

第4章 自顶向下语法分析方法

语法分析的作用是识别由词法分析给出的单词符号序列是否是给定文法的正确句子（程序）。

常用的语法分析分两大类：

一、自顶向下分析

二、自底向上分析

其中自顶向下分析，就是从文法的开始符号出发企图推导出与输入的单词串完全匹配的句子，若输入串是给定文法的句子，则必能推出，反之必然出错。

第 4 章 自顶向下语法分析方法

4.1 确定的自顶向下分析思想

4.2 LL(1) 文法的判别

4.3 某些非LL(1) 文法到LL(1) 文法的等价变换

4.4 确定的自顶向下分析方法

4.5 实例：PL/0 编译程序的语法分析

本章要点

4.1 确定的自顶向下分析思想

定义 4.1 设 $G = (V_T, V_N, S, P)$ 是上下文无关文法，

$$\text{FIRST}(\alpha) = \{a \mid \alpha \Rightarrow a\beta, a \in V_T, \alpha, \beta \in V^*\}$$

若 $\alpha \Rightarrow \varepsilon$ ，则规定 $\varepsilon \in \text{FIRST}(\alpha)$ 。

因此，对于形如 $A \rightarrow \alpha \mid \beta$ 的产生式， $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$ ，

则当 $\text{FIRST}(\alpha) \cap \text{FIRST}(\beta) = \emptyset$ 时，对 A 的

定义 4.2 设 $G = (V_T, V_N, S, P)$ 是上下文无关文法， $A \in V_N$ ， S 是开始符号。

$$\text{FOLLOW}(A) = \{a \mid S \Rightarrow \mu A \beta \text{ 且 } a \in \text{FIRST}(\beta), \mu \in V_T^*, \beta \in V^+\}$$

若 $S \Rightarrow \mu A \beta$ ，且 $\beta \Rightarrow \varepsilon$ ，则

$\# \in \text{FOLLOW}(A)$ 。 $\#$ 为输入串的左右界符。

也可定义为：

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

若有 $S \Rightarrow \dots A$ ，则规定 $\# \in \text{FOLLOW}(A)$ 。

因此，对于形如 $A \rightarrow \alpha | \beta$ 的产生式，设 $\alpha \Rightarrow \varepsilon$, $\beta \Rightarrow \varepsilon$

,

则当 (1) $\text{FOLLOW} (A) \cap \text{FIRST}(\beta) = \phi$

(2) $\text{FIRST} (\alpha) \cap \text{FIRST}(\beta) = \phi$ 时

对 A 的替换可唯一确定用 α 还是用 β 。

合并 (1)、(2) 条件表示为：

定义 4.3 给定上下文无关文法的产生式 $A \rightarrow \alpha$

$(A \in V_N, \alpha \in V^*, \text{若 } \alpha \Rightarrow \varepsilon, \text{ 则})$

综合以上情况定义选择集合 **SELECT** 如下：

$\text{SELECT} (A \rightarrow \alpha) = \text{FIRST} (\alpha)$

如果 $\alpha \Rightarrow \varepsilon$, 则

定义 4.4 一个上下文无关文法是 LL (1) 文法的充分必要条件是, 对每个非终结符 A 的两个不同产生式,
 $A \rightarrow \alpha$, $A \rightarrow \beta$, 满足

$$\text{SELECTC} (A \rightarrow \alpha) \cap \text{SELECT} (A \rightarrow \beta) = \phi$$

其中 α 、 β 不能同时 $\Rightarrow \varepsilon$

例： 设文法 G[S] 为：

$S \rightarrow aAS$

$S \rightarrow b$

$A \rightarrow bA$

$A \rightarrow \varepsilon$

因为 $\text{SELECT} (A \rightarrow bA) = \{ b \}$

$$\begin{aligned} \text{SELECT}(A \rightarrow \varepsilon) &= \{ \varepsilon \} \setminus \{ \varepsilon \} \cup \text{FOLLOW}(A) \\ &= \text{FIRST}(S) = \{ a, b \} \end{aligned}$$

则 $\text{SELECT} (A \rightarrow bA) \cap$

$\text{SELECT}(A \rightarrow \varepsilon) = \{ b \} \neq \phi$

所以此文法不是 LL (1) 文法。

4.2 LL (1) 文法的判别

当我们需选用自顶向下分析技术时，首先必须判别所给文法是否是 LL (1) 文法。因而我们对任给文法需计算 FIRST、FOLLOW、SELECT 集合，进而判别文法是否为 LL (1) 文法。

