第四章 自顶向下语法分析方法

4.3 某些非LL(1)文法到LL(1)文法的等价变换

4.5 LL(1)分析的实现

广东工业大学计算机学院

练习

- 练习
- S::=aAcB|Bc
- A::=AaB|c
- B::=bBcA|b|
- 求此文法

解：

FIRST(S)=FIRST(aAcB)∪FIRST(Bd)

={a}∪{b,d}

={a,b,d}

FIRST(A)=FIRST(AaB)∪FIRST(c)

={c}∪{c}

={c}

FIRST(B)=FIRST(bBcA)∪FIRST(b)

∪FIRST(ε)

={b}∪{b}∪{ε}

={b, ε}

练习2

Logo

■练习　已知文法G[S]:

S::=A

A::=BA′

A′::=iBA′| ε

B::=CB′

B′::=+CB′| ε

C::=)A*|(

求FOLLOW(C)。

解:

FOLLOW(C)=(FIRST(B′)-{ε})

∪FOLLOW(B)

={+,i,*,#}

本课内容

Logo

- **4.3 某些非LL(1)文法到LL(1)文法的等价变换**
  - 提取左公共因子
  - 消除左递归
- **4.5 LL(1)分析的实现**
  - 递归子程序法
  - 预测分析方法

非LL(1)文法的等价变换

Logo

- 一个上下无关文法是LL(1)文法 ⇔ 对每个非终结符A的任意两个不同产生式$A \to \alpha$和$A \to \beta$，满足：

$$SELECT(A \to \alpha) \cap SELECT(A \to \beta) = \varnothing$$

- 其中α、β不能同时$\overset{*}{\Rightarrow}$ ε。

- 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)左递归： $A \to Ab$　　$A \to a$
- (2)左公共因子： $S \to aSb|aS$

- 因而，我们设法消除文法中的左递归，提取左公共因子对文法进行等价变换，在某些特殊情况下可能使其变为LL(1)文法。

1. 提取左公共因子

Logo

- 如果文法中含有形如$A\to\alpha\beta|\alpha\gamma$的产生式，且$SELECT(A\to\alpha\beta)\cap SELECT(A\to\alpha\gamma)\neq\varnothing$，则此文法不满足LL(1)文法的充分必要条件。
- 现将$A\to\alpha\beta|\alpha\gamma$进行等价变换为：

$$A\to\alpha(\beta|\gamma)$$

- 其中‘(’，‘)’为元符号。进而引入新非终结符B替换之：

$$A\to\alpha B \qquad B\to\beta|\gamma$$

- 上述方法可以表述为如下的一般形式：
- 对 $A\to\alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n$，提取左公共因子后变为：
- $A\to\alpha(\beta_1|\beta_2|\ldots|\beta_n)$，再引进非终结符B，变为：

$$A\to\alpha B \qquad B\to\beta_1|\beta_2|\ldots|\beta_n$$

- 若在β_i、β_j、β_k … 中仍含有左公共因子时可再次提取，直到引进新非终结符的产生式再无左公共因子为止。

提取左公共因子举例1

Logo

- 若文法G1的产生式为： (1) S→aSb (2) S→aS (3) S→ε
- 请提取文法中的左公共因子。
- 解：对产生式(1), (2)提取左公共因子，文法变为：
  (1) S→ aS(b|ε) (3) S→ε
- 进一步变换为文法G'1：
  (1) S→aSA (3) S →ε (2) A→b (4) A→ε

① 若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则SELECT(A → α) = FIRST(α)。
② 若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则SELECT(A → α) = (FIRST(α) – {ε}) ∪FOLLOW(A)。

- 问题： G'1是不是LL(1)文法？
- SELECT(S→aSA) = {a}
- SELECT(S→ ε) = ??? (FIRST(ε) – {ε}) ∪ FOLLOW(S)
- = {#} ∪(FIRST(A) - {ε}) = {#, b} /* 对S→aSA */

- (a) S是文法开始符，则{#} ∈ FOLLOW(S)
- (b) 对A→αBβ，则FIRST(β)中的非空元素∈ FOLLOW(B)
- 另外，如果$\beta \overset{*}{\Rightarrow} \varepsilon$则FOLLOW(A) ∈ FOLLOW(B)

提取左公共因子举例1

Logo

- 若文法G1的产生式为： (1) S→aSb (2) S→aS (3) S→ε
- 请提取文法中的左公共因子。
- 解：对产生式(1), (2)提取左公共因子，文法变为：
  (1) S→ aS(b|ε) (3) S→ε
- 进一步变换为文法G'1：
  (1) S→aSA (3) S →ε (2) A→b (4) A→ε
- 问题： G'1是不是LL(1)文法？ 不是LL(1)
- = {b}
- SELECT(A→ ε) = ?(FIRST(ε) – {ε}) ∪ FOLLOW(A)
- = FOLLOW(S) = {#, b} /* 对S→aSA */
- SELECT(A→b) ∩ SELECT(A→ ε) = {b}

① 若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则 SELECT(A → α) = FIRST(α)。

② 若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则 SELECT(A → α) = (FIRST(α) – {ε}) ∪FOLLOW(A)。

- (a) S是文法开始符，则{#} ∈ FOLLOW(S)
- (b) 对A→αBβ，则FIRST(β)中的非空元素∈ FOLLOW(B)
- 另外，如果β⇒ε则FOLLOW(A) ∈ FOLLOW(B)

