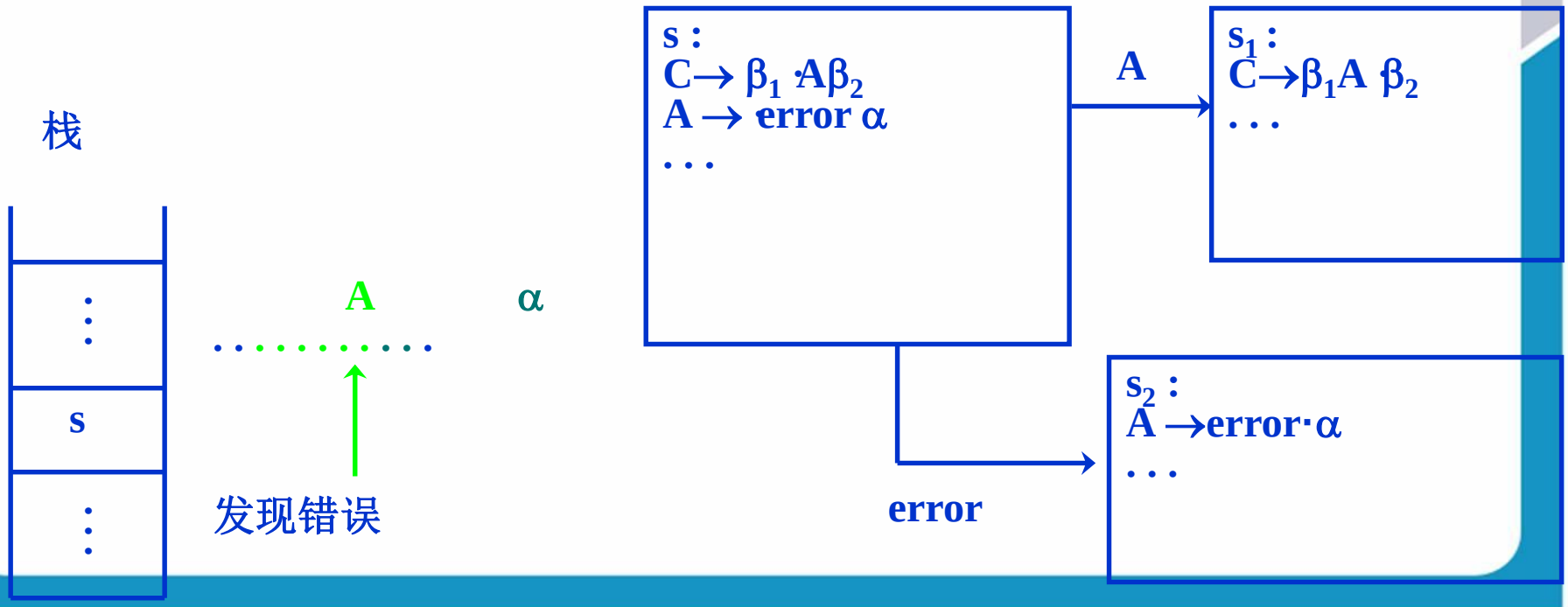
为了C编译器能准确报告yylex函数中错误的位置，需要在生成的程序y.tab.c中使用编译命令#line

3.7.3 Yacc的错误恢复

- 编译器设计者的工作
  - 决定哪些“主要的”非终结符将有错误恢复与它们相关联
  - 为各主要非终结符A加入形式为$A \rightarrow \text{error } \alpha$的错误产生式，其中$\alpha$是文法符号串
  - 为这样的产生式配上语义动作
- Yacc把错误产生式当作普通产生式处理