1、计算 FIRST 集

定义： $\text{FIRST}(\alpha) = \{a \mid \alpha \Rightarrow^* a\beta, a \in V_T, \alpha, \beta \in V^*\}$

(1) 求 $\text{FIRST}(x)$ ， $x \in V$

(a) 若 $x \in V_T$ ，则 $\text{FIRST}(x) = \{x\}$

(b) 若 x 是 ε ，则 $\text{FIRST}(x) = \{\varepsilon\}$

(c) 若 $x \in V_N$ ，且 $x \rightarrow y_1 y_2 \dots y_m \mid \dots \mid z_1 z_2 \dots z_n$

则 $\text{FIRST}(x) = \text{FIRST}(y_1 y_2 \dots y_m) \cup \dots \cup \text{FIRST}(z_1 z_2 \dots z_n)$

1、计算 FIRST 集

(2) 求 $\text{FIRST}(y_1y_2\dots y_m)$, 其中 $y_1, y_2, \dots, y_m \in V$ 。 (a) 若 $y_1 \in VT$, 则

$$\text{FIRST} (y_1y_2\dots y_m) = \{y_1\}$$

(b) 若 $y_1 \in VN$, $\varepsilon \notin \text{FIRST}(y_1)$, 则

$$\text{FIRST} (y_1y_2\dots y_m) = \text{FIRST}(y_1)$$

若 $\varepsilon \in \text{FIRST}(y_1)$, 则


$$\text{FIRST}(y_1y_2\dots y_m) = (\text{FIRST}(y_1) \setminus \{\varepsilon\}) \cup$$

$$\text{FIRST}(y_2y_3\dots y_m)$$

按上法求 $\text{FIRST} (y_2y_3\dots y_m)$, 类推下去。

2、计算 FOLLOW 集

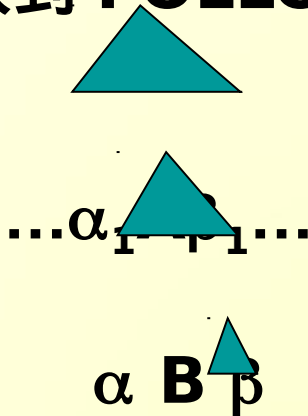
对文法中每一 $A \in V_N$ ，计算 $FOLLOW(A)$

(a) 设 S 为文法的开始符号，则 $\# \in FOLLOW(S)$
即 

输入串 $\#$

(b) 若有 $A \rightarrow \alpha B \beta$ ，则将 $FIRST(\beta) - \{\epsilon\}$ 加入到 $FOLLOW(B)$ 中，如果其中 $\beta \Rightarrow \epsilon$ ，则将 $FOLLOW(A)$ 加入到 $FOLLOW(B)$ 中。

即：



ϵ

3、计算 SELECT 集

定义： $\text{SELECT} (A \rightarrow \alpha) = \text{FIRST} (\alpha)$

其中 $\alpha \neq \varepsilon$ 。

若 $\alpha^* \Rightarrow \varepsilon$ ，则

$\text{SELECT} (A \rightarrow \alpha) =$

$(\text{FIRST} (\alpha) - \{\varepsilon\})$

$\cup \text{FOLLOW}(A)$

3 某些非 LL (1) 文法到 LL (1) 文法的等价变换

对于一个语言的非 LL (1) 文法是否能变换为等价的 LL (1) 形式以及如何变换是本节讨论的主要问题。

由 LL (1) 文法的定义可知，若文法中含有直接或间接左递归，或含有左公共因子，则该文法肯定不是 LL (1) 文法。

1. 提取左公共因子

若文法中含有形如： $A \rightarrow \alpha\beta | \alpha\gamma$ 的产生式，
进行等价变换为： $A \rightarrow \alpha (\beta | \gamma)$ 其中 ‘(’ , ‘)’ 为元符号，
可进一步引进新非终结符 A' ，去掉 ‘(’ , ‘)’ 使产生式变换为：

$$A \rightarrow \alpha A'$$

$$A' \rightarrow \beta | \gamma$$

写成一般形式为：

$$A \rightarrow \alpha\beta_1 | \alpha\beta_2 | \dots | \alpha\beta_n$$

提取左公共因子后变为：

$$A \rightarrow \alpha (\beta_1 | \beta_2 | \dots | \beta_n)$$

再引进非终结符 A' ，变为：

$$A \rightarrow \alpha A'$$

$$A' \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$$

若在 β_i 、 β_j 、 $\beta_k \dots$ （其中 $1 \leq i, j, k \leq n$ ）中仍含有左公共因子，这时可再次提取，这样反复进行提取直到引进新非终结符的有关产生式再无左公共因子为止。