提取左公共因子举例2

Logo

*

- 若文法G2的产生式为:
  (1) A→ad　(2) A→Bc　(3) B→aA　(4) B→bB
- 请提取文法中的隐式左公共因子。
- 解：产生式(2)的右部以非终结符开始，因此左公共因子可能是隐式的。
- 这种情况下对右部以非终结符B开始的产生式，用B为左部且右部以终结符开始的产生式进行相应替换。
- ① 对文法G2分别用(3)、(4)的右部替换(2)中的B，可得：
  (1) A → ad　(2) A → aAc　(3) A → bBc
  (4) B → aA　(5) B → bB
- ② 提取产生式(1)、(2)的左公共因子得文法G'2：
- (1) A→a(d|Ac)　(2) A→bBc　(3) B→aA　(4) B→bB

提取左公共因子举例2(续)

Logo

*

- 若文法G2的产生式为：
  (1) A→ad　(2) A→Bc　(3) B→aA　(4) B→bB
- 请提取文法中的隐式左公共因子。

- 解：② 提取产生式(1)、(2)的左公共因子得：
  A→a(d|Ac)　A→bBc　B→aA　B→bB
- ③引进新非终结符C，去掉'('，')'后得G'2为：
  (1) A→aC　(2) A→bBc　(3) C→d
  (4) C→Ac　(5) B→aA　(6) B→bB

- 问题：　G'2是不是LL(1)文法？
- AnSwer：是

提取左公共因子举例3

Logo

- 若文法G3的产生式为：
  (1) S→aSd　(2) S→Ac　(3) A→aS　(4) A→b
- 请提取文法中的隐式左公共因子。
- 解：①用产生式(3)、(4)中右部替换产生式(2)中右部的A，文法变为：
  (1) S→aSd　(2) S→aSc　(3) S→bc
  (4) A→aS　(5) A→b

提取左公共因子可能会使某些产生式变成无用产生式

- ② 对(1)、(2)提取左公共因子得：S → aS(d|c)
- ③ 引入新非终结符B后变为：
  (1) S→aSB　(2) S→bc　(3) B→d|c
  (4) A→aS　(5) A→b
- 注意：原文法G3中非终结符A变成不可到达的符号，产生式(4)、(5)也就变为无用产生式，所以应删除。

提取左公共因子举例4

Logo

有些文法可能不能在有限步骤内完全提取左公共因子。

- 若文法**G4**的产生式为：
  (1) S→Ap|Bq　(2) A→aAp|d　(3) B→aBq|e
- 请提取文法中的隐式左公共因子。
- 解：① 用**(2)**、**(3)**产生式的右部替换**(1)**中产生式的**A**、**B**使文法变为：
  (1) S→aApp|aBqq　(2) S→dp|eq　(3) A→aAp|d　(4) B→aBq|e
- ② 对**(1)**提取左公共因子得：　**S→a(App|Bqq)**
- ③ 再引入新非终符**C**结果得等价文法为：
  (1) S→aC　(2) S→dp|eq　(3) C→App|Bqq
  (4) A→aAp|d　(5) B→aBq|e
- 注意：用**(4)**、**(5)**产生式的右部替换**(3)**中右部的**A**、**B**再提取左公共因子时，只能使文法的产生式无限增加下去。

提取左公共因子讨论

Logo

- 由上面所举例子可以说明以下问题：
- **(1)** 不一定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，只解决了相同左部的产生式其右部的**FIRST**集不相交问题。
- **(3)** 当改写后的文法不含空产生式，且无左递归时，则改写后的文法是**LL(1)**文法，否则还需用**LL(1)**文法的判别方式进行判断才能确定是否为**LL(1)**文法。

本课内容

Logo

- **4.3 某些非LL(1)文法到LL(1)文法的等价变换**
  - 提取左公共因子
  - 消除左递归
- **4.5 LL(1)分析的实现**

消除左递归

Logo

- 设一个文法含有下列形式的产生式：
  1) $A \rightarrow A\beta$ $A \in V_N$，$\beta \in V^*$
  2) $A \rightarrow B\beta$
     $B \rightarrow A\alpha$ $A,B \in V_N$, $\alpha,\beta \in V^*$
- 可称含有1)中产生式的文法为含有直接左递归的产生式。
- 含2)中产生式的文法有$A \overset{+}{\Rightarrow} A \ldots$ 则称文法中含有间接左递归的产生式。
- 只要含有1)或含有2)的产生式或二者皆有，则均认为文法是左递归的。

左递归文法不能自顶向下分析

Logo

- 一个文法是左递归时不能采用自顶向下分析法。
- 例：文法**G5**含有直接左递归：
  S→Sa　S→b
- 所能产生的语言**L = {ba^n | n≥0}**，对输入串**baaaa#**是该语言的句子，但用自顶向下分析时应如何推导？
- 可看出当输入符为**b**时，为与**b**匹配则应选用**S→b**来推导，但这样就推不出后边部分**aaaa**。
- 而若用**S→Sa**推导则无法确定到什么时候才用**S→b**替换。
- 另一方面，用递归子程序法时，在处理**S**的过程中，没有对当前输入符号匹配就又进入递归调用处理**S**的过程，这样就会造成死循环。
- 注意：对含有间接左递归的文法同样可有上述现象。

