合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

编译原理实验报告

学生姓名： 朱航延

学 号： 2016212009

专 业： 计算机科学与技术

班 级： 2016 级 2 班

指导教师： 李宏芒

完成日期： 2018 年 6 月 12 日

目　　录

实验一　词法分析设计

1.1 开发工具：JAVA 10

1.2 开发环境：

1.3 词法分析实验设计思想及算法

本实验采用的基本思想是根据绝大多数语言的基本状态集，根据各个状态的转移条件构造各个判断逻辑，将程序运行的状态加以区别。源程序来源采用文件导入的方式，对输入源程序的文件进行逐行的词法分析。

同时提供 UI 界面，使用直接友好。为了满足普适性，尽可能的满足多种文法的词法分析需求，本程序的关键字、算数运算符和逻辑运算符均可手动设置，并保存到外部文件，以便下次使用。

对浮点数进行了特殊处理，可以满足部分不支持浮点数的语言（如汇编语言）的词法分析需求；同时支持对”//”和“/*”两种注释方法的处理。

1.3.1 主程序设计基本思路

按照如下状态转移图的形式，对目标分析文件中的内容进行分析：

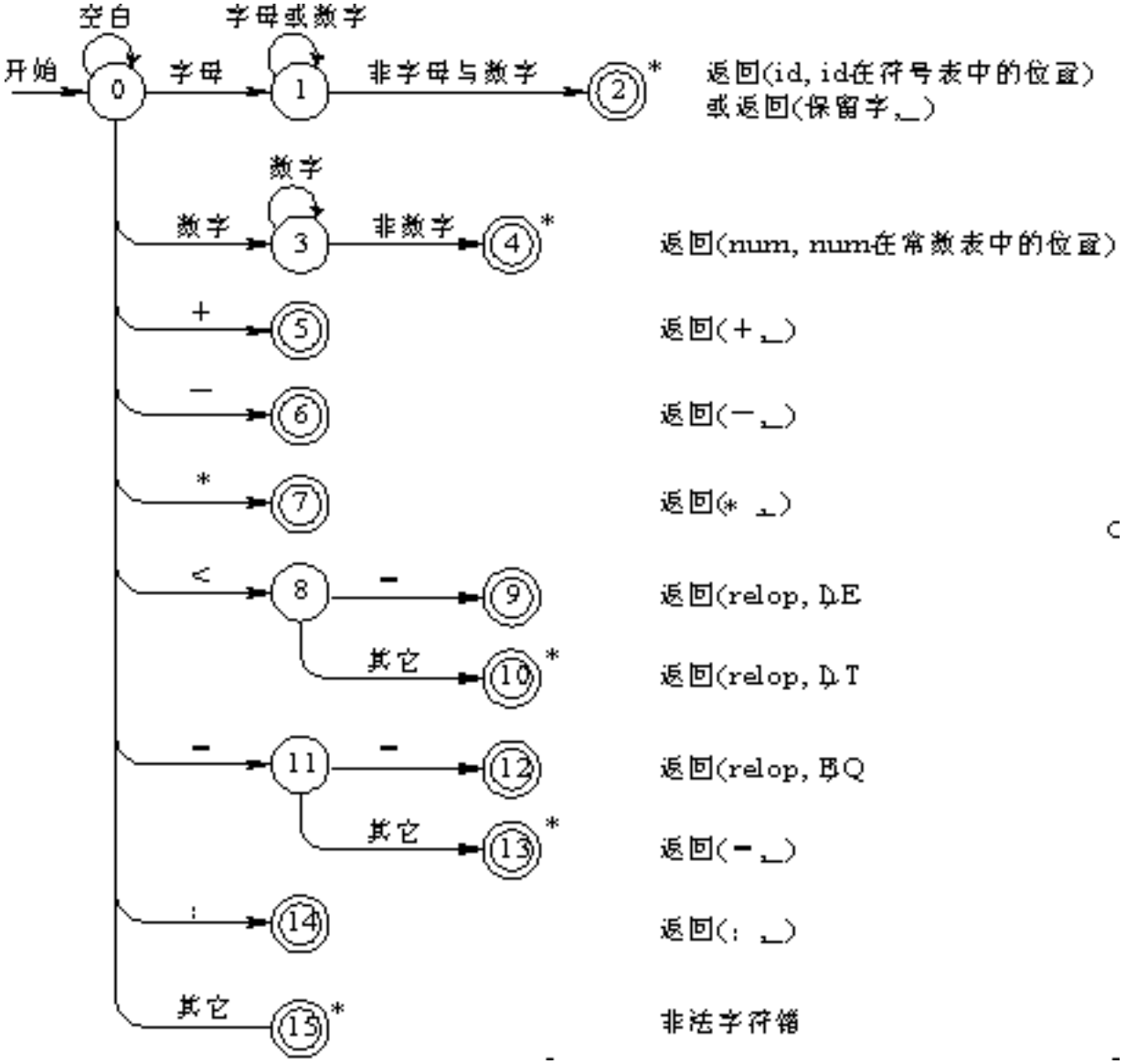

图 1.3.1-1 文法逻辑图

按照上图所示，按照大多数文法的特性，采用超前搜索的核心思路进行分析。源文件读入后，默认进入状态 0，超前搜索指针 counter_end 向前搜索，按照以下分析逻辑依次序进行：