若文法中含隐式左公共因子，可先转换为显式左公共因子，再提取左公共因子。



- (1) 是否每个文法的左公共因子都能在有限的步骤内替换成无左公共因子的文法。
- (2) 一个文法提取了左公共因子后，若文法不含空产生式，且无左递归时，则改写后的文法是 LL(1) 文法，为什么？

2、消除左递归

(1) 左递归定义:

(a) 直接左递归: 文法中含有 $A \rightarrow A \alpha$ 形式的产生式;

(b) 间接左递归: 文法中同时含有 $A \rightarrow B$
 β , $B \rightarrow A \alpha$ 形式的产生式。

(2) 确定的自上而下分析要求文法不含左递归

(a) 文法含左递归不便于使推导按从左往右的顺序匹配, 甚至使分析发生死循环。

(b) 含左递归的文法不是 LL (1) 文法。

(3) 消除文法左递归的方法

(a) 消除直接左递归

一般情况下，直接左递归的形式为：

$$A \rightarrow A \alpha_1 | A \alpha_2 | \dots | A \alpha_m | \beta_1 | \beta_2 | \dots | \beta_n$$

消除左递归后改写为：

$$A \rightarrow \beta_1 A' | \beta_2 A' | \dots | \beta_n A'$$

$$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots | \alpha_m A' | \varepsilon$$

(b) 消除间接左递归

先通过产生式的替换，将间接左递归变为直接左递归，然后再消除直接左递归。

4.4 确定的自顶向下分析方法

4.4.1 递归子程序法

1、基本思想

对每一非终结符构造一个过程，每个过程的功能是识别由该非终结符推出的串。

2、编写程序

IP：是输入串指示器，开始工作前 **IP** 指向串的第一个符号，每个程序工作完后，**IP** 指向下一个未处理符号。

sym：表示 **IP** 所指符号。

ADVANCE：是过程，让 **IP** 指向下一个符号。

ERROR：是出错处理子程序。

例：文法 G 为：

$$S \rightarrow xAy$$

$$A \rightarrow **|*$$

将其改写为 $LL(1)$ 文法。
然后构造相应的递归子程序。

提取左公共因子，得文法 G' 为：

$$S \rightarrow xAy$$

$$A \rightarrow *A\Box$$

$$A\Box \rightarrow *|\epsilon$$

因为

$$\text{Select}(A\Box \rightarrow *) \cap \text{Select}(A\Box \rightarrow \epsilon) = \{*\} \cap \{y\} = \phi$$

所以文法 $G\Box$ 是 $LL(1)$ 文法。

对每个非终结符构造一个过程。

根据 $S \rightarrow xAy$

```
PROCEDURE S
BEGIN
  IF SYM='x' THEN
    BEGIN
      ADVANCE;
      A;
      IF SYM='y'
      THEN
        ADVANCE
      ELSE ERROR
      END
    ELSE ERROR
  END
END
```

文法 G 为:

$S \rightarrow xAy$

$A \rightarrow *A \mid$

$A \mid \rightarrow * \mid \varepsilon$

根据 $A \rightarrow *A'$

```
PROCEDURE A
BEGIN
  IF SYM='*'
  THEN
    BEGIN
      ADVANCE;
      A
    END
  ELSE
    ERROR
  END
```

关于 $A' \rightarrow * | \varepsilon$

对于 $A \rightarrow \varepsilon$,

简单处理为对 A' 不作任何推导,
即结束 A 过程。意味着匹配 ε 。

```
PROCEDURE A'  
  BEGIN  
    IF SYM = '*'  
  THEN ADVANCE  
  END
```

最后考虑产生式 $A \rightarrow B | D$

```
PROCEDURE A  
  BEGIN  
    IF  
  SYM  $\in$  SELECT( $A \rightarrow B$ )  
  THEN B  
    ELSE IF  
  SYM  $\in$  SELECT( $A \rightarrow D$ )  
  THEN D  
    ELSE ERROR  
  END
```