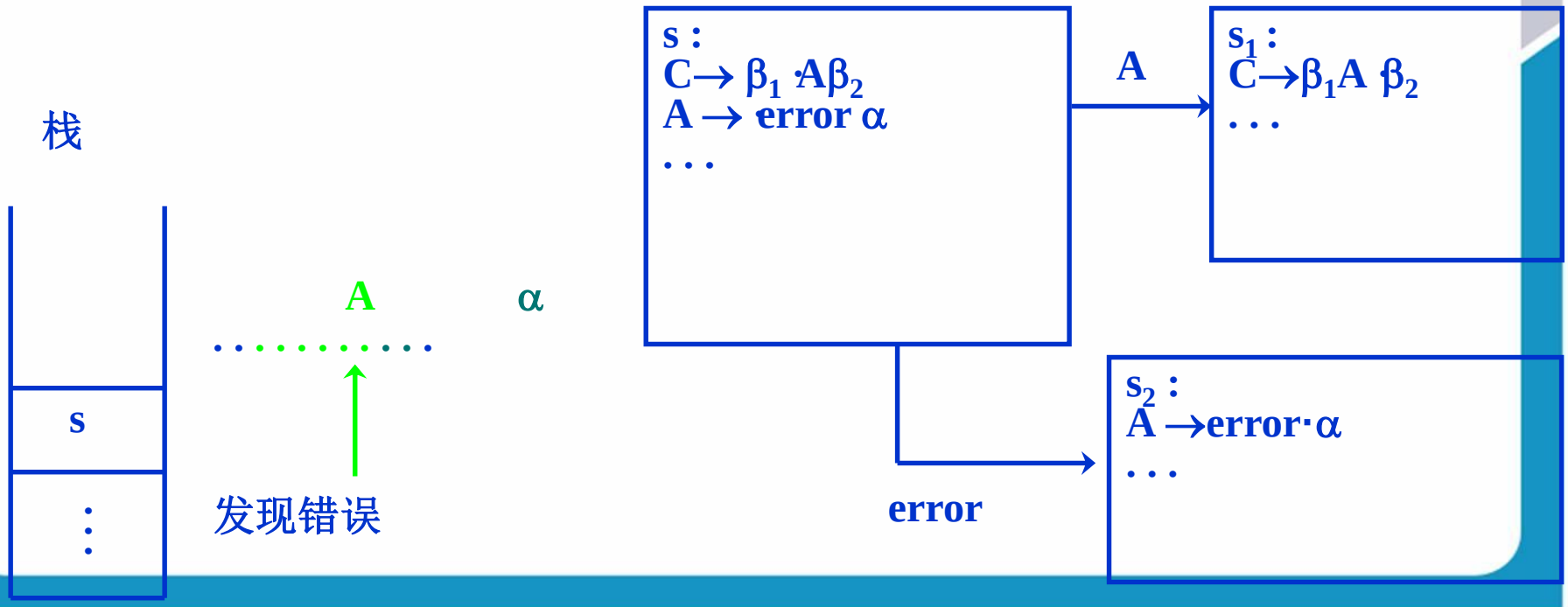
3.7 分析器的生成器

遇到语法错误时



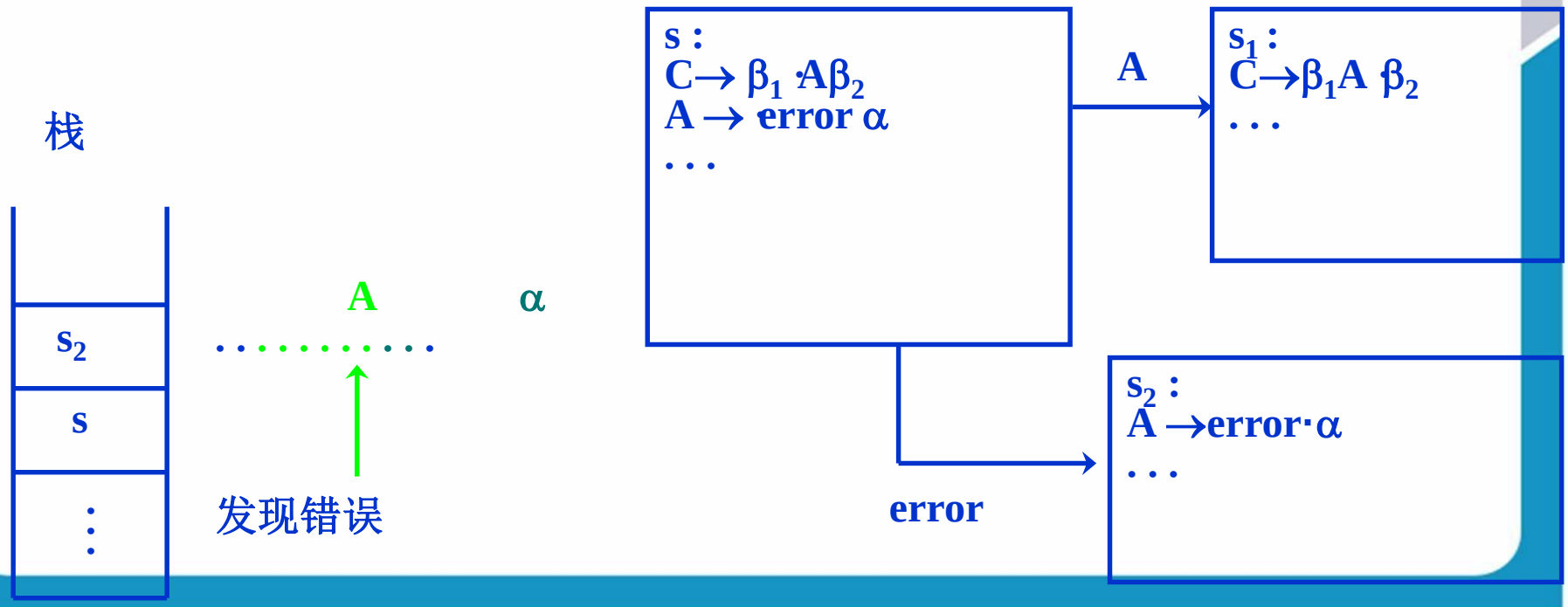
3.7 分析器的生成器