消除左递归

Logo

- 结论：有左递归的文法绝对不是**LL(1)**文法。

- 为了使某些含有左递归的文法经过等价变换消除左递归后可能变为**LL(1)**文法，可采取：
- **1)** 消除直接左递归，把直接左递归改写为右递归。

- 例如：文法**G5[S]**：
  S → Sa　S → b
- 可改写**G5'[S]**为：
  S → bS'　S'→ aS'|ε

- 改写后的文法和原文法产生的语言句子集都为：　{ba^n | **n≥0**}　。
- 同学们自己验证改写后的文法**G5'**为**LL(1)**文法。

消除左递归——消除直接左递归

Logo

- 一般情况下，假定关于A的全部产生式是：

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

- 其中，$\alpha_i(1 \le i \le m)$不等于ε，$\beta_j(1 \le j \le n)$不以A开头，消除直接左递归后改写为：

  $A \rightarrow \beta_1 B|\beta_2 B|\ldots|\beta_n B$

  $B \rightarrow \alpha_1 B|\alpha_2 B|\ldots|\alpha_m B|\varepsilon$

- 例如：文法G5：

  $S \rightarrow Sa \quad S \rightarrow b$

- 可改写G5’为：

  $S \rightarrow bS' \quad S' \rightarrow aS'|\varepsilon$

消除左递归——消除间接左递归

Logo

- **2)** 消除间接左递归。
- 做法：需先将间接左递归变为直接左递归，然后再按**1)**消除直接左递归。
- 例如有文法**G6**：
  (1) A→aB (2) A→Bb (3) B→Ac (4) B→d
- 消除文法中的左递归，并检验改写后的文法是否为**LL(1)**文法。

- 解：用产生式**(1)**、**(2)**的右部代替产生式**(3)**中的非终结符**A**得到左部为**B**的产生式为：
  (3) B→aBc (4) B→Bbc (5) B→d
- 消除左递归后得：**B→(aBc|d)B′ B′→bcB′|ε**
- 再把原来其余的产生式**A→aB**，**A→Bb**加入，最终文法为：
  (1) A→aB (2) A→Bb
- **(3) B→(aBc|d)B′ (4) B′→bcB′|ε**
- 可以检验改写后的文法不是**LL(1)**文法。

本课内容

Logo

- **4.3 某些非LL(1)文法到LL(1)文法的等价变换**
  - 提取左公共因子
  - 消除左递归

- **4.5 LL（1）分析的实现**
  - 递归子程序法
  - 预测分析方法

递归子程序法简介

Logo

- 递归子程序法是比较简单并且易于构造的一种语法分析方法。**PL/0**编译程序的语法分析部分就是采用递归子程序法。

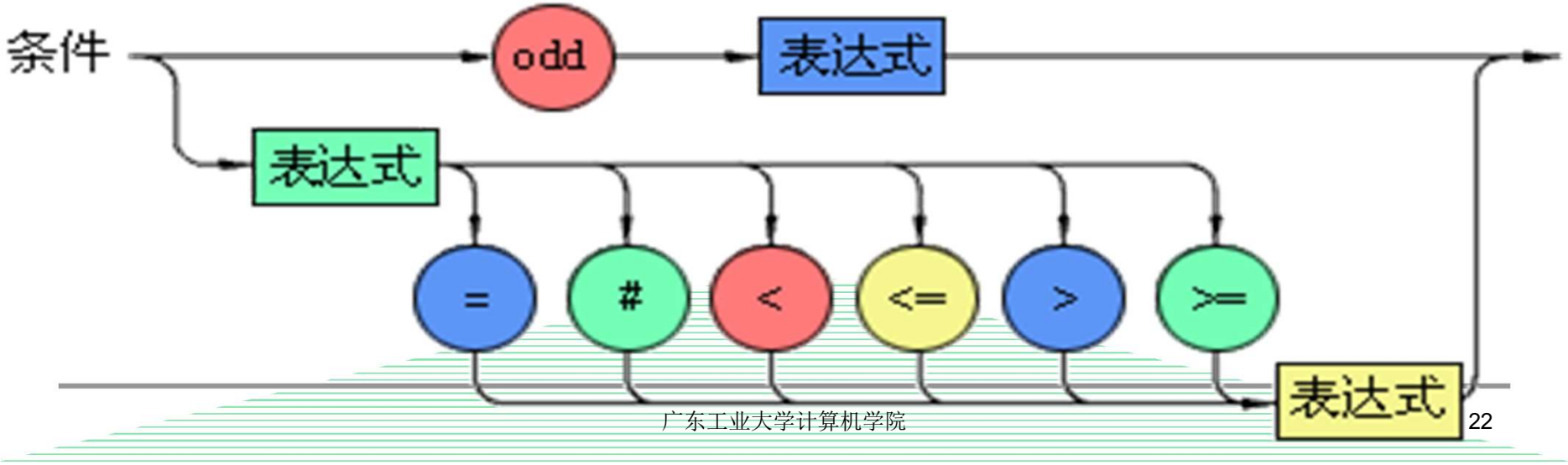
- 实现思想：对应文法中每个非终结符编写一个递归过程。每个过程识别由该非终结符推出的串。

- 实现过程：从读入第一个单词开始，由非终结符**<程序>(**即开始符**)**出发，沿语法描述图箭头所指的方向进行分析。

递归子程序法的步骤

Logo

- 当遇到描述图中的终结符时，则将当前读入的单词与图中的终结符匹配。若匹配，则只需：
- (1) 执行相应的语义程序(即翻译程序)
- (2) 再读取下一个单词继续分析
- (3) 遇到分支点时，将当前的单词与分支点上多个终结符逐个比较，如都不匹配，则可能是进入下一个非终结符语法单位，或者是出错。