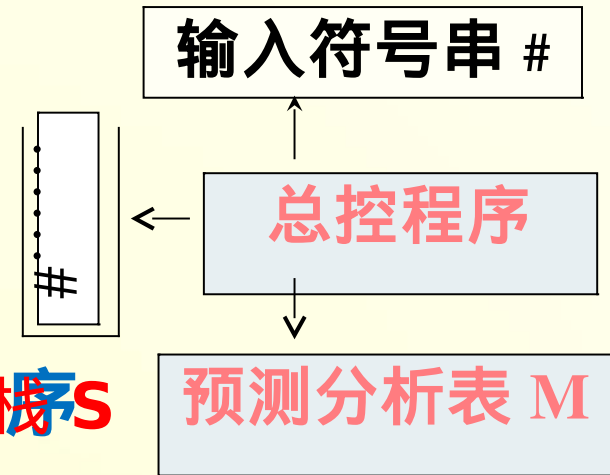
4.4.2 预测分析方法

1、预测分析器的组成 见右图：

三部分：一张预测分析表 **M**

一个符号栈 **S**

一个预测分析总控程序 **S**

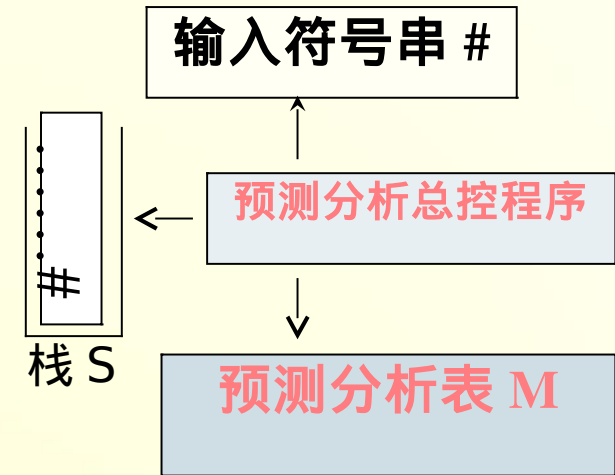


(1) 预测分析表 **M**

如下矩阵形式：矩阵 **M**

- 行标题用文法的非终结符表示。
- 列标题用文法的终结符号和 **#** 表示。
- 矩阵元素 $M[A, a]$ 的内容是产生式 $A \rightarrow \alpha$ (或 $\rightarrow \alpha$) 表明当对 A 进行推导，面临输入符号 a 时，应采用候选 α 进行推导。

出错处理标志 (即表中空白项) 表明 A 不该面临输入符号 a 。



如上例的预测分析表：

	x	y	$*$	$\#$
S	$S \rightarrow xAy$			
A			$A \rightarrow * A \square$	
A'		$A \square \rightarrow \epsilon$	$A \square \rightarrow *$	