- 遇到语法错误时
  - 从栈中弹出状态，直到发现栈顶状态的项目集包含形为$A \rightarrow \cdot error\ \alpha$的项目为止



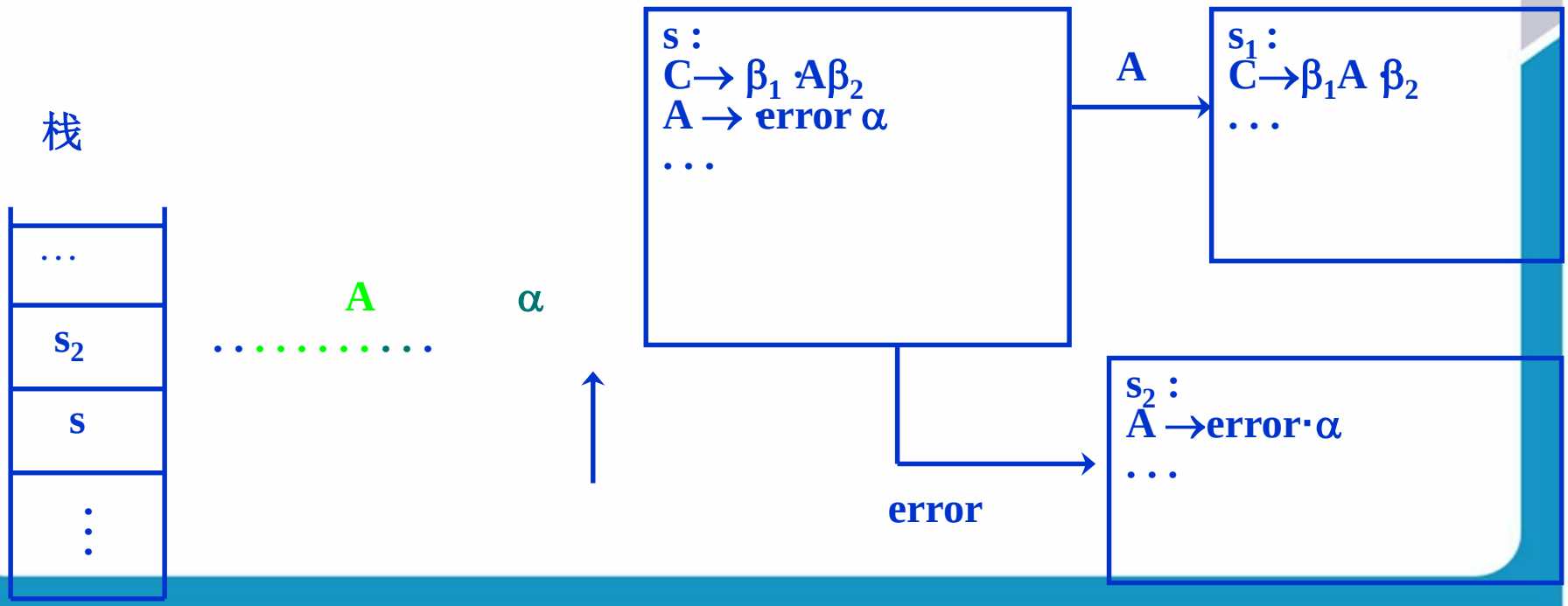
3.7 分析器的生成器

- 遇到语法错误时
  - 从栈中弹出状态，直到发现栈顶状态的项目集包含形为$A \rightarrow \cdot error\ \alpha$的项目为止
  - 把虚构的终结符error“移进”栈



