专题4\_算符优先语法分析

李若森 13281132 计科1301

# 理论传授

语法分析的设计方法和实现原理；算符优先文法、最左素短语、算符优先矩阵、优先函数的基本概念；算符优先文法句型最左素短语的确定；算符优先分析算法的实现。

# 目标任务

## 实验项目

实现算符优先分析算法，完成以下描述算术表达式的算符优先文法的算符优先分析过程。

G[E]:

E→E+T|E-T|T

T→T\*F|T/F|F

F→(E)|i

## 设计说明

终结符号i为用户定义的简单变量，即标识符的定义。加减乘除即运算符。

## 设计要求

1. 构造该算符优先文法的优先关系矩阵或优先函数；
2. 输入串应是词法分析的输出二元式序列，即某算术表达式“专题 1”的输出结果，输出为输入串是否为该文法定义的算术表达式的判断结果；
3. 算符优先分析程序应能发现输入串出错；
4. 设计两个测试用例（尽可能完备，正确和出错），并给出测试结果。

## 任务分析

重点解决算符优先矩阵的构造和算符优先算法的实现。

## 能力培养

深入理解理论对实践的指导作用；基本原理、实现技术和方法的正确运用。

# 实现过程

## OPG优先关系

设G是一OG（简单优先）文法，a,b∈Vt，U,V,W ∈Vn

1. a = b 当且仅当OG有

形如U→….ab….或U→….aVb….的规则

1. a < b 当且仅当OG有

形如U→….aW….的规则，而且W=+>b….或W=+>Vb…..

1. a > b 当且仅当OG有

形如U→….Wb….的规则，而且W=+> …. a或W=+>….aV

若a,b之间至多存在上述三种优先关系之一，OG为OPG（算符优先）文法。

## FIRSTVT集和LASTVT集

1. **FIRSTVT集**

* 两条原则

1. 若有规则U→b…或U→Vb…，则b ∈ FIRSTVT(U)
2. 若有规则U→V…且b ∈ FIRSTVT(V)，则b ∈ FIRSTVT(U)

* 过程描述

数据结构：STACK栈

布尔数组F(U,b) U∈Vn，b∈Vt

F(U,b) 真 b∈FIRSTVT(U)

假 b∉FIRSTVT(U)

* 算法分析

初始：F(U,b)初值(根据原则①)

F(U,b)为真的(U,b)对进STACK栈

循环：直至STACK空(根据原则②)

弹出栈顶元素，记(V,b)

对每一个形如U→V…的规则

若F(U,b) 为假，变为真，进STACK栈

若F(U,b) 为真，再循环

结果：FIRSTVT(U)={b∣F(U,b)=TRUE}

FIRSTVT(E)={ +, -, \*, /, (, i }

FIRSTVT(T)={ \*, /, (, i }

FIRSTVT(F)={ (, i }

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表3-1 | | | | | | | |
| b U | + | - | \* | / | ( | ) | i |
| E | T | T | T | T | T |  | T |
| T |  |  | T | T | T |  | T |
| F |  |  |  |  | T |  | T |

1. **LASTVT集**

* 两条原则

①若有规则U→…a或U→…aV，则a ∈ LASTVT (U)

②若有规则U→…V且a ∈ LASTVT (V)，则a ∈ LASTVT (U)

* 过程描述

数据结构：STACK栈

布尔数组F(U,a) U∈Vn，a∈Vt

F(U,a) 真 a∈LASTVT (U)

假 a∉LASTVT (U)

* 算法分析

初始：F(U,a)初值(根据原则①)

F(U,a)为真的(U,a)对进STACK栈

循环：直至STACK空(根据原则②)

弹出栈顶元素，记(V,a)

对每一个形如U→V…的规则

若F(U,a) 为假，变为真，进STACK栈

若F(U,a)为真，再循环

结果： LASTVT (U)={a∣F(U,a)=TRUE}

LASTVT(E)={ +, -, \*, /, ), i }

LASTVT(T)={ \*, /, ), i }

LASTVT(F)={ ), i }

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表3-2 | | | | | | | |
| a U | + | - | \* | / | ( | ) | i |
| E | T | T | T | T |  | T | T |
| T |  |  | T | T |  | T | T |
| F |  |  |  |  |  | T | T |