(2) 符号栈

用于存放文法符号，栈顶为推导过程中句型尚未匹配部分的开头符号。

分析开始时，栈底先放一个 # ，然后放进文法开始符号，S
#

(3) 预测分析总控程序

总是按栈顶符号 x 和当前输入符号行事。

对于任何 (x, a) ，总控程序每次都执行下述三种可能动作之一：

(a) 若 $x=a='#'$ ，则宣布分析成功。

(b) 若 $x=a \neq '#'$ ，则把 x 从栈顶逐出，指针指向下一输入符号。

(c) 若 x 是一个非终结符，则查看分析表 M 。

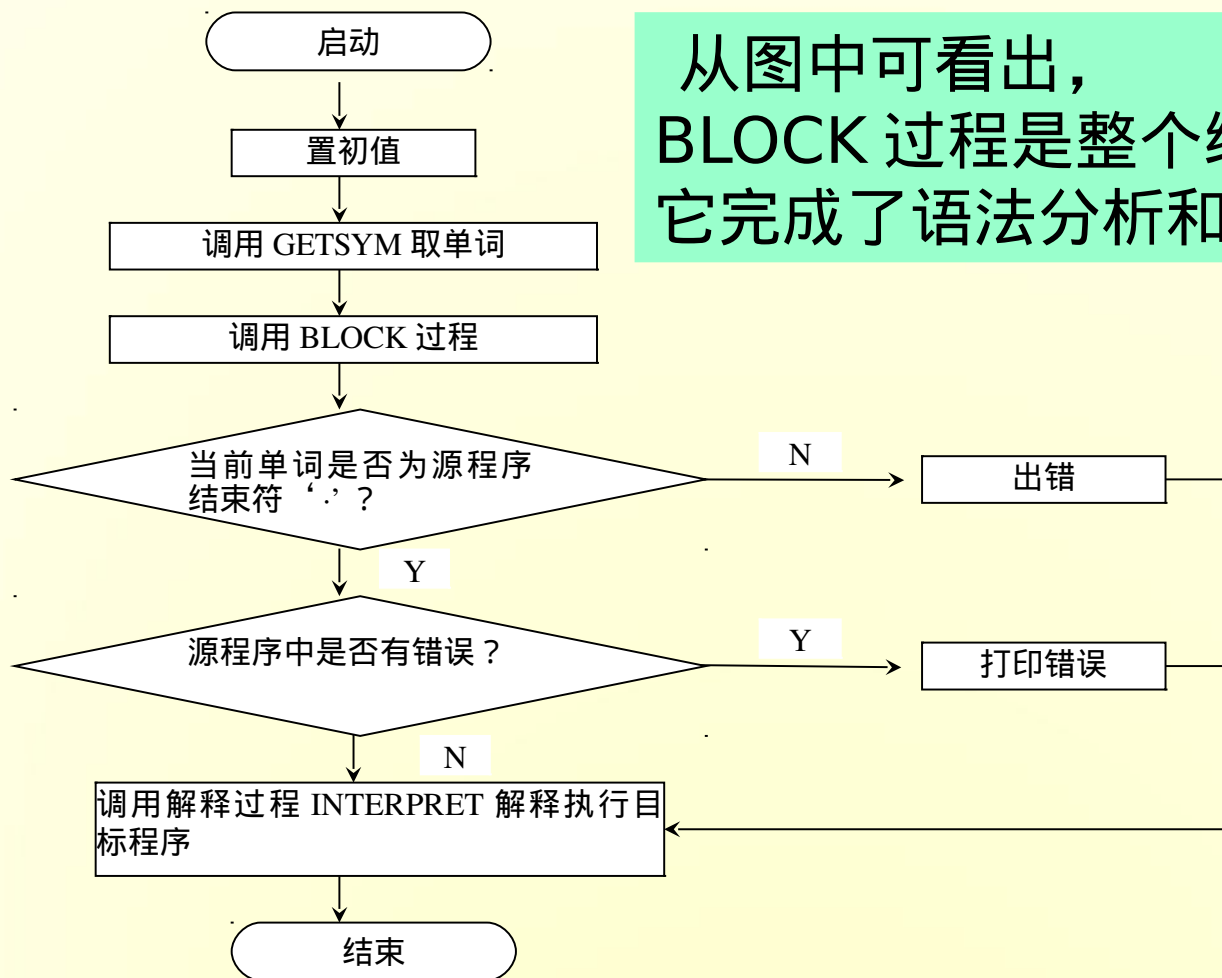
① 如果 $M[A, a]$ 中存放关于 X 的一个产生式，那么，首先把 X 顶出栈，然后把产生式右部符号串按反序一一推进栈。

② 如果 $M[A, a]$ 中存放“出错标志”，则调用出错处理程

4.5 实例：PL/0 编译程序的语法分析

PL/0 编译程序语法分析方法是确定的自顶向下分析，采用的是递归子程序法。

PL/0 编译程序总体流程图如下：



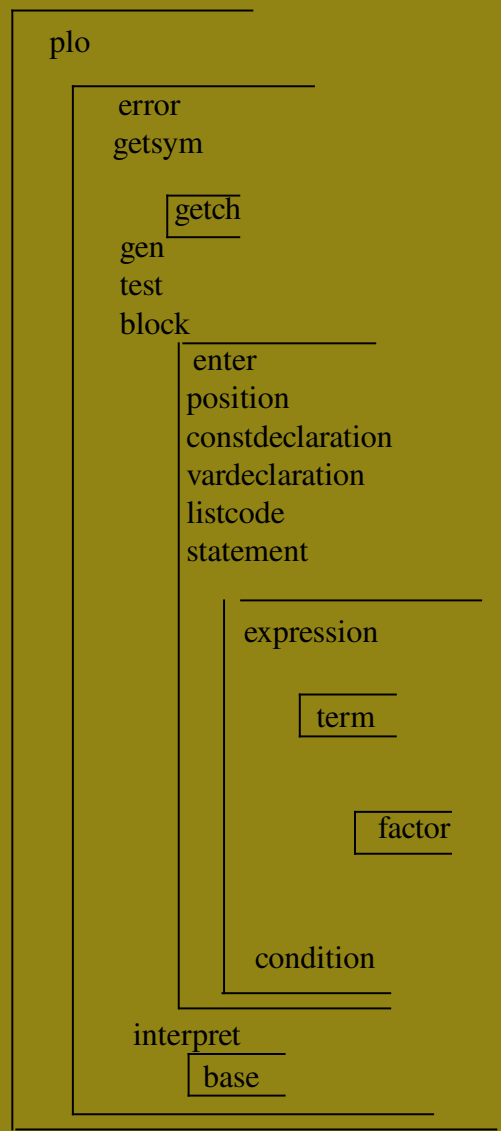
从图中可看出，
BLOCK 过程是整个编译过程的核心，
它完成了语法分析和代码生成的任务。