（1）.如果识别到空格，进度指针 counter_start 的进度向前推进，随后 counter_end 指针向前移进；

（2）.如果识别到字母，程序进入状态 1，超前搜索指针 counter_end 的进度向前推进，如果 counter_end 识别到的字符为字母或数字，比对关键字集合，识别是否为关键字，如果匹配成功，则识别本识别子串为保留字，如果进行了其他状态的转化，则认为该子串为变量名，对变量名表进行比对分析，查重后加入；此次运行结束后，进度指针 counter_start 的进度向前推进到 counter_end，随后 counter_end 指针向前移进；

（3）.如果识别到数字，程序进入状态 3，超前搜索指针 counter_end 的进度向前推进，如果 counter_end 识别到的字符为数字 counter_end 向前推进；第一次识别到“.”后，程序 flag 置为 1，继续分析；如果再识别到“.”程序报错。当识别到非数字和“.”时，程序状态改变，将以识别的子串加入数字集。此次运行结束后，进度指针 counter_start 的进度向前推进到 counter_end，随后 counter_end 指针向前移进；（为了保证改程序的普适性，对“.”进行了特殊处理，如果在程序中支持的算术运算符表中识别到了“.”则按照浮点数的逻辑处理，若未识别到，则按照错误处理）

（4）.如果识别到运算符或分隔符，程序进入相应的状态中，进行各自的词法分析逻辑；

程序逻辑图如下：

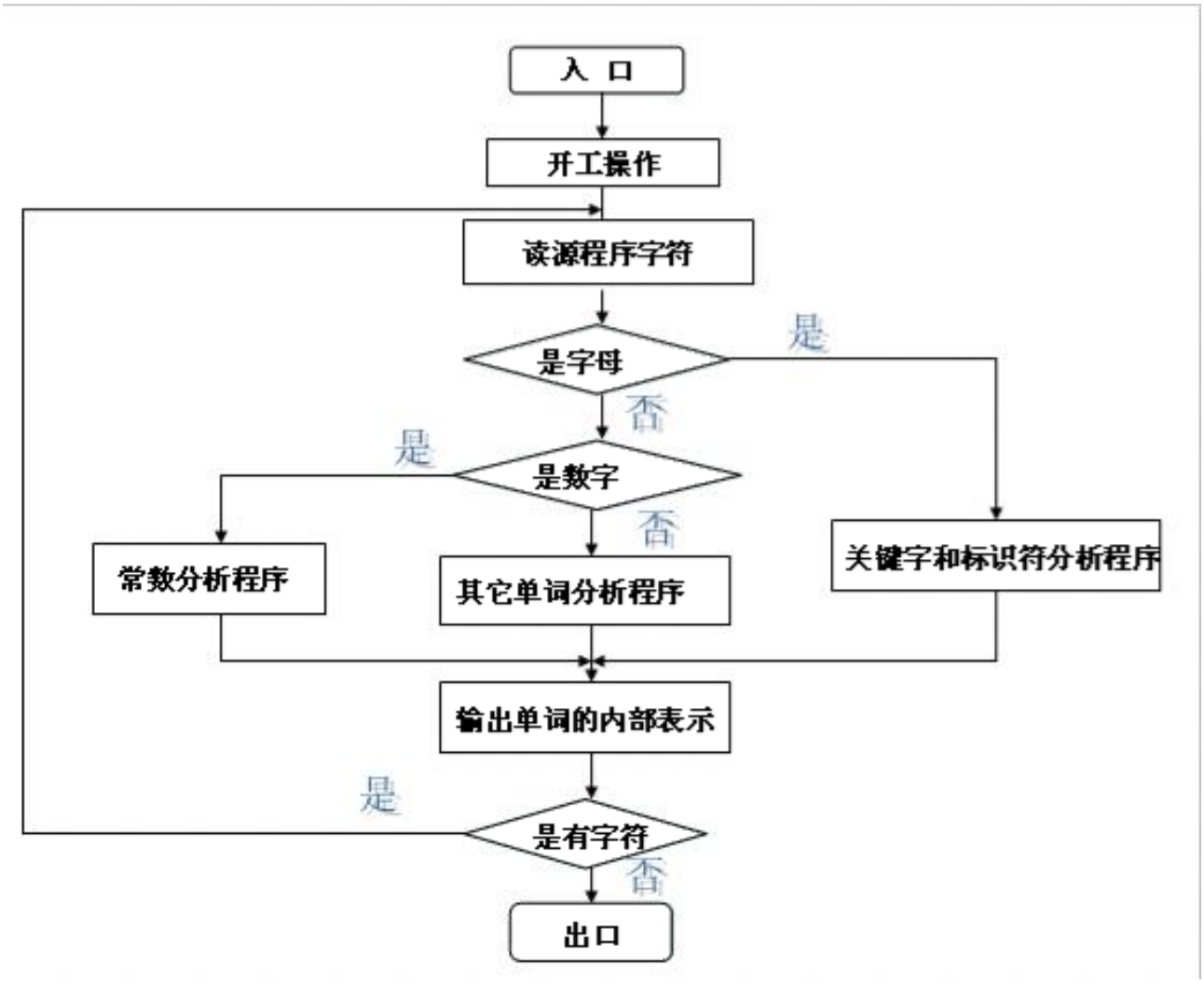

图 1.3.1-2 程序逻辑图

1.3.2 具体功能函数思想与实现：

analyzer:分析器的主体，其构造函数为读入源文件的初始化

String[]　k：保存语言的关键字；

String[]　as：保存语言的算数运算符；

String[]　rs：保存语言的关系运算符；

Vector<String>　di：保存语言的分隔符；

Vector<String>　id：保存识别到的标识符；

Vector<String>　ci：保存识别到的常数；

klistener：　初始化：读入输入的关键字并存储到外部文件；

aslistener：初始化：读入输入的算数运算符并存储到外部文件；

rslistener：初始化：读入输入的逻辑运算符并存储到外部文件；

Confirmlistener：执行词法分析主程序；

1.3.3 以下为源程序关键代码(字符串解析主程序部分)：

（完整源程序位于附件“Lexcial_analyzer.java”）

```
class Confirmlistener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            int line_num=0;
            int counter_start=0,counter_end=0;
            try
            {
                FileReader fr1 = new FileReader("/Users/miku/Desktop/source.txt");
 //处理的文件为 fr1->source
                BufferedReader br1 = new BufferedReader(fr1);
                String handle = br1.readLine();
                while(handle.length()!=0)
                {
                    line_num++;
                    counter_end = 0;
                    counter_start = 0;

                    if (handle.charAt(handle.length()-1) != ';')
                    {
```

```
                    //  result.append("Error:行"+line_num+"出现非法字符!\n");
                }
                while(counter_start < handle.length())        //处理每一行
                {
                    if((handle.charAt(counter_end)==' ')||(handle.charAt(counter_end)==';')||(handle.charAt(counter_end)=='(')||(handle.charAt(counter_end)==')')||(handle.charAt(counter_end)=='{')||(handle.charAt(counter_end)=='}'))
                    {
                        if (handle.charAt(counter_end)==';')
                        {
                            result.append(";\t\t\t"+"(2,;)\t\t\t 分界符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        }
                        if (handle.charAt(counter_end)=='(')
                        {
                            result.append("(\t\t\t"+"(2,()\t\t\t 分界符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        }
                        if (handle.charAt(counter_end)==')')
                        {
                            result.append(")\t\t\t"+"(2,))\t\t\t 分界符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        }
                        if (handle.charAt(counter_end)=='{')
                        {
                            result.append("{\t\t\t"+"(2,{)\t\t\t 分界符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        }

                        if (handle.charAt(counter_end)=='}')
                        {
                            result.append("}\t\t\t"+"(2,})\t\t\t 分界符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        }


                        counter_end++;
                        counter_start = counter_end; //start,end同时指向下一个字符                    }

                    else if((handle.charAt(counter_end)>='a')&&(handle.charAt(counter_end)<='z'))//识别到字母
                    {
                        String temp = "";

                        while((handle.charAt(counter_end)>='a')&&(handle.charAt(counter_end)<='z'))
                        {
```

```
                int flag = 0;
                temp = temp + handle.charAt(counter_end);


if((handle.charAt(counter_end+1)<'a'||handle.charAt(counter_end+1)>'z')&&flag==0)
                {

                    for(int i=0;i < k.length;i++)
                   {
                      if(temp.equals(k[i]))  //成功匹配到保留字
                     {
                        result.append(temp+"\t\t\t"+"(1,"+temp+")\t\t\t
关键字\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        counter_end = counter_end + 1;
                        counter_start = counter_end;
                        temp = "";