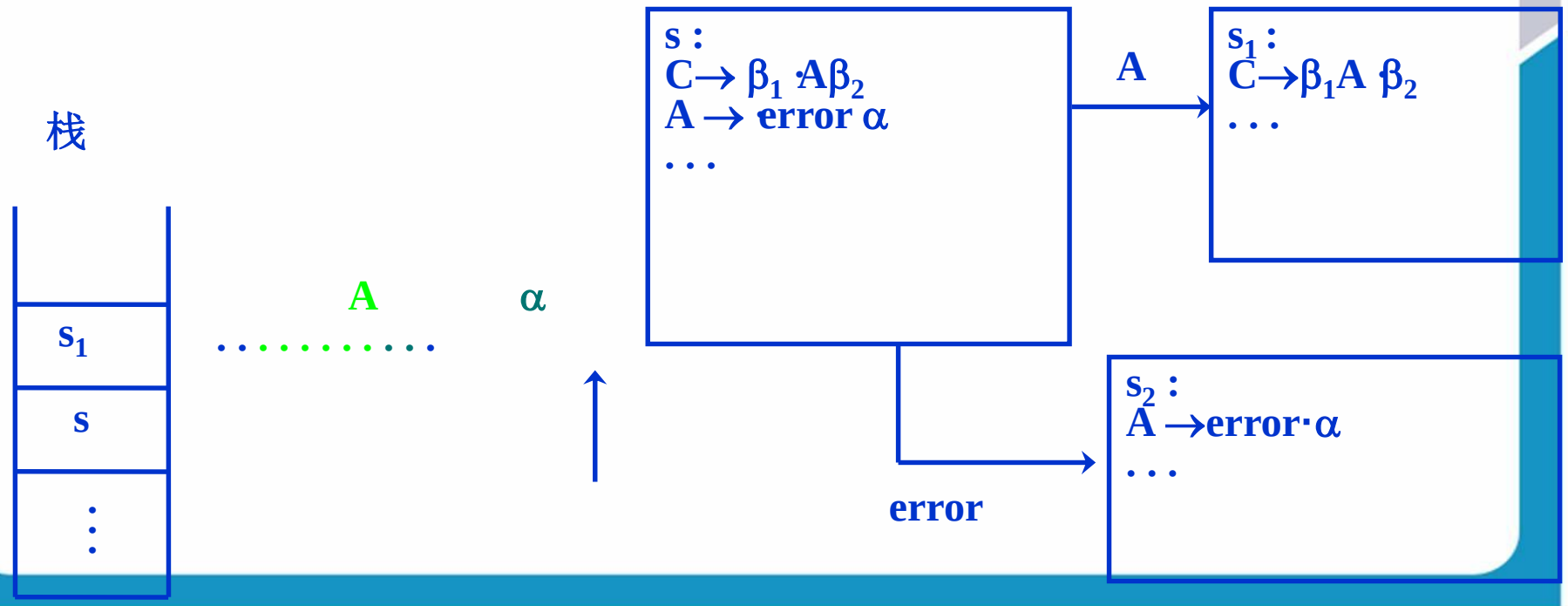
3.7 分析器的生成器

- 遇到语法错误时
  - 从栈中弹出状态，直到发现栈顶状态的项目集包含形为$A \rightarrow \cdot error\ \alpha$的项目为止
  - 把虚构的终结符error“移进”栈
  - 忽略若干输入符号，直至找到α，把α移进栈



3.7 分析器的生成器

- 遇到语法错误时
  - 从栈中弹出状态，直到发现栈顶状态的项目集包含形为$A \rightarrow \cdot error\ \alpha$的项目为止
  - 把虚构的终结符error“移进”栈
  - 忽略若干输入符号，直至找到α，把α移进栈
  - 把error α归约为A，恢复正常分析



3.7 分析器的生成器

- 增加错误恢复的简单计算器

```
lines : lines expr '\n'  {printf ( "%g \n", $2 ) }
              | lines '\n'
              |  /* ε */
              |  error  '\n'{yyerror  (  "重新输入上一行
  " );
                                  yyerrok;}
              ;
```

本 章 要 点

- 文法和语言的基本知识
- 自上而下的分析方法：预测分析，非递归的预测分析，LL(1)文法
- 自下而上的分析方法：SLR(1)方法，规范LR(1)方法和LALR(1)方法
- LR方法如何用于二义文法
- 语法分析的错误恢复方法

例　题　1

下面的二义文法描述命题演算公式的语法，为它写一个等价的非二义文法

$$S \rightarrow S \text{ and } S \mid S \text{ or } S \mid \text{not } S \mid p \mid q \mid (S)$$

非二义文法的产生式如下：

$$E \rightarrow E \text{ or } T \mid T$$

$$T \rightarrow T \text{ and } F \mid F$$

$$F \rightarrow \text{not } F \mid (\ E) \mid p \mid q$$

例　题　1

下面的二义文法描述命题演算公式的语法，为它写一个等价的非二义文法

$S \rightarrow S$ and $S \mid S$ or $S \mid$ not $S \mid p \mid q \mid (S)$

非二义文法的产生式如下：

$E \rightarrow E$ or $T \mid T$

$T \rightarrow T$ and $F \mid F$

$F \rightarrow$ not $E \mid (E) \mid p \mid q$　？

not p and q有两种不同的最左推导

例　题　1

下面的二义文法描述命题演算公式的语法，为它写一个等价的非二义文法

$S \rightarrow S$ and $S \mid S$ or $S \mid$ not $S \mid p \mid q \mid (S)$

非二义文法的产生式如下：

$E \rightarrow E$ or $T \mid T$　　not p and q

$T \rightarrow T$ and $F \mid F$　　not p and q

$F \rightarrow$ not $E \mid (E) \mid p \mid q$　　?