PL/O 编译程序是用 PASCAL 语言书写的，
整个编译程序是由 18 个嵌套及并列的过程或函数组成，如下表所示：

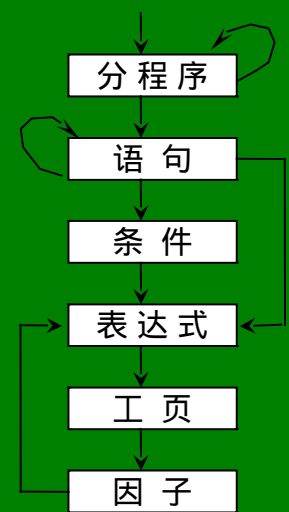
PL/O 编译程序的过程或函数的功能表

过程或函数名	简要功能说明
PL0	主程序
error	出错处理，打印出错位置和错误编码
getsym	词法分析，读取一个单词
getch	滤掉空格，读取一个字符
gen	生成目标代码，并送入目标程序区
test	测试当前单词符号是否合法
block	分程序分析处理过程
enter	登录名字表
position (函数)	查找标识符在名字表中位置
constdeclaration	常量定义处理
vardeclaration	变量说明处理
listcode	列出目标代码清单
statement	语句部分处理
expression	表达式处理
term	项处理
factor	因子处理
condition	条件处理
interpret	对目标代码的解释执行程序
base (函数)	通过静态链求出数据区的基地址

这些过程或函数的嵌套定义层次结构如图：



各分析子程序之间的调用关系如下图：



PL/0 语法调用关系图。

程 序

```
PROCEDURE statement (fsys:symset);
  var i, xc1, cx2: integer;
  procedure expression (fsys:symset);
    var addop:symbol;
  procedure term (fsys:symset);
    var mulop:symbol;
  procedure factor (fsys:symset);
    var i:integer;
  begin
    while sym in facbegsys do
      begin
        if sym=ident
          then
            getsym
          else
            if sym=number
              then getsym
              else
                if sym=lparen
                  then
                    begin
                      getsym;l
                    expression([rparen]+fsys);
                    if sym=rparen
                      then getsym
                      else error (22)
                    end;
                  end;
                end;
```

注 解

< 因子 >::=< 标识符 >| 无符号号整数 |< 表达式 >
facbegsys:=[ident, number, lparen]
ident 是标识符的种别
getsym 分析一个单词符号
number 是常数的种别
lparen 是左括号的种别
rparen 是右括号的种别

```
PROCEDURE statement (fsys:symset);  
var i, xc1, cx2: integer;  
  procedure expression (fsys:symset);  
    var addop:symbol;  
    procedure term (fsys:symset);  
      var mulop:symbol;  
      procedure factor (fsys:symset);  
        var i:integer;  
        begin  
          while sym in facbegsys do  
            begin  
              if sym=ident  
                then  
                  getsym  
              else  
                if sym=number  
                  then getsym  
                else  
                  if sym=lparen  
                    then  
                      begin  
                        getsym;l  
                        expression([rparen]+fsys);  
                        if sym=rparen  
                          then getsym  
                          else error (22)  
                      end;  
                    end;  
                  end;  
                end;  
            end;  
          end;  
        end;  
      end;  
    end;  
  end;  
end;
```

< 因子 > ::= < 标识符 > | 无符号号整数 | < 表达式 >

facbegsys:=[ident, number, lparen]

ident 是标识符的种别

getsym 分析一个单词符号

number 是常数的种别

lparen 是左括号的种别

rparen 是右括号的种别

程 序	注 解
<pre> begin (* term *) factor ([times, slash]+fsys); while sym in [times, slash] do begin getsym; factor (fsys+[times, slash]); end end begin (* expression *) if sym in [plus, minus] then begin getsym; term (fsys+[plus, minus]) end else term (fsys+[plus, minus]); while sym in [plus, minus] do begin getsym; term (fsys+[plus, minus]) end; end end </pre>	<p>< 项 > ::= < 因子 > { < 乘法运算符 > < 因子 > }</p> <p>< 表达式 > ::= [+ -] < 项 > { < 加法运算符 > < 项 > }</p>