                        flag = 1;
                        break;
                     }

                   }
                    if (flag == 0)
                    {
                        result.append(temp+"\t\t\t"+"(6,"+temp+")\t\t\t
标识符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                        for (int i = 0; i < id.size(); i++)
                        {
                            if (temp.equals(id.get(i)))
                            {

                                break;
                            }
                            if (i == id.size() - 1)
                            {
                                id.add(temp);
                                //result.append
                            }

                        }

                        temp = "";
                        counter_end = counter_end + 1;
                        counter_start = counter_end;
                        flag = 1;
                    }
                }

                if(flag == 1) break; //如果从这里跳出，则是匹配到了保留
字，或是下一个字符已不是字母，end 和 start 指向下一个字符
```

```
                counter_end = counter_end +1;
            }
        } else
if((handle.charAt(counter_end)>='0')&&(handle.charAt(counter_end)<='9')) //识别到数字
        {
            String temp = "";

while((handle.charAt(counter_end)>='0'&&handle.charAt(counter_end)<='9'))
            {
                temp = temp + handle.charAt(counter_end);
                counter_end = counter_end + 1;
            }
            result.append(temp+"\t\t\t"+"(5,"+temp+")\t\t\t 常数
\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
            for(int i = 0;i<ci.size();i++)
            {
                if(temp.equals(ci.get(i)))
                {

                    break;
                }
                if(i==ci.size()-1)
                {
                    ci.add(temp);
                                                    }
            }

            counter_start = counter_end;
//此时，start 和 end 指向下一个位（不再是数字）                    } else
        {   String temp = String.valueOf(handle.charAt(counter_end));
            int flag1=0,flag2=0;


if(handle.charAt(counter_end)=='/'&&handle.charAt(counter_end+1)=='/')
            {
                counter_end = 0;
                counter_start = 0;
                handle = br1.readLine();

                continue;
            }

            if(handle.charAt(counter_end)=='/' &&
handle.charAt(counter_end+1)=='*')
            {

while(handle.charAt(counter_end)!='*'||handle.charAt(counter_end+1)!='/') ///识别到的
量和下一个量不是"*/"                                {
                    if(counter_end == handle.length()-1)
                    {
```

```
                                counter_end = 0;
                                counter_start = 0;
                                handle = br1.readLine();
                            }
                            counter_end = counter_end+1;
                            counter_start = counter_end;
                        }
                        counter_end = counter_end+2;
                        counter_start = counter_end;

                        continue;
                    }

                    for(int i=0;i<as.length;i++)
                    {
                        if(temp.equals(as[i]))
                        {
                            result.append(temp+"\t\t\t"+"(4,"+temp+")\t\t\t 算
术运算符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                            flag1 = 1;
                        }
                    }

                    for(int i=0;i<rs.length;i++)
                    {
                        if(temp.equals(rs[i]))
                        {
                            result.append(temp+"\t\t\t"+"(4,"+temp+")\t\t\t 关
系运算符\t\t\t"+"("+line_num+","+(counter_start+1)+")\n");
                            flag2 = 1;
                        }
                    }

                    if(flag1==0&&flag2==0&&(handle.charAt(counter_end)!='/'))
                    {

result.append(temp+"\t\t\t"+"Error\t\t\tError\t\t\t"+"("+line_num+","+(counter_start+1)
+")\n");
                    }
                    counter_end++;
                    counter_start=counter_end;
                }
            }//单行处理完毕
            handle = br1.readLine();
        }
    }catch (IOException es){result.append(es.getMessage());}
  }
}
```

