编译原理课程实验报告

实验 2：语法分析

姓名	马宇斌	院系	软件学院	学号	161110407
任课教师	韩希先		指导教师	韩希先	
实验地点	研究院中 507-508		实验时间	2018．10.27	
实验课表现	出勤、表现得分		实验报告得分		实验总分	
	操作结果得分					

一、实验目的

1. 巩固对语法分析的基本功能和原理的认识。
2. 通过对语法分析表的自动生成加深语法分析表的认识。
3. 理解并处理语法分析中的异常和错误。
4. 通过编程实现 SLR(1)语法分析

二、实验内容

要求：对如下工作进行展开描述

1.给出如下语言成分的文法描述

本次实验的规定文法：

1)S->E

2)E->E+T

3)E->T

4)T->T*F

5)T->F

6)F->(E)

7)F->id

文法描述：

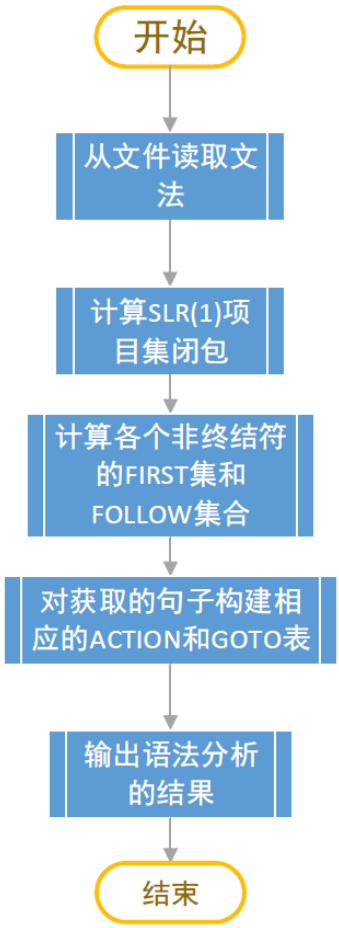
FOLLOW(E) = {), +, # }

FOLLOW(T) = {), +, #, * }

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

该文法是识别+与*的运算法则，乘法的优先级比加法的要高，当然也识别括号的。

2.语法分析程序的总体结构及物理实现



该语法分析器使用 SLR(1)的方式实现

求解 First 集：

```
public void First() {//求解first集
int index,after,before;
after = 0;
before = 0;
first = new String[nonterminal.length()];
for(int j=0;j<nonterminal.length();j++) {
    first[j] = ""+nonterminal.charAt(j);
}
do{
    before=after;
    for(int i=0;i<nonterminal.length();i++) {
        for(int j=0;j<wenfa.size();j++) {
            if(first[i].charAt(0)==wenfa.get(j).charAt(0)) {
                if(isTerminal(wenfa.get(j).charAt(3))) {
                    first[i]=removeSameString(first[i]+""+wenfa.get(j).charAt(3));
                }//将首字母为终结符的情况的first集添加进去 first数组的第一个字符是非终结符
                else{
                    index =getNonterminal(wenfa.get(j).charAt(3));//找到文法产生式的右部
```

```
第一个非终结符在非终结符的数组中所在的位置
                                if(first[index].length()>1) {
                                    for(int k=1;k<first[index].length();k++) {

    first[i]=removeSameString(first[i]+""+first[index].charAt(k));//在这是把能推出来的第一个非终结
符的first集此时可以写入产生式左边的非终结符的first集中
                                    }
                                }
                            }
                        }
                    }
                }
                after =0;
                for(int a=0;a<nonterminal.length();a++) {
                    after += first[a].length();
                }
            }while(after!=before);//after表示进行一轮循环后添加的first集中的所有的终结符数量当不产生变
化的时候代表求完了
    }
```