递归下降分析算法

Logo

它由一个总控程序和若干个非终结符号的分析子程序**P(U)**构成。其中若有**n**个非终结符号就有**n**个分析子程序。

对满足条件的文法按如下方法构造相应的语法分析子程序：

- ①对于每个非终结符号**U**,编写一个相应的子程序**P(U)**。
- ②对于产生式**U::=α1|α2|…|αn, αi∈V**， **P(U)**按如下方法构造：

if ch∈**SELECT(A→α1)** then P(α1)
else if ch∈**SELECT(A→α2)** then P(α2)
else if ch∈**SELECT(A→α3)** then P(α3)
…
else if ch∈**SELECT(A→αn)** then P(αn)
else error
(也可以类似书本P86-87,用switch case结构）

- ③对于产生式U::=α1α2…αn，P(U) 按如下方法构造：

```
begin
 P(α1);
 P(α2);
  …;
 P(αn)
end
```

■说明：如果$\alpha_i \in V_n$，则$P(\alpha_i)$就代表调用处理α_i的子程序；如果$\alpha_i \in V_t$，则$P(\alpha_i)$为形如下述语句的一段程序：

if ch=α_i then READ(ch) else error

即如果当前文法中的符号与输入符号匹配，则继续读入输入符号串的下一个符号到ch中，否则出错。

PL/0语言的EBNF

Logo

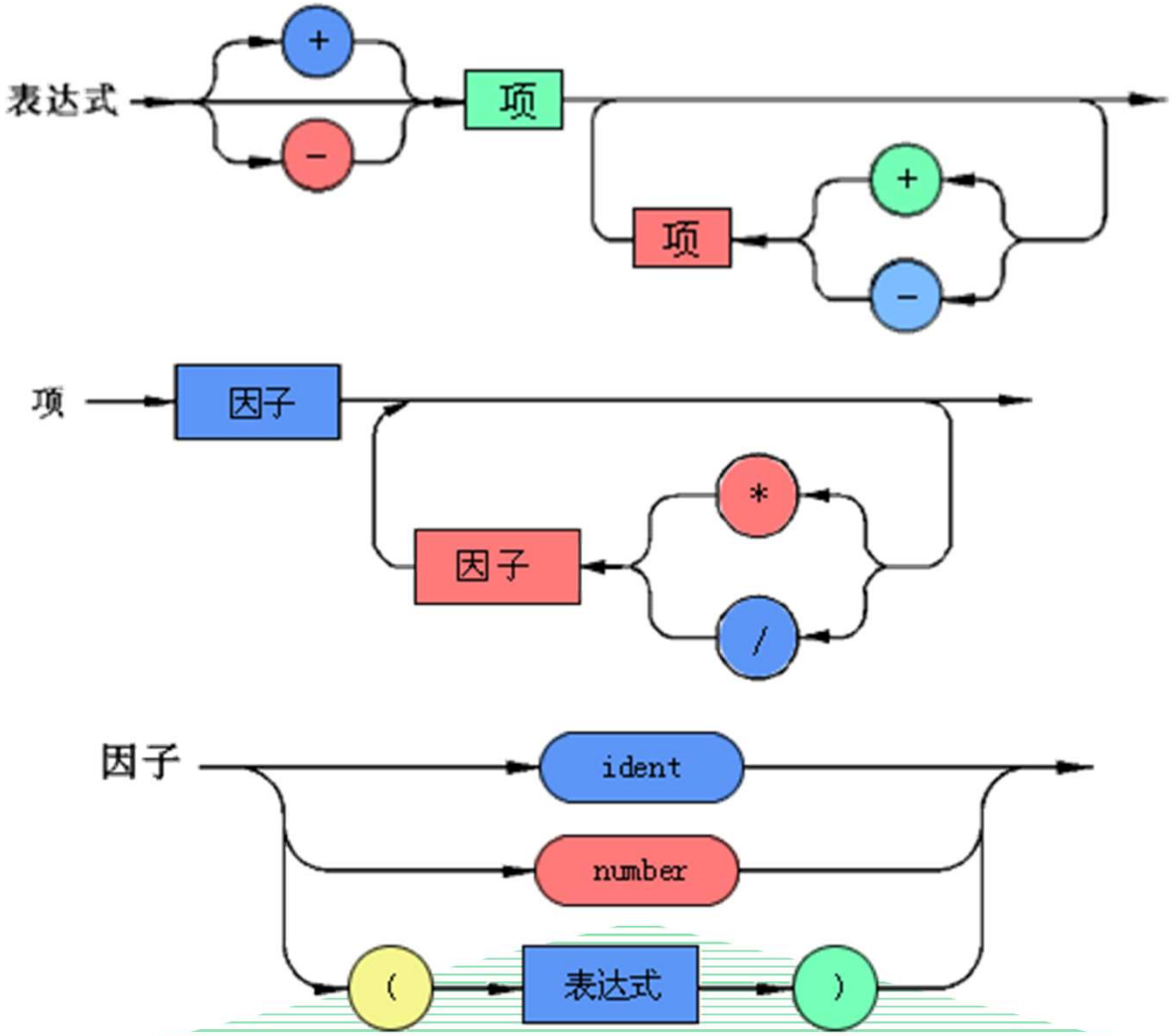
- 〈程序〉::=〈分程序〉.
- 〈分程序〉::=[〈常量说明部分〉][〈变量说明部 分〉][〈过程说明部分〉]〈语句〉
- 〈常量说明部分〉::=CONST〈常量定义部分〉{，〈常量 定义〉};
- 〈变量说明部分〉::=VAR〈标识符〉{，〈标识符〉};
- 〈过程说明部分〉::= PROCEDURE 〈标识符〉〈分程序〉 {;〈过程说明部分〉};

- 〈语句〉::= 〈标识符〉：=〈表达式〉 |IF 〈条件〉 then 〈语句〉 |CALL...|READ...|BEGIN 〈语句〉{; 〈语句〉} END|WHILE...|...