1.4 程序运行结果如下：

导入源文件为“/Users/miku/Desktop/source.txt”

导出的关键字文件为”/Users/miku/Desktop/保留字.txt“

导出的算数运算符文件为”/Users/miku/Desktop/算数运算符.txt“

导出的逻辑运算符文件为”/Users/miku/Desktop/逻辑运算符.txt“

图 1.4-1 程序运行结果

实验二　LL(1)分析法

2.1 开发工具：Python 3.6

2.2 开发环境：

2.3 LL(1)分析法实验设计思想及算法

本程序用途为对 LL（1）文法进行全自动处理和识别，包括以下子功能：

（1）识别文法是否存在直接左递归，如果存在，消除左递归；

（2）识别文法是否存在间接左递归，如果存在，消除左递归；

（3）检测文法是否需要去回溯，如果需要，使用提取左因子的方式去除回溯；

（4）求文法所有产生式右部的 FIRST 集；

（5）求文法所有非终结符的 FOLLOW 集；

（6）通过 FIRST 集与 FOLLOW 集构建分析表；

（7）根据分析表进行 LL（1）文法的语句分析。

同时提供了 UI 界面，使用更直观方便。

2.3.1 主程序设计基本思路

首先，通过 UI 上方的视窗键入需要分析的文法，随后通过 UI 的视窗键入需要识别到分析串，按钮“文法分析开始”触发后，程序依次执行以下命令：

（1）.判断文法分析视窗内文法是否为间接左递归，如果为间接左递归，则按照处理函数的逻辑进行处理，最后以字符串方式返回给主程序，同时在 UI“去除左递归后”视窗内显示；

（2）.判断文法分析视窗内文法是否为直接左递归，如果为直接左递归，则按照处理函数的逻辑进行处理，最后以字符串方式返回给主程序，同时在 UI“去除左递归后”视窗内显示；