## 最左素短语

算符优先文法句型的一般形式：

#N1a1N2a2 …… NnanNn+1# ai∈Vt Ni∈Vn(可有可无)

最左素短语的确定：

一个OPG句型的最左素短语是满足以下条件的最左子串：

Njaj…..Niai Ni+1其中：aj-1 < aj

aj = aj+1 = … = ai-1 = ai

ai > ai+1

aj-1 < aj = aj+1 = … = ai-1 = ai > ai+1

所以，

|  |
| --- |
|  |

## 构造OPG优先矩阵算法

|  |
| --- |
|  |

根据构造OPG优先矩阵算法，可得如下优先关系：

+ FIRSTVT(T) LASTVT(E)  +

- FIRSTVT(T) LASTVT(E)  -

\* FIRSTVT(F) LASTVT(T)  \*

/ FIRSTVT(F) LASTVT(T)  /

( FIRSTVT(E) LASTVT(E)  ) ( )

# FIRSTVT(E) LASTVT(E)  #

所以，构造OPG优先矩阵如表3.3所示：

表3.3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | - | \* | / | ( | ) | i | # |
| + |  |  |  |  |  |  |  |  |
| - |  |  |  |  |  |  |  |  |
| \* |  |  |  |  |  |  |  |  |
| / |  |  |  |  |  |  |  |  |
| ( |  |  |  |  |  |  |  |  |
| ) |  |  |  |  |  |  |  |  |
| i |  |  |  |  |  |  |  |  |
| # |  |  |  |  |  |  |  | acc |

## 算法框架

|  |
| --- |
|  |

## 主要数据结构

**pair<int, string>:**

用pair<int, string>来存储单个二元组。该对照表由专题1定义。

**map<string, int>:**

存储离散化后的终结符和非终结符。

**int[][]:**

存储算符优先矩阵。0代表空，1代表，2代表，3代表，4代表acc。

## 函数定义

|  |
| --- |
| **init:** |
| void init(); |
| **功能：** |
| 初始化算符优先矩阵，关键字及识别码对照表，离散化终结符 |
| **传入参数：** |
| （无） |
| **传出参数：** |
| （无） |
| **返回值：** |
| （无） |
|  |
| **sentenceAnalysis:** |
| bool sentenceAnalysis( vector<PIS> &vec, int &ncol ); |
| **功能：** |
| 进行该句的语法分析 |
| **传入参数：** |
| vec:该行二元式序列 |
| **传出参数：** |
| ncol:出错标识符为第ncol个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **errMsg:** |
| void errMsg( string filename, int rowNo, int colNo ); |
| **功能：** |
| 向屏幕输出错误信息 |
| **传入参数：** |
| filename:正在处理的文件的文件名称 |
| rowNo:出错行 |
| colNo:出错列 |
| **传出参数：** |
| （无） |
| **返回值：**（无） |

# 程序测试

测试用例详见文件夹中test1.lexer和test2.lexer。

其中，

test1.lexer、test2.lexer为测试输入文件。

在命令行中运行parse [name]即可运行测试用例。

test1为正确文法二元序列，test2为非法文法输入二元序列。

|  |
| --- |
|  |

# 心得体会

通过本次专题实验，我更加深刻的理解了算符优先语法分析的构造方法以及其程序的构造方式。

该专题的实现和LL(1)语法分析的实现方式上比较相近，都比较容易实现。整个分析过程中我认为推算优先矩阵时过程比较多比较繁琐，我认为这个推导过程比较容易出错。在实现程序过程中我认为课上PPT中所讲的程序框图有些复杂，因此我基于自己的理解加上多次对整个算符优先语法分析的过程进行手工模拟，我摸索出了一个自己的程序框架并且程序可以正确执行。

目前该工程已上传至我的Github仓库中。