程 序	注 解
<pre>procedure condition (fsys: symset); begin if sym=oddsym then begin getsym; expression (fsys); end else begin expression ([eq, neq, lss, leq, gtr, geq]+fsys), if not (symin [eq, neq, lss, leq, gtr, geq]) then error (20) else begin getsym; expression (fsys) end; end end end</pre>	<p>< 条件 > ::= < 表达式 > < 关系运算符 > < 表达式 > ODD < 表达式 ></p>

程 序

注 解

else

```

if sym=beginsym
then
begin
getsym;
statement ([semicolon, erdsym]+fsys);
while sym in [semicolon]+statbegsys do
begin
if sym=semicolon
then getsym
else error (10)
statement ([semicolon, endshm]+fsys)
end;
if sym=endsym
then getsym
else error (17)
end
    
```

< 复合语句 > ::= begin < 语句 > { ; 语句
> } end

程 序

```

else
  if sym=whilesym
    then
      begin
        getsym;
        condition ([dosym]+fsym);
        if sym=dosym
          then getsym
          else error (18);
        statement (fsys)
      end
    end
  end (* statement *)

```

注 解

< 当型循环语句 > ::= WHILE < 条件 > DO < 语句 >

本章要点

确定的自顶向下分析思想

LL (1) 文法的定义

计算 FIRST 集、 FOLLOW 集和 SELECT 集。

提取文法的左公共因子、消除文法左递归。

递归子程序法

预测分析法

第 4 章 自顶向下语法分析方法的综合例题

已知文法 G 为：

$S \rightarrow SbB \mid aB$

$B \rightarrow Ab \mid e$

$A \rightarrow a \mid \varepsilon$

要求：

1. 求消去左递归后的文法 G'
2. 求 G' 的所有非终结符的 FIRST、FOLLOW 和产生式的 SELECT
3. 构造 G' 的预测分析表
4. 根据 G' 的预测分析表分析输入串 aab, 写出分析

1. 求消去左递归后的文法 G'

G' 为: $S \rightarrow aBS'$
 $S' \rightarrow bBS' \mid \varepsilon$
 $B \rightarrow Ab \mid e$
 $A \rightarrow a \mid \varepsilon$

文法 G 为:

$S \rightarrow SbB \mid aB$

$B \rightarrow Ab \mid e$

$A \rightarrow a \mid \varepsilon$

一般情况下, 直接左递归的形式为

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$

消除左递归后改写为:

$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_n A'$

$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_m A' \mid \varepsilon$

2. 求 G' 的所有非终结符的 FIRST 集、 FOLLOW 集和产生式的 SELECT 集

(1) 计算 G' 的 FIRST 集。

$\text{FIRST}(\alpha) = \{a \mid \alpha \Rightarrow a\beta, a \in VT, \alpha, \beta \in V^*\}$

若 $\alpha \Rightarrow \varepsilon$, 则规定 $\varepsilon \in \text{FIRST}(\alpha)$ 。

$$\text{FIRST}(S) = \{a\}$$

$$\text{FIRST}(S') = \{b, \varepsilon\}$$

$$\begin{aligned}\text{FIRST}(B) &= \text{FIRST}(Ab) \cup \{e\} \\ &= (\text{FIRST}(A) - \varepsilon) \cup \{b\} \cup \{e\} \\ &= \{a, b, e\}\end{aligned}$$

$$\text{FIRST}(A) = \{a, \varepsilon\}$$

G' :

$$S \rightarrow aBS'$$

$$S' \rightarrow bBS \mid \varepsilon$$

$$B \rightarrow Ab \mid e$$

$$A \rightarrow a \mid \varepsilon$$

(2) 计算 G' 的所有非终结符的 FOLLOW 集

$\text{FOLLOW}(A) = \{a | S^* \Rightarrow \dots Aa \dots, a \in VT\}$

若有 $S \Rightarrow \dots A$, 则规定 $\# \in \text{FOLLOW}(A)$

计算 FOLLOW 集

(a) 设 S 为文法的开始符号, 则

$\# \in \text{FOLLOW}(S)$

(b) 若有 $A \rightarrow \alpha B \beta$, 则将 $\text{FIRST}(\beta) - \{\epsilon\}$
加入到 FOLLOW