（3）.如果不存在左递归，则直接将原文法在 UI“去除左递归后”视窗内显示，随后执行主程序内 getTable（）函数，获得分析表；

（4）.获取分析表后，启动分析程序，结合分析表对读入的输入串进行语句分析，并以表格的形式将分析表、分析过程显示到 UI 相应位置。

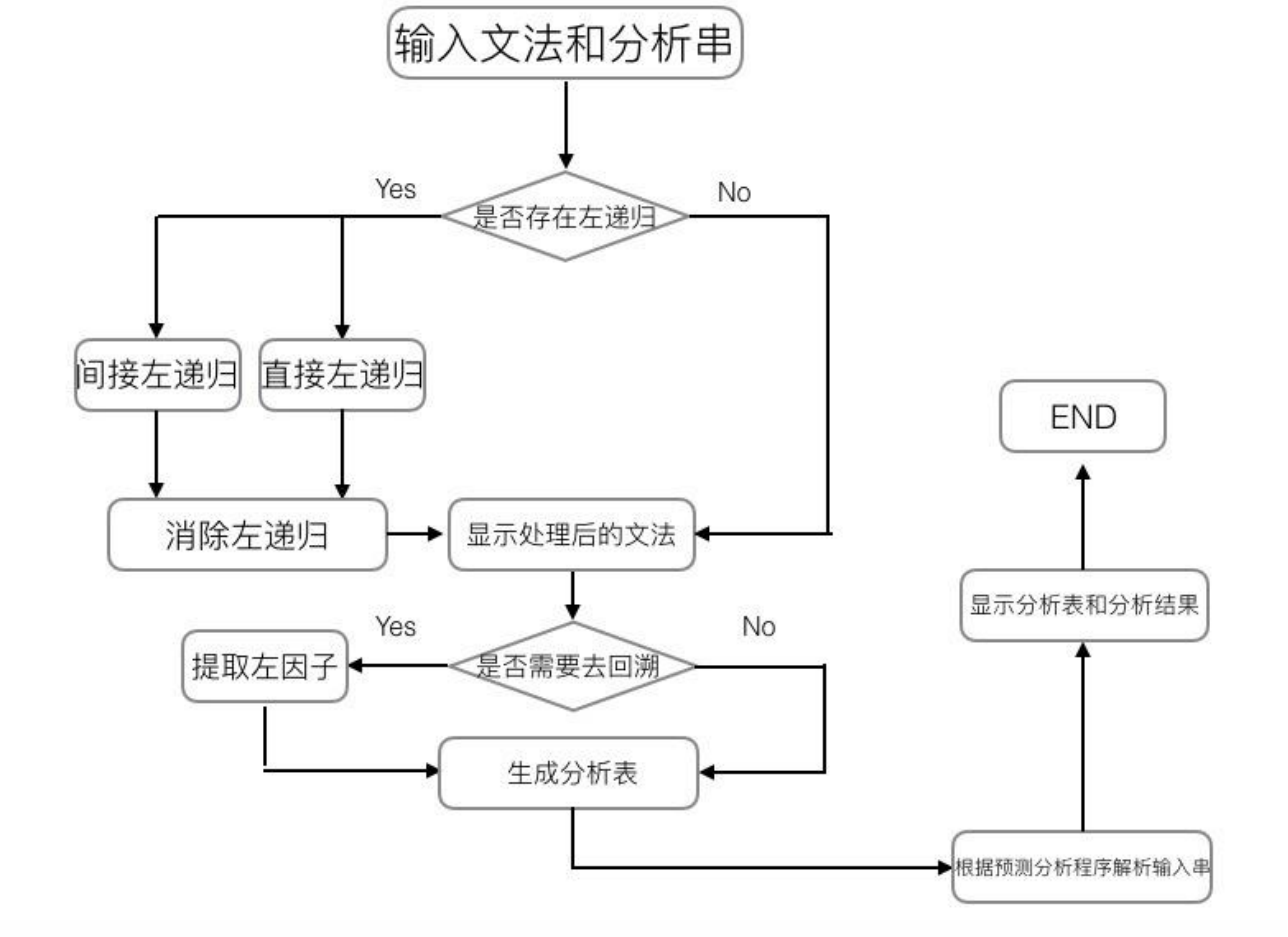

图 2.3.1-1 程序逻辑图

2.3.2 具体函数逻辑：

（1）.getTable 函数：生成分析表

首先利用__getFirstSet()函数求出所有的产生式右部的 FIRST 集，判断是否存在交集不为空的情况，若不为空，则进行字符串匹配，并按照

A->δA’ |γ1|γ2|...|γn

A’ ->β1|β2|...|βn

的规则进行左因子提取，并将提取结果作为下一步的输入；若为空，则跳过这一步；

构造所有非终结符的 FOLLOW 集，通过将各终结符与每个产生式的 FIRST 集作比较、同时再利用存在产生式推出空串可能性的 FOLLOW 集，生成分析表。

（2）.FirstSet 函数

此函数的用途是产生该产生式右部对应的 FIRST 集;

（3）.getFirstSet 函数

此函数的用途是遍历所有产生式右部，调用 FirstSet 函数得到所有的产生式右部的 First 集；

（4）.followset 函数

此函数的作用为得到指定的非终结符的 follow 集;

（5）.getFollow 函数

此函数的作用为计算出所有非终结符的 Follow 集;

（6）.left_cleanr_pre 函数

检测并处理间接左递归，若检测到，进行处理（需要调用 left_clean 函数）；若不含，返回原字符串；

（7）.left_clean 函数

检测并处理直接左递归；若检测到，进行处理；若不含，返回原字符串；

（8）.OnAnalysis 函数

为按钮的响应函数，本质上调用 getTable 等函数，为主分析控制程序开关。

（9）.getItSplited 函数

把输入的字符串分割为 firstset 函数可以识别的数组形式;

2.3.3 以下为关键源代码(各函数作用已注释):