not p and q有两种不同的最左推导

例　题　2

设计一个文法:字母表{*a*, *b*}上*a*和*b*的个数相等的所有串的集合

- 二义文法：　　$S \rightarrow aSbS \mid bSaS \mid \varepsilon$

a<u>*ab*</u>*b*<u>*abab*</u>　　*a*<u>*abba*</u>*b*<u>*ab*</u>

S　*S*　　*S*　*S*

例　题　2

设计一个文法:字母表{*a*, *b*}上*a*和*b*的个数相等的所有串的集合

- 二义文法：　　$S \to a\ S\ b\ S \mid b\ S\ a\ S \mid \varepsilon$

  *a**ab**b**abab*　　*a**abba**b**ab*

- 二义文法：

$$S \to a\ B \mid b\ A \mid \varepsilon$$
$$A \to a\ S \mid b\ A\ A$$
$$B \to b\ S \mid a\ B\ B$$

*a**ab**babab*　　*a**abba**bab*

*a**abbaba**b*

B　*B*　　*B*　*B*　　*B*　*B*

例　题　2

设计一个文法:字母表{*a*, *b*}上*a*和*b*的个数相等的所有串的集合

- 二义文法:　　$S \to a\ S\ b\ S \mid b\ S\ a\ S \mid \varepsilon$

  aabbabab　　*aabbabab*

- 二义文法:　　$S \to a\ B \mid b\ A \mid \varepsilon$
  $A \to a\ S \mid b\ A\ A$
  $B \to b\ S \mid a\ B\ B$

  aabbabab　　*aabbabab*

  aabbabab

- 非二义文法:　　$S \to a\ B\ S \mid b\ A\ S \mid \varepsilon$
  $A \to a \mid b\ A\ A$
  $B \to b \mid a\ B\ B$

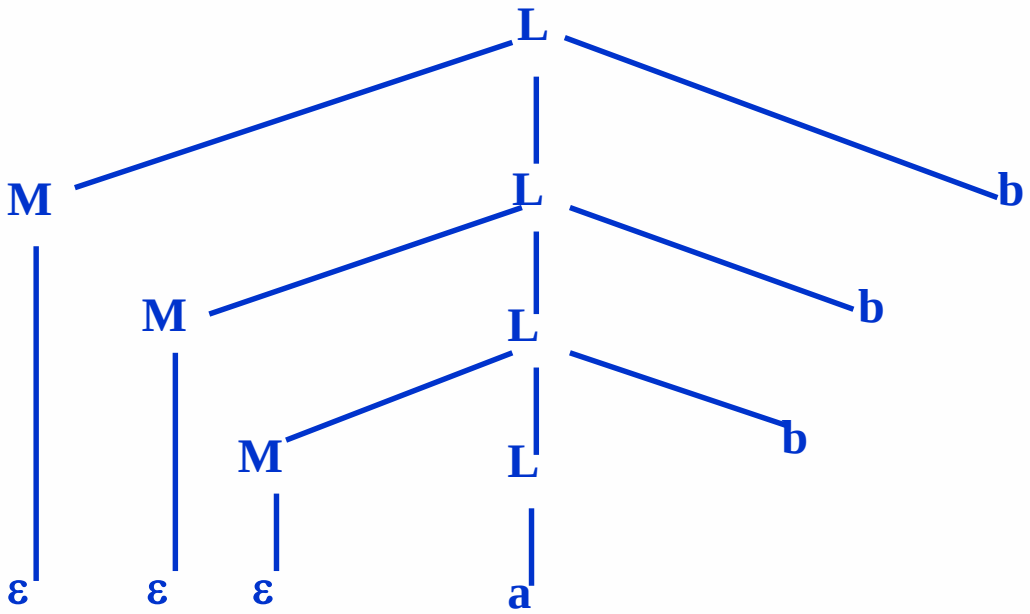
  a abb abab

例　题　3

试说明下面文法不是LR(1)的：

$L \rightarrow M\ L\ b \mid a$

$M \rightarrow \varepsilon$



句子abbb的分析树