



第四章 语法分析

FIRST集和 FOLLOW集的计算

哈尔滨工业大学 陈鄞



计算文法符号 X 的 $FIRST(X)$

➤ $FIRST(X)$: 可以从 X 推导出的所有串首终结符构成的集合

➤ 如果 $X \Rightarrow^* \varepsilon$, 那么 $\varepsilon \in FIRST(X)$

➤ 例

$$\textcircled{1} \quad E \rightarrow TE' \quad FIRST(E) = \{ (\text{ id } \}$$

$$\textcircled{2} \quad E' \rightarrow +TE' | \varepsilon \quad FIRST(E') = \{ + \varepsilon \}$$

$$\textcircled{3} \quad T \rightarrow FT' \quad FIRST(T) = \{ (\text{ id } \}$$

$$\textcircled{4} \quad T' \rightarrow *FT' | \varepsilon \quad FIRST(T') = \{ * \varepsilon \}$$

$$\textcircled{5} \quad F \rightarrow (E)|\text{id} \quad FIRST(F) = \{ (\text{ id } \}$$

算法

- 不断应用下列规则，直到没有新的终结符或 ε 可以被加入到任何 $FIRST$ 集合中为止
 - 如果 X 是一个终结符，那么 $FIRST(X) = \{X\}$
 - 如果 X 是一个非终结符，且 $X \rightarrow Y_1 \dots Y_k \in P$ ($k \geq 1$)，那么如果对于某个 i ， a 在 $FIRST(Y_i)$ 中且 ε 在所有的 $FIRST(Y_1), \dots, FIRST(Y_{i-1})$ 中(即 $Y_1 \dots Y_{i-1} \Rightarrow^* \varepsilon$)，就把 a 加入到 $FIRST(X)$ 中。如果对于所有的 $j = 1, 2, \dots, k$ ， ε 在 $FIRST(Y_j)$ 中，那么将 ε 加入到 $FIRST(X)$
 - 如果 $X \rightarrow \varepsilon \in P$ ，那么将 ε 加入到 $FIRST(X)$ 中

计算串 $X_1X_2 \dots X_n$ 的 *FIRST* 集合

- 向 $FIRST(X_1X_2 \dots X_n)$ 加入 $FIRST(X_1)$ 中所有的非 ε 符号
- 如果 ε 在 $FIRST(X_1)$ 中，再加入 $FIRST(X_2)$ 中的所有非 ε 符号；
如果 ε 在 $FIRST(X_1)$ 和 $FIRST(X_2)$ 中，再加入 $FIRST(X_3)$ 中的所有非 ε 符号，以此类推
- 最后，如果对所有的 i ， ε 都在 $FIRST(X_i)$ 中，那么将 ε 加入到 $FIRST(X_1X_2 \dots X_n)$ 中

计算非终结符 A 的 $FOLLOW(A)$

➤ $FOLLOW(A)$: 可能在某个句型中紧跟在 A 后边的终结符 a 的集合

$$FOLLOW(A) = \{a \mid S \Rightarrow^* \alpha A a \beta, a \in V_T, \alpha, \beta \in (V_T \cup V_N)^*\}$$

➤ 如果 A 是某个句型的的最右符号, 则将结束符“\$”添加到 $FOLLOW(A)$ 中

例

- ① $E \rightarrow TE'$ $FIRST(E) = \{ (\text{id} \}$ $FOLLOW(E) = \{ \$) \}$
- ② $E' \rightarrow +TE' \mid \varepsilon$ $FIRST(E') = \{ + \varepsilon \}$ $FOLLOW(E') = \{ \$) \}$
- ③ $T \rightarrow FT'$ $FIRST(T) = \{ (\text{id} \}$ $FOLLOW(T) = \{ + \$) \}$
- ④ $T' \rightarrow *FT' \mid \varepsilon$ $FIRST(T') = \{ * \varepsilon \}$ $FOLLOW(T') = \{ + \$) \}$
- ⑤ $F \rightarrow (E) \mid \text{id}$ $FIRST(F) = \{ (\text{id} \}$ $FOLLOW(F) = \{ * + \$) \}$

算法

- 不断应用下列规则，直到没有新的终结符可以被加入到任何 *FOLLOW* 集合中为止
- 将 $\$$ 放入 *FOLLOW*(S) 中，其中 S 是开始符号， $\$$ 是输入右端的结束标记
- 如果存在一个产生式 $A \rightarrow \alpha B \beta$ ，那么 *FIRST*(β) 中除 ϵ 之外的所有符号都在 *FOLLOW*(B) 中
- 如果存在一个产生式 $A \rightarrow \alpha B$ ，或存在产生式 $A \rightarrow \alpha B \beta$ 且 *FIRST*(β) 包含 ϵ ，那么 *FOLLOW*(A) 中的所有符号都在 *FOLLOW*(B) 中