(完整源代码请见文件”LL(1)Analysis.py”)

```
class Table():
    def __init__(self,s):
        self.__s = re.split("\\r\\n|\\n",s)      #将读入的字符串分成数组
       self.__SplitSet = {}
        self.__SplitSet["keys"] = []
        self.__FirstSet = {}
        self.__FollowSet = {}
    def __getItSplited(self,s):
        if len(s) == 0:
            return None
        t = s.split("->")     #t 为 s 用箭头作为前后分割后的情况
       ans = {}
        ans[t[0]] = []        #对 ans 中,E(假如此次分析的是左端为 E 的产生式)为索引的项进行操作
       t2 = t[1].split("|") #t2 为 E 产生式右侧,并用|分割后,保存为一个数组
       for i in range(len(t2)):  #把 t2 里面 E 产生式右侧的值一个个放到 st 里面
           st = t2[i]
            sta = []
            MeetLetter = False
            ss = ""
            for i in range(len(st)):  #开始按位分析 E 产生式的每一个用|分割的右侧
             if MeetLetter:
                    if ord(st[i]) >= ord('A') and ord(st[i]) <= ord('Z'): #遇到非终结符,把
ss 现有值放到 sta 里,ss 置为该非终结符
                    sta.append(ss)
                        ss = st[i]
                    elif st[i] == '\'':     #如果检测到的这一位是',在 ss 里加上
                    ss+="\'"
                    else:
                        sta.append(ss)      #不是 E 或 E`这种非终结符,把 ss 现有值放到 sta,ss 更新
为识别到的值,(即识别到终结符)将识别到非终结符标志清零
                    ss = st[i]
                        MeetLetter = False
                else:
                    if ord(st[i]) >= ord('A') and ord(st[i]) <= ord('Z'):
                        if len(ss) > 0:
                            sta.append(ss)
                        ss = st[i]
                        MeetLetter = True
                    else:
                        if len(ss) > 0:
                            sta.append(ss)
                        ss = st[i]
            if len(ss) > 0:                         #当前产生式识别完了,把最后的 ss 放进 sta
                sta.append(ss)
            ans[t[0]].append([i for i in sta])     #把 sta 放到 ans[t[0]]里
       return ans
#     First 集
    def __firstSet(self,Key):
        Ans=[]
        if self.__SplitSet.get(Key) == None:
            Ans.append(Key)
            return Ans
        val=self.__SplitSet.get(Key)
        for i in range(len(val)):
            if self.__SplitSet.get(val[i][0])==None:
                if not(val[i][0] in Ans):
                    Ans.append(val[i][0])
            else:
                CountNone=0
                for j in range(len(val[i])):
```

```
                    NextRound=self.__firstSet(val[i][j])
                    CanBreak=True
                    for next in NextRound:
                        if next=="ε":
                            CanBreak=False
                            CountNone+=1
                        elif not(next in Ans):
                            Ans.append(next)
                    if CanBreak:
                        break
                if CountNone==len(val[i]):
                    if not("ε" in Ans):
                        Ans.append("ε")
        return Ans

    def __getFirstSet(self):
        for i in range(len(self.__s)):
            t=self.__getItSplited(self.__s[i])
            if t:
                key=list(t.keys())[0]
                self.__SplitSet[key]=t[key]
                self.__SplitSet["keys"].append(key)
        keys=self.__SplitSet["keys"]
        for i in range(len(keys)):
            AnsList=self.__firstSet(keys[i])
            self.__FirstSet[keys[i]]=copy.deepcopy(AnsList)
#     Follow 集
    def __followSet(self,Target,AnsList,HasChange):
        keys=self.__SplitSet["keys"]
        for i in keys:
            child=self.__SplitSet[i]
            for j in child:
                k=0
                size=len(j)
                while k<size:
                    if j[k]==Target:
                        next=k+1
                        if next==size:
                        #表示找到 target 的时候刚好在结尾，这时将 key 的 follow 集放到 target 的
follow 集
                            if Target!=i:
                                for m in self.__FollowSet[i]:
                                    if not (m in AnsList):
                                        AnsList.append(m)
                                        HasChange[0]=True
                            break
                        else:
                            MeetEmpty=True
                            while next<size and MeetEmpty:
                                MeetEmpty=False
                                if self.__FirstSet.get(j[next])==None:
                                    if not (j[next] in AnsList):#防止出现重复
                                     AnsList.append(j[next])
                                        HasChange[0]=True
                                    k=next
                                    break
                                else:
                                    NeedFirstSet=self.__FirstSet.get(j[next])
                                    for m in NeedFirstSet:
                                        if m!='ε':
                                            if not (m in AnsList):#防止出现重复
                                            AnsList.append(m)
                                                HasChange[0]=True
                                        else:
                                            MeetEmpty=True
                                next+=1
```

```
                        if next==size and MeetEmpty:
                            if Target!=i:
                                for m in self.__FollowSet[i]:
                                    if not (m in AnsList):#防止出现重复
                                     AnsList.append(m)
                                            HasChange[0]=True
                            k=size
                            break
                    k+=1
    def __getFollowSet(self):
        self.__getFirstSet()
        keys=self.__SplitSet["keys"]
        First=True
        Change=True
        for i in keys:
            self.__FollowSet[i]=[]
        while Change:
            Change=False
            for i in keys:
                AnsList=copy.deepcopy(self.__FollowSet[i])
                if First:
                    AnsList.append('#')
                    First=False
                HasChange=[False]
                self.__followSet(i,AnsList,HasChange)
                self.__FollowSet[i]=copy.deepcopy(AnsList)
                if HasChange[0]:
                    Change=True
    #构建分析表&主分析程序
    def getTable(self):
        self.__getFollowSet()
        TableSet={}
        keys=self.__SplitSet["keys"]
        TableSet["Start"]=keys[0]
        for i in keys:
            TableSet[i]={}
        for key in keys:
            for child in self.__SplitSet[key]:
                FirstKey=child[0]
                if FirstKey=='ε':
                    for i in self.__FollowSet.get(key):
                        TableSet[key][i]=child
                elif self.__FirstSet.get(FirstKey)==None:
                    TableSet[key][FirstKey]=child
                else:
                    SizeOfChild=len(child)
                    Pointer=0
                    while Pointer<SizeOfChild:
        #为了防止入 S->ABC,A-> ε 的情况产生
                        if self.__FirstSet.get(child[Pointer])==None:
                                TableSet[key][child[Pointer]]=child
                                break
                            else:
                                FSet=self.__FirstSet.get(child[Pointer])
                                MeetNone=False
                                for i in FSet:
                                    if i!='ε':
                                        TableSet[key][i]=child
                                    else:
                                        MeetNone=True
                                if not MeetNone:
                                    break
                                else:
                                    Pointer+=1
                        if Pointer==SizeOfChild:#说明这个产生式可以产生空串
                        for i in self.__FollowSet.get(key):
```

```
                    TableSet[key][i]=child
        return TableSet
```