求解 Follow 集：

```
public void Follow() {//计算follow集
        int index,before,after,position;
        before = 0;
        after = 0;
        follow = new String[nonterminal.length()];
        for(int j=0;j<nonterminal.length();j++) {
            follow[j] = ""+nonterminal.charAt(j);
        }
        follow[0]=follow[0]+'#';
        do {
            before = after;
            for(int i=0;i<nonterminal.length();i++) {
                for(int j=0;j<wenfa.size();j++) {
                    position=nonterminalPositon(wenfa.get(j),nonterminal.charAt(i));//找到非终结
符在文化的候选式当中的位置
                    if(position!=-1) {
                        if(position!=wenfa.get(j).length()-1) {//不在候选式当中的最后一个
                            if(isTerminal(wenfa.get(j).charAt(position+1))){//文法的候选式后面
一个是一个终结符

    follow[i]=removeSameString(follow[i]+""+wenfa.get(j).charAt(position+1));
                            }
                            else {
                                index =getNonterminal(wenfa.get(j).charAt(position+1));//得到文
```

```
法的候选式后面一个的非终结符的位置
                                    if(first[index].length()>1) {
                                        for(int k=1;k<first[index].length();k++) {

    follow[i]=removeSameString(follow[i]+""+first[index].charAt(k));//把后一个的非终结符的first集加
入
                                        }
                                    }
                                }
                            }
                            else {
                                index =getNonterminal(wenfa.get(j).charAt(0));//产生式左边的非终结符
的位置
                                if(follow[index].length()>1) {

                                    for(int k=1;k<follow[index].length();k++) {

    follow[i]=removeSameString(follow[i]+""+follow[index].charAt(k));//能写在产生式左边的follow可以
写在产生式的右边

                                    }
                                }
                            }
                        }
                    }
                }
            after =0;
            for(int a=0;a<nonterminal.length();a++) {
                after += follow[a].length();
            }
        }while(after!=before);
    }
```

求解项目集闭包：

```
public void CLOSURE(Closure C) {
        int i,j,k;
        String item;
        for(i=0;i<C.getItems().size();i++) {
            j=3;
            while(C.getItems().get(i).charAt(j)!='.'&&j<C.getItems().get(i).length()) {
                j++;
            }
            j++;
            item =C.getItems().get(i);
            if(j<item.length()&&isTerminal(item.charAt(j))==false) {        //如果是非终结符
```

```
                    for(k=0;k<wenfa.size();k++) {    //遍历文法
                        if(wenfa.get(k).charAt(0)==item.charAt(j)) {        //如果文法左部变量中有与
刚刚的非终结符
                                if(IsExist(GetItem(wenfa.get(k)),C)==false) {     //如果Ix中没有该项
                                    C.getItems().add(GetItem(wenfa.get(k)));    //将该项存入项目集中
                                }

                        }
                    }
                }
            }
              }
```