例：表达式文法各产生式的SELECT集

X	$FIRST(X)$	$FOLLOW(X)$
E	(id	\$)
E'	+ ϵ	\$)
T	(id	+) \$
T'	* ϵ	+) \$
F	(id	* +) \$

表达式文法是 $LL(1)$ 文法

- (1) $E \rightarrow T E'$ $SELECT(1) = \{ (\text{ id } \}$
- (2) $E' \rightarrow + T E'$ $SELECT(2) = \{ + \}$
- (3) $E' \rightarrow \epsilon$ $SELECT(3) = \{ \$) \}$
- (4) $T \rightarrow F T'$ $SELECT(4) = \{ (\text{ id } \}$
- (5) $T' \rightarrow * F T'$ $SELECT(5) = \{ * \}$
- (6) $T' \rightarrow \epsilon$ $SELECT(6) = \{ +) \$ \}$
- (7) $F \rightarrow (E)$ $SELECT(7) = \{ (\}$
- (8) $F \rightarrow \text{id}$ $SELECT(8) = \{ \text{id} \}$

预测分析表

	产生式	<i>SELECT</i>
<i>E</i>	$E \rightarrow TE'$	(id
<i>E'</i>	$E' \rightarrow +TE'$	+
	$E' \rightarrow \varepsilon$	\$)
<i>T</i>	$T \rightarrow FT'$	(id
<i>T'</i>	$T' \rightarrow *FT'$	*
<i>F</i>	$F \rightarrow (E)$	(
	$F \rightarrow id$	id

非终结符	输入符号					
	id	+	*	()	\$
<i>E</i>	$E \rightarrow TE'$			$E \rightarrow TE'$		
<i>E'</i>		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
<i>T</i>	$T \rightarrow FT'$			$T \rightarrow FT'$		
<i>T'</i>		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
<i>F</i>	$F \rightarrow id$			$F \rightarrow (E)$		

$LL(1)$ 文法的分析方法

- 递归的预测分析法
- 非递归的预测分析法



第四章 语法分析

FIRST集和 FOLLOW集的计算

哈尔滨工业大学 陈鄞





第四章 语法分析

递归的预测分析法

哈尔滨工业大学 陈鄞



递归的预测分析法

- 递归的预测分析法是指：在递归下降分析中，根据预测分析表进行产生式的选择
- 根据每个非终结符的产生式和LL(1)文法的预测分析表，为每个非终结符编写对应的过程

```
void A() {  
1)  选择一个A产生式,  $A \rightarrow X_1 X_2 \dots X_k$  ;  
2)  for (  $i = 1$  to  $k$  ) {  
3)      if (  $X_i$  是一个非终结符号 )  
4)          调用过程  $X_i()$  ;  
5)      else if (  $X_i$  等于当前的输入符号  $a$  )  
6)          读入下一个输入符号;  
7)      else /* 发生了一个错误 */;  
    }  
}
```

例

- (1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$
- (2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$
- (3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$
- (4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$
- (5) $\langle \text{STLIST} \rangle \rightarrow \text{s } \langle \text{STLISTN} \rangle$
- (6) $\langle \text{STLISTN} \rangle \rightarrow ; \text{s } \langle \text{STLISTN} \rangle$
- (7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$
- (8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$
- (9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}
SELECT(7)={end}

```
program DESCENT;  
  begin  
    GETNEXT(TOKEN);  
    PROGRAM(TOKEN);  
    GETNEXT(TOKEN);  
    if TOKEN≠'$' then ERROR;  
  end
```

例

(1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$

(2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$

(3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$

(4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$

(5) $\langle \text{STLIST} \rangle \rightarrow \text{s } \langle \text{STLISTN} \rangle$

(6) $\langle \text{STLISTN} \rangle \rightarrow ; \text{s } \langle \text{STLISTN} \rangle$

(7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$

(8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$

(9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}

SELECT(7)={end}

```
procedure PROGRAM(TOKEN);
begin
  → if TOKEN≠'program' then ERROR;
    GETNEXT(TOKEN);
  → DECLIST(TOKEN);
  → if TOKEN≠':' then ERROR;
    GETNEXT(TOKEN);
  → TYPE(TOKEN);
    GETNEXT(TOKEN);
  → STLIST(TOKEN);
  → if TOKEN≠'end' then ERROR;
end
```