2.4 以下为运行结果：

（实例为含有间接左递归）

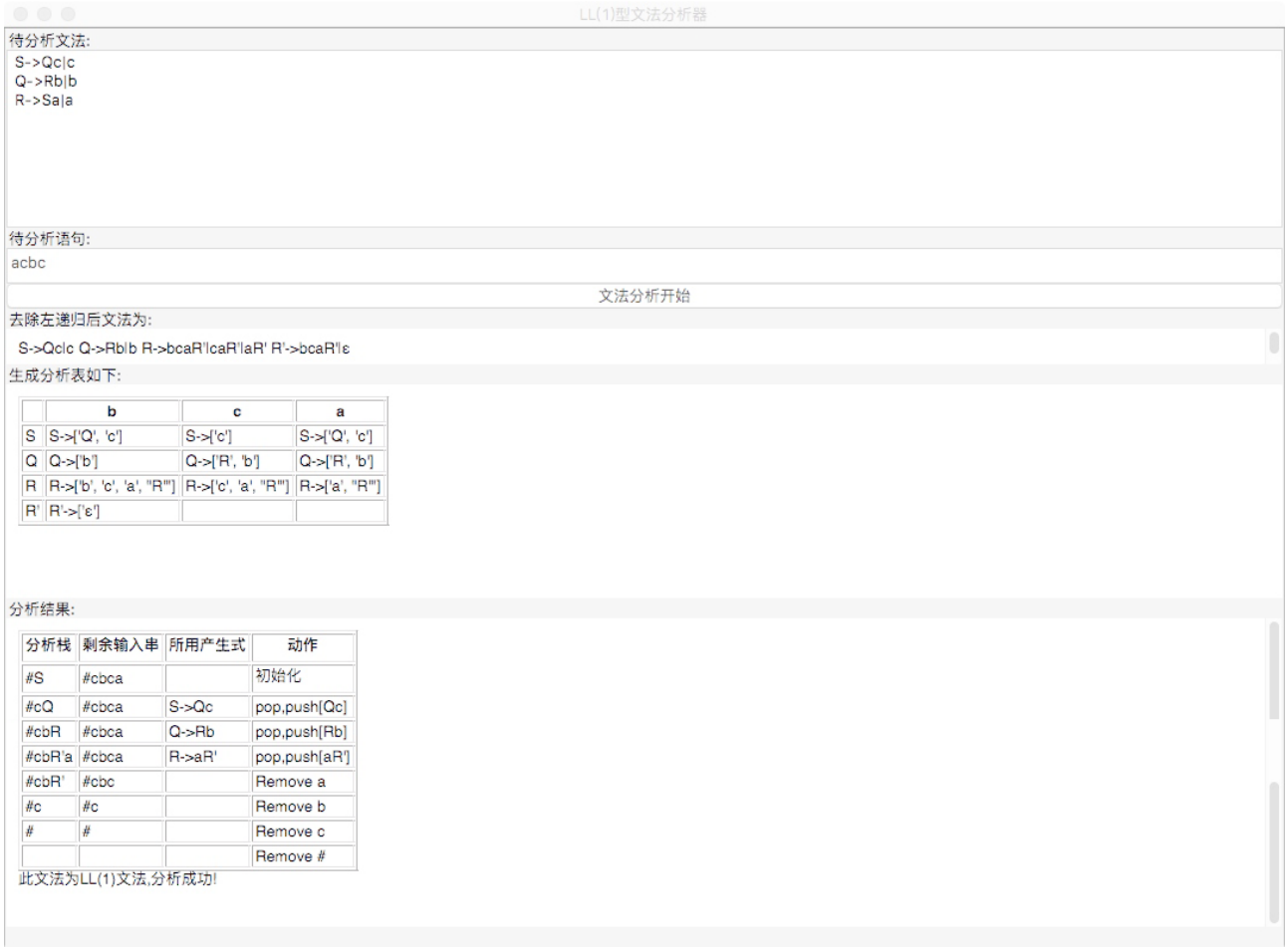

图 2.4-1 程序运行结果

实验三　LL(1)分析法

3.1 开发工具：Python 3.6

3.2 开发环境：

3.3 LR(1)分析法实验设计思想及算法

本程序可以对各种内容的 LR（1）文法进行全自动的全方位分析包含以下子功能：

（1）求产生式右部 FIRST 集；

（2）根据产生式所在的文法，求得非终结符的展望符；

（3）根据文法和产生式求其 ε 闭包，并以此为基础建立 LR（1）项目集族的状态集；

（4）根据求得的项目集族的状态集，根据对不同语法字符的识别，求其衍生的新状态集（即 GOTO 函数的功能）从而构建完整的 LR（1）项目集族；

（5）根据构建的 LR（1）项目集族，构筑分析表；

（6）根据构成的 LR（1）分析表，进行语句分析；

（7）　为了保证逻辑的严密性,此程序对于空串做了特殊分析.

同时提供了 UI 界面，使用更直观方便。

3.3.1 主程序设计基本思路

首先，通过 UI 上方的视窗键入需要分析的文法，随后通过 UI 的视窗键入需要识别到分析串，按钮“文法分析开始”触发后，程序依次执行以下命令：

（1）根据文法第一个识别到的非终结符（视为开始符号，假设为 S），增加产生式项目“S’ ->·S，#”；

（2）利用“S’ ->·S，#”为 I_0，求 Closure(I_0)，获得 I_0；

（3）利用 Closure(I_0)，计算 I0 中所有非终结符的展望符，遍历所有的 X（X 指所有 I_0 中“·”后的文法符号）,利用 GOTO（I_0，X），产生新的状态集 Ix；随后计算新产生的状态