(B) 中,

如果其中 $\beta \Rightarrow \epsilon$, 则将 $\text{FOLLOW}(A)$

$\text{FOLLOW}(S) = \{\#\}$ 加入到 FOLLOW

$\text{FOLLOW}(S') = \text{FOLLOW}(S) = \{\#\}$

$\text{FOLLOW}(B) = (\text{FIRST}(S') - \epsilon)$

$\cup \text{FOLLOW}(S)$

$\cup \text{FOLLOW}(S')$

$= \{b, \#\}$

$\text{FOLLOW}(A) = \{b\}$

$G' :$

$S \rightarrow aBS'$

$S' \rightarrow bBS' \mid \epsilon$

$B \rightarrow Ab \mid e$

$A \rightarrow a \mid \epsilon$

$\text{FIRST}(S) = \{a\}$

$\text{FIRST}(S') = \{b, \epsilon\}$

$\text{FIRST}(B) = \{a, b, e\}$

$\text{FIRST}(A) = \{a, \epsilon\}$

(3) 计算产生式的 SELECT 集

定义: $\text{SELECT} (A \rightarrow \alpha) = \text{FIRST} (\alpha)$ 其中 $\alpha^* \Rightarrow \varepsilon$

若 $\alpha^* \Rightarrow \varepsilon$, 则 $\text{SELECT} (A \rightarrow \alpha) = \{\text{FIRST} (\alpha) \setminus \varepsilon\}$

$\cup \text{FOLLOW}(A)$

$\text{Select} (S \rightarrow aBS') = \{a\}$

$\text{Select} (S' \rightarrow bBS') = \{b\}$

$\text{Select} (S' \rightarrow \varepsilon) = (\text{FIRST} (\varepsilon) - \varepsilon)$

$\cup \text{FOLLOW} (S') = \{\#\}$

$\text{Select} (B \rightarrow Ab) = \{a, b\}$

$\text{Select} (B \rightarrow e) = \{e\}$

$\text{Select} (A \rightarrow a) = \{a\}$

$\text{Select} (A \rightarrow \varepsilon) = (\text{FIRST} (\varepsilon) - \varepsilon)$

$\cup \text{FOLLOW} (A) = \{b\}$

$G' :$

$S \rightarrow aBS'$

$S' \rightarrow bBS' \mid \varepsilon$

$B \rightarrow Ab \mid e$

$A \rightarrow a \mid \varepsilon$

$\text{FIRST} (S) = \{a\}$

$\text{FIRST} (S') = \{b, \varepsilon\}$

$\text{FIRST} (B) = \{a, b, e\}$

$\text{FIRST} (A) = \{a, \varepsilon\}$

$\text{FOLLOW} (S) = \{\#\}$

$\text{FOLLOW} (S') = \{\#\}$

$\text{FOLLOW} (B) = \{b, \#\}$

$\text{FOLLOW} (A) = \{b\}$

3. 构造 G' 的预测分析表

G' :

$S \rightarrow aBS'$

$S' \rightarrow bBS \mid \epsilon$

$B \rightarrow Ab \mid e$

$A \rightarrow a \mid \epsilon$

$\text{Select}(S \rightarrow aBS') = \{a\}$

$\text{Select}(S' \rightarrow bBS') = \{b\}$

$\text{Select}(S' \rightarrow \epsilon) = \{\#\}$

$\text{Select}(B \rightarrow Ab) = \{a, b\}$

$\text{Select}(B \rightarrow e) = \{e\}$

$\text{Select}(A \rightarrow a) = \{a\}$

$\text{Select}(A \rightarrow \epsilon) = \{b\}$

输入 符状态	a	b	e	#
S	$S \rightarrow aBS'$			
S'		$S' \rightarrow bBS'$		$S' \rightarrow \epsilon$
B	$B \rightarrow Ab$	$B \rightarrow Ab$	$B \rightarrow e$	
A	$A \rightarrow a$	$A \rightarrow \epsilon$		

4. 根据 G' 的预测分析表分析输入串 aab，写出分析步骤

分析栈	输入串	所用产生式
#S	aab#	$S \rightarrow aBS'$
#S'Ba	aab#	a 匹配
#S'B	ab#	$B \rightarrow Ab$
#S'bA	ab#	$A \rightarrow a$
#S'ba	ab#	a 匹配
#S'b	b#	b 匹配
#S'	#	$S' \rightarrow \epsilon$
#	#	接受

	a	b	e	#
S	$S \rightarrow aBS'$			
S'		$S' \rightarrow bBS'$		$S' \rightarrow \epsilon$
B	$B \rightarrow Ab$	$B \rightarrow Ab$	$B \rightarrow e$	
A	$A \rightarrow a$	$A \rightarrow \epsilon$		