书P90

部分语法描述图

Logo



PL/0语言递归子程序举例

Logo

- 表达式的EBNF（代码见书本P91）

〈表达式〉::=[+|-]〈项〉{(+|-) 〈项〉}

```
int expression(bool* fsys, int* ptx, int lev)
{
    if(sym==plus || sym==minus)  /* 开头的正负号*/
    {                    …
        getsym;
        …
        term(nxtlev, ptx, lev);    /* 处理项 */
        …
    }
    else    /* 此时表达式被看作项的加减 */
    {                    …
        term(nxtlev, ptx, lev);    /* 处理项 */
    }
    while (sym==plus || sym==minus)
    {
        …
        getsym;
        …
        term(nxtlev, ptx, lev);    /* 处理项 */
        …
  }
  return 0;
}
```

PL/0语言递归子程序举例

Logo

- 〈项〉::=〈因子〉{（*|/）〈因子〉}

```
int term(bool* fsys, int* ptx, int lev)
{   …
    factor (nxtlev, ptx, lev);/* 处理因子 */
    while(sym==times || sym==slash)
   {            …
            getsym;
            factor (nxtlev, ptx, lev);
            …
     }
  return  0；
}
```

PL/0语言递归子程序举例

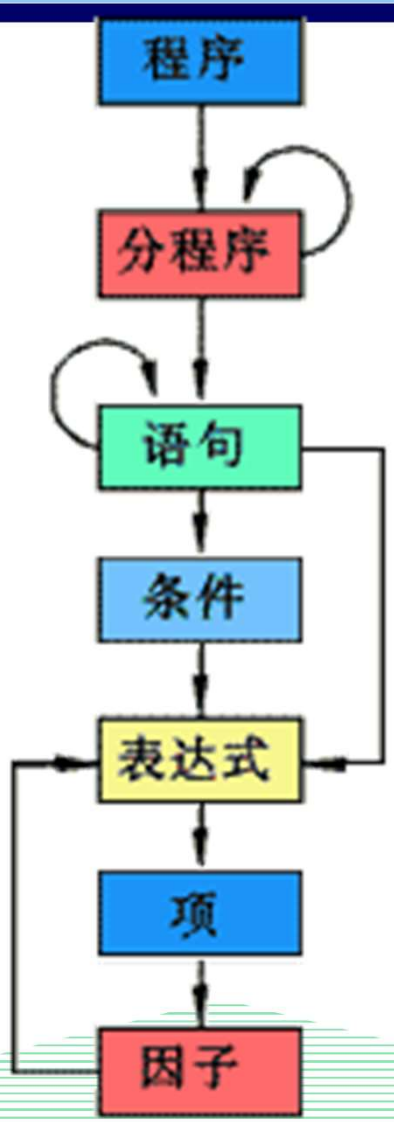
Logo

- 〈因子〉::=〈标识符〉|〈无符号整数〉|‘（’〈表达式〉‘）’

```
int factor(bool* fsys, int* ptx, int lev)
{   if(sym == ident)          /* 因子为常量或变量 */
      getsym;
    else
      {if(sym == number) /* 因子为数 */
        getsym    else
          {if (sym == lparen)          /* 因子为表达式 */
                    { expression(nxtlev, ptx, lev);
                      if (sym == rparen)
                        getsym;
                        else  error(22);          /* 缺少右括号 */
                    }
         else  error( )
          }
}
```