例

(1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$

(2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$

(3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$

(4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$

(5) $\langle \text{STLIST} \rangle \rightarrow \text{s } \langle \text{STLISTN} \rangle$

(6) $\langle \text{STLISTN} \rangle \rightarrow ; \text{s } \langle \text{STLISTN} \rangle$

(7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$

(8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$

(9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}

SELECT(7)={end}

```
procedure DECLIST(TOKEN);  
  begin  
    if TOKEN≠'id' then ERROR;  
  
    GETNEXT(TOKEN);  
    DECLISTN(TOKEN);  
  end
```

例

(1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$

(2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$

(3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$

(4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$

(5) $\langle \text{STLIST} \rangle \rightarrow \text{s } \langle \text{STLISTN} \rangle$

(6) $\langle \text{STLISTN} \rangle \rightarrow ; \text{s } \langle \text{STLISTN} \rangle$

(7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$

(8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$

(9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}

SELECT(7)={end}

```
procedure DECLISTN(TOKEN);  
  begin  
    if TOKEN = ',' then  
      begin  
        GETNEXT(TOKEN);  
        if TOKEN ≠ 'id' then ERROR;  
  
        GETNEXT(TOKEN);  
        DECLISTN(TOKEN);  
      end  
    else if TOKEN ≠ ':' then ERROR;  
  end
```


例

(1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$

(2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$

(3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$

(4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$

(5) $\langle \text{STLIST} \rangle \rightarrow s \langle \text{STLISTN} \rangle$

(6) $\langle \text{STLISTN} \rangle \rightarrow ; s \langle \text{STLISTN} \rangle$

(7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$

(8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$

(9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}

SELECT(7)={end}

```
procedure STLIST(TOKEN);  
  begin  
    if TOKEN≠'s' then ERROR;  
    GETNEXT(TOKEN);  
    STLISTN(TOKEN);  
  end
```

例

(1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$

(2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$

(3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$

(4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$

(5) $\langle \text{STLIST} \rangle \rightarrow \text{s } \langle \text{STLISTN} \rangle$

(6) $\langle \text{STLISTN} \rangle \rightarrow ; \text{s } \langle \text{STLISTN} \rangle$

(7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$

(8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$

(9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}

SELECT(7)={end}

```
procedure STLISTN(TOKEN);  
  begin  
    if TOKEN = ';' then  
      begin  
        GETNEXT(TOKEN);  
        if TOKEN ≠ 's' then ERROR;  
  
        GETNEXT(TOKEN);  
        STLISTN(TOKEN);  
      end  
    else if TOKEN ≠ 'end' then ERROR;  
  
  end
```

例

(1) $\langle \text{PROGRAM} \rangle \rightarrow \text{program } \langle \text{DECLIST} \rangle : \langle \text{TYPE} \rangle ; \langle \text{STLIST} \rangle \text{ end}$

(2) $\langle \text{DECLIST} \rangle \rightarrow \text{id } \langle \text{DECLISTN} \rangle$

(3) $\langle \text{DECLISTN} \rangle \rightarrow , \text{id } \langle \text{DECLISTN} \rangle$

(4) $\langle \text{DECLISTN} \rangle \rightarrow \varepsilon$

(5) $\langle \text{STLIST} \rangle \rightarrow \text{s } \langle \text{STLISTN} \rangle$

(6) $\langle \text{STLISTN} \rangle \rightarrow ; \text{s } \langle \text{STLISTN} \rangle$

(7) $\langle \text{STLISTN} \rangle \rightarrow \varepsilon$

(8) $\langle \text{TYPE} \rangle \rightarrow \text{real}$

(9) $\langle \text{TYPE} \rangle \rightarrow \text{int}$

SELECT(4)={:}
SELECT(7)={end}

```
procedure TYPE(TOKEN);  
  begin  
    if TOKEN≠'real' and TOKEN≠'int'  
    then ERROR;  
  end
```