3.语法分析表及其数据结构和查找算法以及打印

ACTION 表以及 GOTO 表的生成以及打印（ACTION 和 GOTO 表合并构成 SLR(1)分析表）：

```
public void setActionandGo(){
    char element,r,s;
    r='r';
    s='s';
    int c,t,n,index,position;
    c=closure.size();
    t=total.length();
    n=nonterminal.length();
    String acc = closure.get(1).getItems().get(0);
    String item;
    SLR = new String[c][t];
    ACTION = new String[c][t-n];
    GOTO = new int[c][n];
    for(int i=0;i<c;i++) {
        for(int j=0;j<t;j++) {
            SLR[i][j]="";
        }
    }
    for(int i=0;i<c;i++) {
        for(int j=0;j<t-n;j++) {
            ACTION[i][j]="error";
        }
    }
    for(int i=0;i<c;i++) {
        for(int j=0;j<n;j++) {
            GOTO[i][j]=-1;
        }
    }
    SLR[1][terminal.length()-1] ="acc";
    ACTION[1][terminal.length()-1] ="acc";
```

```
        for(int i=0;i<closure.size();i++) {
            for(int j=0;j<closure.get(i).getItems().size();j++) {
                item = closure.get(i).getItems().get(j);
                element=getWord(item);
                index = getElementPositon(element);//得到字符在所有字符中的位置
                if(element=='$'&&!(item.equals(acc))) {
                        for(int k=0;k<terminal.length();k++) {
                            position = findWenfa(item);
                            if(k<terminal.length()&&isInFollow(position,terminal.charAt(k))) {
                                SLR[i][k]=""+r+position;
                                ACTION[i][k]=""+r+position;
                            }
                            /*if(k==terminal.length()&&isInFollow(position,'#')) {
                                SLR[i][k]=""+r+position;
                                ACTION[i][k]=""+r+position;
                            }*/
                        }
                }
                else if(isTerminal(element)&&element!='$') {
                    position = ifRepeat(Go(closure.get(i),element));
                    SLR[i][index] = ""+s+position;
                    ACTION[i][index]=""+s+position;
                }
                else if(!isTerminal(element)) {
                    position = ifRepeat(Go(closure.get(i),element));
                    SLR[i][index] = ""+position;
                    index = index-terminal.length();
                    GOTO[i][index] =position;
                }
            }
        }
    }
public void showGOTO() {
        for(int i=0;i<closure.size();i++) {
            System.out.print(i+" ");
            for(int j=0;j<nonterminal.length();j++) {
                if(GOTO[i][j]==-1) {
                    System.out.print("  ");
                }
                System.out.print(GOTO[i][j]+" ");
            }
            System.out.println("");
        }
    }
```

```
    public void showACTION() {
        for(int i=0;i<closure.size();i++) {
            System.out.print(i+" ");
            for(int j=0;j<terminal.length();j++) {
                if(ACTION[i][j]=="") {
                    System.out.print("  ");
                }
                System.out.print(ACTION[i][j]+" ");
            }
            System.out.println("");
        }
    }
    public void showSLR() {
        System.out.println("SLR(1)分析表: ");
        System.out.printf("%-5s","");
        for(int k=0;k<total.length();k++) {
            System.out.printf("%-6s",total.charAt(k)+" ");
        }
        System.out.println("");
        for(int i=0;i<closure.size();i++) {
            System.out.printf("%-5s",i);
            for(int j=0;j<total.length();j++) {
                System.out.printf("%-6s",SLR[i][j]);
            }
            System.out.println("");
        }
        System.out.println("-------------------------------------------------------------");
            }
```

4.错误处理

语法分析时可能的错误有当扫描下一个输入时出现一个不正确的后继，此时的处理即不会把出错点的输入符号移进入栈，当语法分析程序发现错误时将会从栈顶开始退栈，直到发现在特定语法变量A上具有转移的状态S为止，然后丢弃0个或多个输入符号，直到找到符号a∈FOLLOW（(A)为止。接着分析程序把状态GOTO(S,A)压入栈，并恢复正常分析。

三、实验结果

要求：将实验获得的结果进行描述，基本内容包括：

针对一测试程序输出其语法分析结果

项目集闭包：

```
closure:0       S->.E
closure:0       E->.E+T
closure:0       E->.T
closure:0       T->.T*F
closure:0       T->.F
closure:0       F->.(E)
closure:0       F->.d
-------------------------------------------------------
closure:1       S->E.
closure:1       E->E.+T
-------------------------------------------------------
closure:2       E->T.
closure:2       T->T.*F
-------------------------------------------------------
closure:3       T->F.
-------------------------------------------------------
closure:4       F->(.E)
closure:4       E->.E+T
closure:4       E->.T
closure:4       T->.T*F
closure:4       T->.F
closure:4       F->.(E)
closure:4       F->.d
-------------------------------------------------------
closure:5       F->d.
-------------------------------------------------------
closure:6       E->E+.T
closure:6       T->.T*F
closure:6       T->.F
closure:6       F->.(E)
closure:6       F->.d
-------------------------------------------------------
closure:7       T->T*.F
closure:7       F->.(E)
closure:7       F->.d
-------------------------------------------------------
closure:8       F->(E.)
closure:8       E->E.+T
-------------------------------------------------------
closure:9       E->E+T.
closure:9       T->T.*F
-------------------------------------------------------
closure:10      T->T*F.
-------------------------------------------------------
closure:11      F->(E).
-------------------------------------------------------
```

SLR(1)分析表：

```
SLR(1)分析表:
     +     *     (     )     d     #     S     E     T     F
0                  s4          s5                a1    a2    a3
1    s6                                  acc
2    r2    s7          r2          r2
3    r4    r4          r4          r4
4                  s4          s5                a8    a2    a3
5    r6    r6          r6          r6
6                  s4          s5                      a9    a3
7                  s4          s5                            a10
8    s6                  s11
9    r1    s7          r1          r1
10   r3    r3          r3          r3
11   r5    r5          r5          r5
-------------------------------------------------------------
```