PL/0语言语法调用关系图

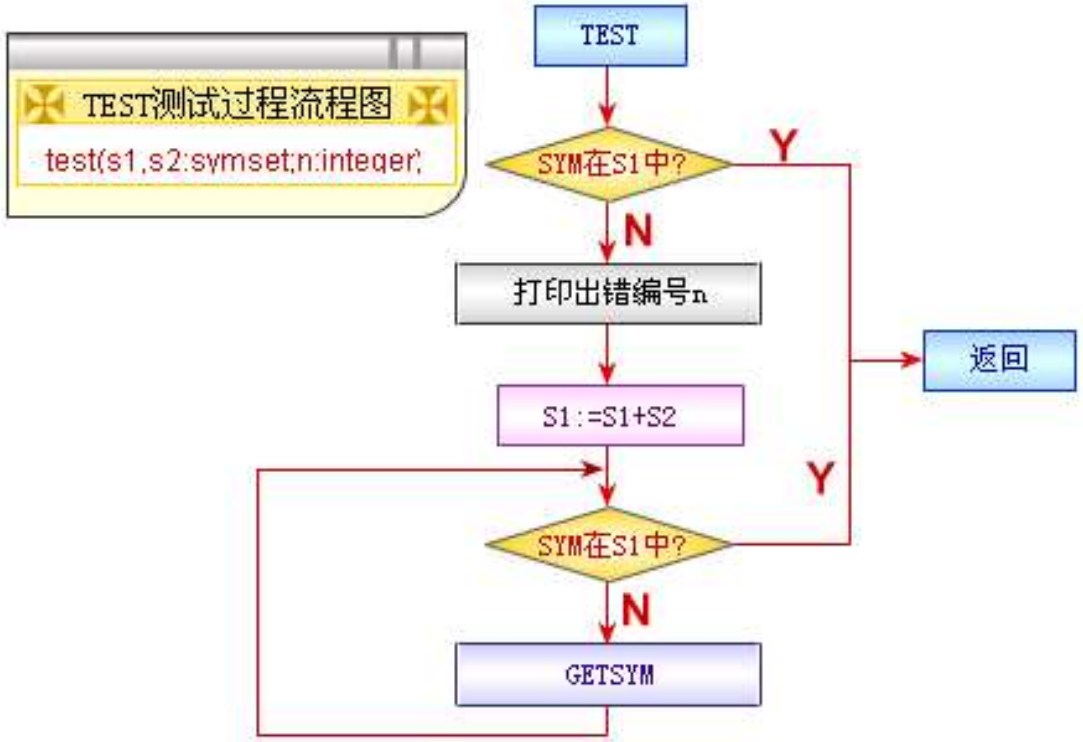
Logo



PL/0语言的语法错误处理（4.6.3）

Logo

- 当语法分析进入以某些关键字(保留字)或终结符集合为开始符的语法单元时，通常在它的入口和出口处，调用一个测试程序**TEST** 。（**S1,S2 P98)**



- 在进入某个语法单位时，调用**TEST**，检查当前符号是否属于该语法单位的开始符号集合。若不属于，则滤去开始符号和后继符号集合外的所有符号。
- 在语法单位分析结束时，调用**TEST**，检查当前符号是否属于调用该语法单位时应有的后继符号集合。若不属于，则滤去后继符号和开始符号集合外的所有符号。
- 举例见书**P98-99**

递归子程序法小结

Logo

- 递归子程序法：对应文法中每个非终结符编写一个递归过程。每个过程识别由该非终结符推出的串。
- 当某非终结符的产生式有多个候选时，能够按**LL(1)**形式可唯一地确定选择某个候选进行推导。
- 递归子程序法的缺点：
- **(1)** 对文法要求高，必须满足**LL(1)**文法；
- **(2)** 递归调用多，所以速度慢占用空间多。

本课内容

Logo

- **4.3** 某些非**LL(1)**文法到**LL(1)**文法的等价变换
  - 提取左公共因子
  - 消除左递归

- **4.5 LL(1)**分析的实现
  - 递归子程序法
  - 预测分析方法

预测分析方法

Logo

- 预测分析方法是自顶向下分析的另一种方法，一个预测分析器是由三个部分组成。
  - 预测分析程序(总控程序)
  - 先进后出栈(stack)
  - 预测分析表
- 其中只有预测分析表与文法有关，而分析表又可用一个矩阵M(或称二维数组)表示。
- 矩阵的元素M[A, a]中，A表示非终结符，a为终结符或句子括号“#”。
- 矩阵元素M[A, a]中的元素是一条A为左部的产生式，表明当用非终结符A向下推导，且面临输入符a时，所应采取的候选产生式。
- 当M[A, a]中的元素为空时，表明以A为左部向下推导时遇到不该出现的符号，此时元素内容为转向出错处理的信息。

预测分析方法的步骤

Logo

- 现以以表达式文法为例构造预测分析表。表达式文法为：

  E→E+T|T　T→T*F|F　F→i|(E)

- 步骤1：判断文法是否为LL(1)文法。
- 由于文法中含有左递归，所以必须先消除左递归，使文法变为：

  E → TE′　E′→ +TE′|ε　T → FT′

  T′→ *FT′|ε　F → i|(E)

- ① 可推出ε的非终结符表X[ ]：

E	E′	T	T′	F
否	是	否	是	否

预测分析方法的步骤(续1)

Logo

- 现以以表达式文法为例构造预测分析表。表达式文法为：
  E→E+T|T　T→T*F|F　F→i|(E)

此文法是LL(1)文法

- 步骤1：② 计算各非终结符的FIRST集：
- FIRST(E) = {(, i}　FIRST(E′) = {+, ε}
  FIRST(T) = {(, i}　FIRST(T′) = {*, ε}　FIRST(F) = {(, i}
- ③ 计算各非终结符的FOLLOW集：
- FOLLOW(E) = {), #}　FOLLOW(E′) = {)，#}
  FOLLOW(T) = {+，)，#}　FOLLOW(T′) = {+，)，#}
  FOLLOW(F) = {*，+，)，#}
- ④ 各产生式的SELECT集合：
  SELECT(E→TE′) = {(, i}　SELECT(E′→+TE′) = {+}
  SELECT(E′→ε) = {), #}　SELECT(T→FT′) = {(, i}
  SELECT(T′→*FT′) = {*}　SELECT(T′→ε) = {+,), #}
  SELECT(F→(E)) = {(}　SELECT(F→i) = {i}

预测分析方法的步骤(续)2

Logo

- 现以以表达式文法为例构造预测分析表。表达式文法为：
  E→E+T|T　T→T*F|F　F→i|(E)
- 经过步骤1后的文法：
- SELECT(E→TE') = { (, i }; SELECT(E'→+TE') = { + }
  SELECT(E'→ε) = {), # } SELECT(T→FT') = { (, i }
  SELECT(T'→*FT') = { * } SELECT(T'→ε) = { +,), # }
  SELECT(F→(E)) = { (} SELECT(F→i) = { i }
- 步骤2：构造预测分析表M

E	i	+	*	(	)	#
E	→ TE′			→ TE′		
E′		→ +TE′			→ε	→ε
T	→FT′			→FT′		
T′		→ε	→*FT′		→ε	→ε
F	→i			→(E)		