第四章 语法分析

递归的预测分析法

哈尔滨工业大学 陈鄞





第四章 语法分析

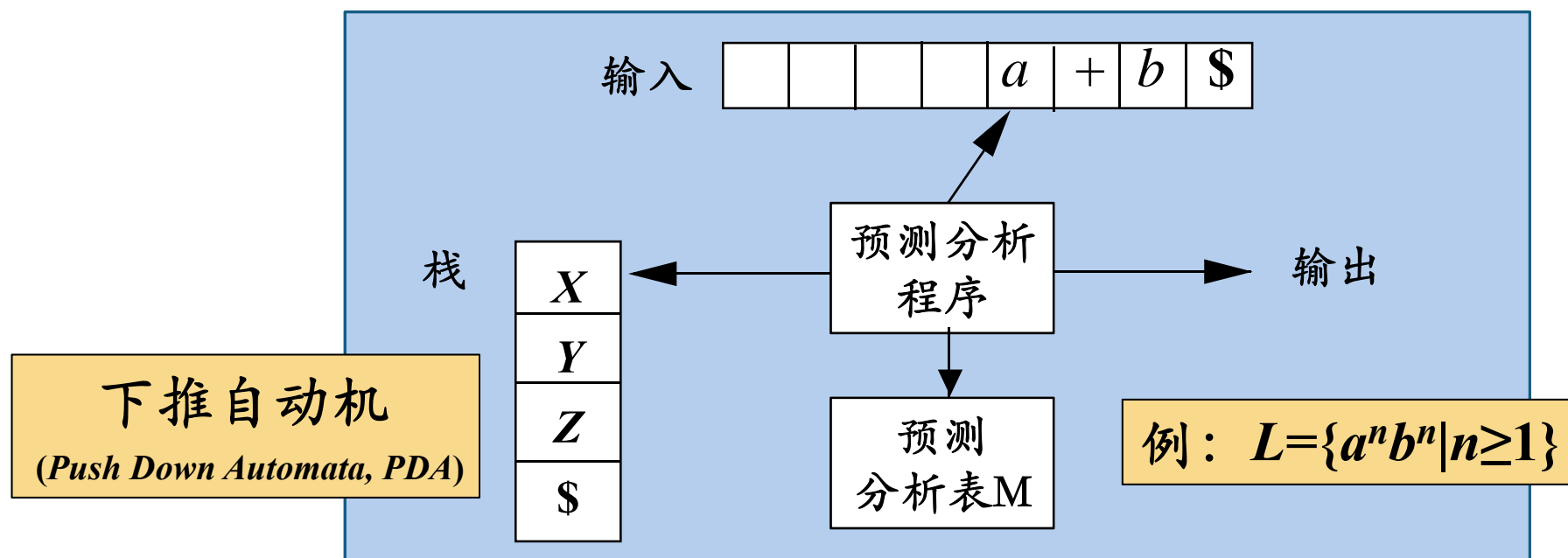
非递归的预测分析法

哈尔滨工业大学 陈鄞



非递归的预测分析法

- 非递归的预测分析不需要为每个非终结符编写递归下降过程，而是根据预测分析表构造一个自动机，也叫表驱动的分析



例

非终结符	输入符号					
	id	+	*	()	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow id$			$F \rightarrow (E)$		

如果 w 是至今为止已经匹配完成的输入部分，那么栈中保存的文法符号序列 α 满足
 $S \Rightarrow_{lm}^* w\alpha$

栈	剩余输入	输出
$E \$$	$id+id*id \$$	
$TE' \$$	$id+id*id \$$	$E \rightarrow TE'$
$FT'E' \$$	$id+id*id \$$	$T \rightarrow FT'$
$idT'E' \$$	$id+id*id \$$	$F \rightarrow id$
$T'E' \$$	$+id*id \$$	
$E' \$$	$+id*id \$$	$T' \rightarrow \epsilon$
$+TE' \$$	$+id*id \$$	$E' \rightarrow +TE'$
$TE' \$$	$id*id \$$	
$FT'E' \$$	$id*id \$$	$T \rightarrow FT'$
$idT'E' \$$	$id*id \$$	$F \rightarrow id$
$T'E' \$$	$*id \$$	
$*FT'E' \$$	$*id \$$	$T' \rightarrow *FT'$
$FT'E' \$$	$id \$$	
$idT'E' \$$	$id \$$	$F \rightarrow id$
$T'E' \$$	$\$$	
$E' \$$	$\$$	$T' \rightarrow \epsilon$
$\$$	$\$$	$E' \rightarrow \epsilon$