输入为：1*(2+5)+5*6#对应的语法分析结果

状态栈	符号栈	
[0]	#	d*(d+d)+d*d# 移入
[0, 5]	#d	*(d+d)+d*d# 根据F->d归约
[0, 3]	#F	*(d+d)+d*d# 根据T->F归约
[0, 2]	#T	*(d+d)+d*d# 移入
[0, 2, 7]	#T*	(d+d)+d*d# 移入
[0, 2, 7, 4]	#T*(	d+d)+d*d# 移入
[0, 2, 7, 4, 5]	#T*(d	+d)+d*d# 根据F->d归约
[0, 2, 7, 4, 3]	#T*(F	+d)+d*d# 根据T->F归约
[0, 2, 7, 4, 2]	#T*(T	+d)+d*d# 根据E->T归约
[0, 2, 7, 4, 8]	#T*(E	+d)+d*d# 移入
[0, 2, 7, 4, 8, 6]	#T*(E+	d)+d*d# 移入
[0, 2, 7, 4, 8, 6, 5]	#T*(E+d	)+d*d# 根据F->d归约
[0, 2, 7, 4, 8, 6, 3]	#T*(E+F	)+d*d# 根据T->F归约
[0, 2, 7, 4, 8, 6, 9]	#T*(E+T	)+d*d# 根据E->E+T归约
[0, 2, 7, 4, 8]	#T*(E	)+d*d# 移入
[0, 2, 7, 4, 8, 11]	#T*(E)	+d*d# 根据F->(E)归约
[0, 2, 7, 10]	#T*F	+d*d# 根据T->T*F归约
[0, 2]	#T	+d*d# 根据E->T归约
[0, 1]	#E	+d*d# 移入
[0, 1, 6]	#E+	d*d# 移入
[0, 1, 6, 5]	#E+d	*d# 根据F->d归约
[0, 1, 6, 3]	#E+F	*d# 根据T->F归约
[0, 1, 6, 9]	#E+T	*d# 移入
[0, 1, 6, 9, 7]	#E+T*	d# 移入
[0, 1, 6, 9, 7, 5]	#E+T*d	# 根据F->d归约
[0, 1, 6, 9, 7, 10]	#E+T*F	# 根据T->T*F归约
[0, 1, 6, 9]	#E+T	# 根据E->E+T归约
[0, 1]	#E	# 接受

四、实验中遇到的问题总结

要求：主要阐述两方面的问题

（一） 实验过程中遇到的问题如何解决的？

问题一：计算 Closure 闭包的问题

根据课本上Closure的生成算法，首先生成文法的I(0)闭包，然后根据读入的下一个字符和I(0)闭包来计算以后的项目集闭包。

问题二：文法与项目集闭包的存储结构的选择

由于文件中的文法条目未知，如果选用确定空间大小的数组来存储，有可能存在空间不足或者空间浪费的情况，因此Arraylist成了更好的选择，首先创建一个对象，让后用add方法直接往该对象中添加即可。

（二）思考题的思考与分析

思考题 1：给出在生成语法分析表时所遇到的困难，以及是如何处理的？

直接生成语法分析表难以实现，分成 ACTION 表和 GOTO 两张表组合成语法分析表

思考题 2：思考还可以什么形式来给出语法分析的结果？

自顶向下的方法 LL(1)分析法

思考题 3：如果在语法分析中遇到了语法错误，是应该中断语法分析呢，还是应该进行适当处理后继续语法分析，你是怎么处理的？

应该进行适当处理继续分析，设置出错处理函数，对于错误情况进行适当的处理

五、实验体会

通过这次语法分析 SLR（1）实验，我加深了对语法分析的理解，并且能熟练利用分析表进行 GOTO 函数和 ACTION 函数的应用，以前在书本上学到的东西，没有实验之前都很抽象，但是自己通过实验，利用数组产生了分析表，对 SLR（1）自动机也有了很深的理解！

指导教师评语：

日期：