举例：对符号串i+i*i#的分析过程

Logo

步骤	分析栈	剩余输入串	所用产生式
1	#E	i+i*i#	E→TE′
2	#E′ T	i+i*i#	T→FT′
3	#E′ T′ F	i+i*i#	F→i
4	#E′ T′ i	i+i*i#	i匹配
5	#E′ T′	+i*i#	T′ → ε
6	#E′	+i*i#	E′ →+TE′
7	#E′ T+	+i*i#	+匹配

E	i	+	*	(	)	#
E	→ TE′			→ TE′		
E′		→ +TE′			→ε	→ε
T	→FT′			→FT′		
T′		→ε	→*FT′		→ε	→ε
F	→i			→(E)		

举例：对符号串i+i*i#的分析过程(续1)

步骤	分析栈	剩余输入串	所用产生式
7	#E′ T+	+i*i #	+匹配
8	#E′ T	i*i #	T→FT′
9	#E′ T′ F	i*i #	F→i
10	#E′ T′ i	i*i #	i匹配
11	#E′ T′	*i #	T′ →*FT′
12	#E′ T′ F*	*i #	*匹配

E	i	+	*	(	)	#
E	→ TE′			→ TE′		
E′		→ +TE′			→ε	→ε
T	→FT′			→FT′		
T′		→ε	→*FT′		→ε	→ε
F	→i			→(E)		

举例：对符号串i+i*i#的分析过程(续2)

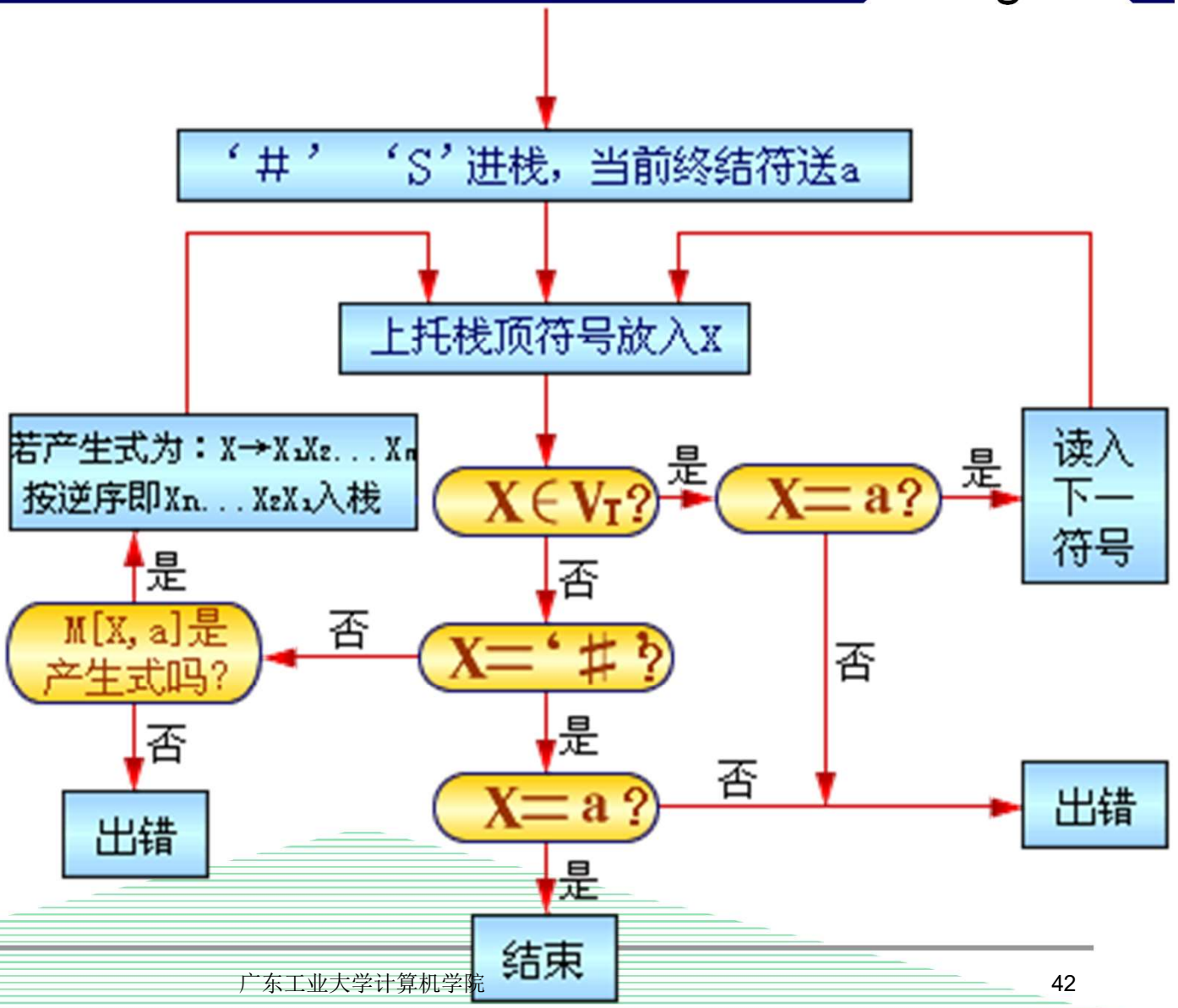
步骤	分析栈	剩余输入串	所用产生式
12	#E′ T′ F*	*i#	*匹配
13	#E′ T′ F	i#	F→i
14	#E′ T′ i	i#	i匹配
15	#E′ T′	#	T′ → ε
16	#E′	#	E′ → ε
17	#	#	接受

	i	+	*	(	)	#
E	→ TE′			→ TE′		
E′		→ +TE′			→ε	→ε
T	→FT′			→FT′		
T′		→ε	→*FT′		→ε	→ε
F	→i			→(E)		