表驱动的预测分析法

- 输入：一个串 w 和文法 G 的分析表 M
- 输出：如果 w 在 $L(G)$ 中，输出 w 的最左推导；否则给出错误指示
- 方法：最初，语法分析器的格局如下：输入缓冲区中是 $w\$$ ， G 的开始符号位于栈顶，其下面是 $\$$ 。下面的程序使用预测分析表 M 生成了处理这个输入的预测分析过程


```
设置 $ip$ 使它指向 $w$ 的第一个符号，其中 $ip$ 是输入指针；
令 $X$ =栈顶符号；
while (  $X \neq \$$  ) { /* 栈非空 */
    if (  $X$  等于  $ip$  所指向的符号  $a$  ) 执行栈的弹出操作，将  $ip$  向前移动一个位置；
    else if (  $M$  是一个终结符号 )  $error()$ ；
    else if (  $M[X, a]$  是一个报错条目 )  $error()$ ；
    else if (  $M[X, a] = X \rightarrow Y_1 Y_2 \dots Y_k$  ) {
        输出产生式  $X \rightarrow Y_1 Y_2 \dots Y_k$ ；
        弹出栈顶符号；
        将  $Y_k, Y_{k-1} \dots, Y_1$  压入栈中，其中  $Y_1$  位于栈顶。
    }
    令  $X$ =栈顶符号
}
```


递归的预测分析法vs.非递归的预测分析法

	递归的预测分析法	非递归的预测分析法
程序规模	程序规模较大， 不需载入分析表	主控程序规模较小， 需载入分析表（表较小） 😊
直观性	较好 😊	较差
效率	较低	分析时间大约正比于待分析程序的长度 😊
自动生成	较难	较易 😊

预测分析法实现步骤

- 1) 构造文法
- 2) 改造文法：消除二义性、消除左递归、消除回溯
- 3) 求每个变量的 $FIRST$ 集和 $FOLLOW$ 集，从而求得每个候选式的 $SELECT$ 集
- 4) 检查是不是 $LL(1)$ 文法。若是，构造预测分析表
- 5) 对于递归的预测分析，根据预测分析表为每一个非终结符编写一个过程；对于非递归的预测分析，实现表驱动的预测分析算法




第四章 语法分析

非递归的预测分析法

哈尔滨工业大学 陈鄞





第四章 语法分析

预测分析中的错误处理

哈尔滨工业大学 陈鄞



预测分析中的错误检测

- 两种情况下可以检测到错误
 - 栈顶的终结符和当前输入符号不匹配
 - 栈顶非终结符与当前输入符号在预测分析表对应项中的信息为空

预测分析中的错误恢复

➤ 恐慌模式

- 忽略输入中的一些符号，直到输入中出现由设计者选定的 **同步词法单元** (*synchronizing token*) 集合中的某个词法单元
 - 其效果依赖于 **同步集合的选取**。集合的选取应该使得语法分析器能从实际遇到的错误中 **快速恢复**
 - 例如可以把 ***FOLLOW(A)*** 中的 **所有终结符** 放入非终结符 *A* 的同步记号集合
- 如果终结符在栈顶而不能匹配，一个简单的办法就是弹出此终结符

例

非终结符	输入符号					
	id	+	*	()	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$	<i>synch</i>	<i>synch</i>
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$	<i>synch</i>		$T \rightarrow FT'$	<i>synch</i>	<i>synch</i>
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \text{id}$	<i>synch</i>	<i>synch</i>	$F \rightarrow (E)$	<i>synch</i>	<i>synch</i>

X	$\text{FOLLOW}(X)$
E	$\$)$
E'	$\$)$
T	$+) \$$
T'	$+) \$$
F	$* +) \$$

*Synch*表示根据相应非终结符的*FOLLOW*集得到的同步词法单元


➤ 分析表的使用方法

- 如果 $M[A,a]$ 是空，表示检测到错误，根据恐慌模式，忽略输入符号 a
- 如果 $M[A,a]$ 是*synch*，则弹出栈顶的非终结符 A ，试图继续分析后面的语法成分
- 如果栈顶的终结符和输入符号不匹配，则弹出栈顶的终结符



非终结符	输入符号					
	id	+	*	()	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$	synch	synch
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$	synch		$T \rightarrow FT'$	synch	synch
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \text{id}$	synch	synch	$F \rightarrow (E)$	synch	synch

栈	剩余输入	
$E \$$	$+id*+id \$$	ignore +
$E \$$	$id*+id \$$	
$TE' \$$	$id*+id \$$	
$FT'E' \$$	$id*+id \$$	
$idTE' \$$	$id*+id \$$	
$T'E' \$$	$*+id \$$	
$*FT'E' \$$	$*+id \$$	
$FT'E' \$$	$+id \$$	error
$T'E' \$$	$+id \$$	
$E' \$$	$+id \$$	
$+TE' \$$	$+id \$$	
$TE' \$$	$id \$$	
$FT'E' \$$	$id \$$	
$idTE' \$$	$id \$$	
$T'E' \$$		\$
$E' \$$	\$	
\$	\$	



第四章 语法分析

预测分析中的错误处理

哈尔滨工业大学 陈鄞