预测分析程序的框图

Logo

- 说明：
- **#**：句子终结符
- **S**：文法开始符号
- **X**：存放当前栈顶符号的工作单元
- **a**：存放当前输入符号**a**的工作单元



(10分) 有一文法G：S→BA
A→BS |d
B→aA |bS|c
（1）证明它是LL（1）文法。
（2）构造它的预测分析表。

作业

Logo

- 写上作业日期
- 及时要作业！

主观题 10分

设置

Logo

附加题1：对文法G’ [S]:

S→a|∧|(T)　　T → SU　　U →,SU|ε

证明该文法是LL(1)的，然后构造该文法的预测分析表，并判断(a,a)#是不是该文法的句子。

正常使用主观题需2.0以上版本雨课堂

作答

主观题 10分

设置

Logo

已知文法G[S]：G[S]：S→aBc|bAB　　A→aAb|b

B→b|ε

其预测分析表如下所示。请给出符号串babb#的LL(1)分析过程。

	a	b	c	#
S	→aBc	→bAB		
A	→aAb	→b		
B		→b	→ε	→ε

（注：答案格式为　步骤 分析栈 剩余输入串　推导所用产生式或匹配）

正常使用主观题需2.0以上版本雨课堂

作答

主观题 10分

设置

Logo

对于一个文法若消除了左递归，提取了左公共因子后是否一定为LL(1)文法？试对下面的文法进行改写，并对改写后的文法进行判断。

（1）A→baB| ε
B→Abb|a

（2）S→Aa|b
A→SB
B→ab

正常使用主观题需2.0以上版本雨课堂

作答

附录

Logo

- 消除左递归算法

消除左递归的算法

Logo

- 消除文法中一切左递归，要求文法中不含回路，即无$A \overset{*}{\Rightarrow} A\ldots$的推导。
- 要求：文法中不包含形如$A \rightarrow A$的有害规则和空产生式。
- 算法步骤如下：
- (1) 把文法的所有非终结符排序，如$A_1, A_2, \ldots, A_n$
- (2) ① 从A_1开始消除左部为A_1的产生式的直接左递归。
- ② 然后把A_1的所有规则的右部逐渐替换$A_2 \rightarrow A_1\ldots$中的$A_1$，并消除左部为$A_2$的产生式的直接左递归。
- ③ 以同样方式，把$A_3 \rightarrow A_1\ldots$或者$A_3 \rightarrow A_2\ldots$中的$A_1$和$A_2$使用相应的产生代替，将消除$A_3 \ldots, A_n$的直接左递归。
- ④ 继而以同样方式， 消除$A_4\ldots, A_n$的直接左递归。
- (3) 去掉无用产生式。

消除左递归算法的伪码表示

Logo

- 把上述算法归结为：
- (1) 若非终结符的排序为A_1，A_2，…，A_n。
- (2)　　FOR i := 1 TO N DO
- 　　BEGIN
- 　　…… // 消除A_i中的一切直接左递归。
- 　　FOR j := 1 TO i-1 DO
  BEGIN
  若A_j的所有产生式为：
  $A_j \rightarrow \delta_1|\delta_2|\ldots|\delta_k$
  将形如$A_i \rightarrow A_j r$的产生式替换为：
  $A_i \rightarrow \delta_1 r|\delta_2 r|\ldots|\delta_k r$
  END
- 　　END
- (3) 最后删除无用产生式。

消除左递归的算法举例

Logo

- 例如文法的产生式为：
  (1) S→Qc|c　(2) Q→Rb|b　(3) R→Sa|a
- 按上述方法消除该文法的一切左递归。
- 解一：　① 将非终结符排序为**S、Q、R**
- ②　左部为**S**的产生式**(1)**无直接左递归，**(2)**中右部不含**S**，所以把**(1)**右部代入**(3)**得：　**(4) R→Qca|ca|a**
- 再将**(2)**的右部代入**(4)**得：　**(5) R→Rbca|bca|ca|a**
- 对**(5)**消除直接左递归得：
  (6) R→(bca|ca|a)R′　(7) R′→bcaR′|ε
- ③ 最终文法变为：
  (1) S→Qc|c　(2) Q→Rb|b
- **(6) R→(bca|ca|a)R′　(7) R′→bcaR′|ε**

消除左递归的算法举例(续)

Logo

- 例如文法的产生式为：
  (1) S→Qc|c　(2) Q→Rb|b　(3) R→Sa|a
- 按上述方法消除该文法的一切左递归。

- 解二：　① 如果若非终结符的排序为**R、Q、S**, 则把**(3)**代入**(2)**得：
  Q→Sab|ab|b
- ② 再将此代入**(1)**得：
  S→Sabc|abc|bc|c

当非终结符的排序不同时，最后结果的产生式形式不同，但它们是等价的。

- ③ 消除该产生式的左递归后，最终文法变为：
  S→(abc|bc|c)S′　　S′→abcS′|ε
  Q→Rb|b　　R→Sa|a
- ④ 由于**Q、R**为不可到达的非终结符，所以以**Q、R**为左部及包含**Q、R**的产生式应删除。