（4）遍历每一个 I_x，利用 GOTO（I_x，X），并计算非终结符的展望符，完成完整的 LR（1）项目集族构建；

（5）根据完整的 LR（1）项目集族，构建项目分析表；

（5）获取分析表后，启动分析程序，结合分析表对读入的输入串进行语句分析，并以表格的形式将分析表、分析过程显示到 UI 相应位置。

其逻辑图如下：

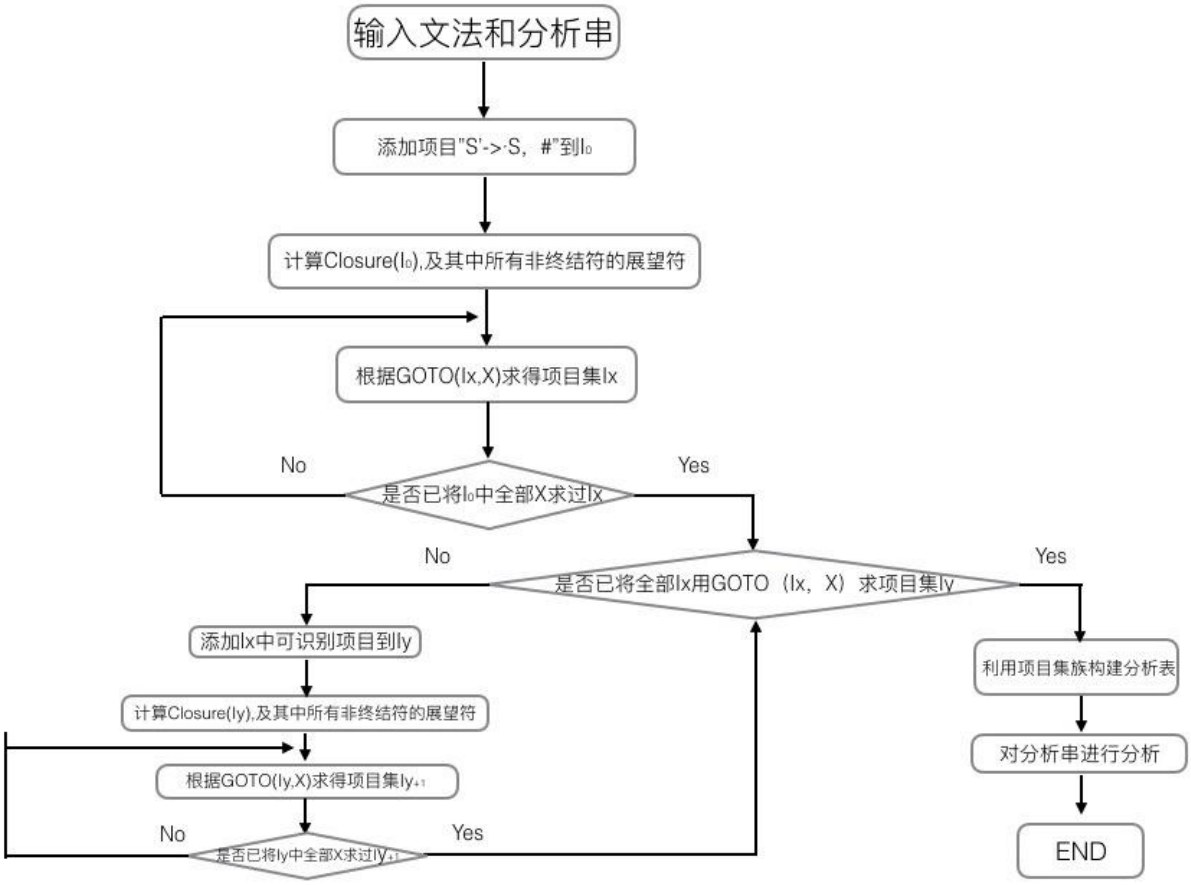

图 3.3.1-1 程序逻辑图

3.3.2 具体函数逻辑：

（1）getGrammar 函数：将识别到的字符串分割为数组,使之可以被程序处理;

（2）.FirstSet 函数

此函数的用途是产生该产生式右部对应的 FIRST 集;

（3）.getFirstSet 函数

此函数的用途是遍历所有产生式右部，调用 FirstSet 函数得到所有的产生式右部的 First 集；

（4）.Expand 函数

利用求得到 First 集等数据,计算生产 LR(1)状态分析集族;

（5）. CheckBelong 函数

检查当前项目是否属于某一集族;

（6）.getTable 函数

利用已有的集族构造 LR(1)分析表；

（7）.showTable 函数

输出分析表；

（8）.OnAnalysis 函数

为按钮的响应函数，本质上调用 getTable 等函数，为主分析控制程序开关。

3.3.3 以下为关键源代码(具体函数作用已注释)：

（完整源代码请见文件”LR(1)Analysis.py”）

```
class GrammarAnalysis():
    def __init__(self, s):
        self.__s = s
        self.__GrammarTable = {}
        self.__GrammarList = []
        self.__FirstSet = {}
        self.__Status = []
        self.__Action = {}
        self.__Goto = {}

    def __getGrammar(self):
        GrammarList = re.split("\n|\r\n", self.__s)
        count = 0
        for grammar in GrammarList:
            if len(grammar) == 0:
                continue
```

```
            SplitList = grammar.split("->")
            if self.__GrammarTable.get(SplitList[0]) == None:
                self.__GrammarTable[SplitList[0]] = []
            self.__GrammarTable[SplitList[0]].append([SplitList[1], False, count])
            count += 1
            self.__GrammarList.append(copy.deepcopy(SplitList))
    #获取变量的First集
    def __firstSet(self, Key):
        Ans = []
        if self.__GrammarTable.get(Key) == None:
            Ans.append(Key)
            return Ans
        val = self.__GrammarTable.get(Key)
        for i in range(len(val)):
            if not self.__GrammarTable[Key][i][1]:
                self.__GrammarTable[Key][i][1] = True
                if self.__GrammarTable.get(val[i][0][0]) == None:
                    if not (val[i][0][0] in Ans):
                        Ans.append(val[i][0][0])
                else:
                    CountNone = 0
                    for j in range(len(val[i][0])):
                        NextRound = self.__firstSet(val[i][0][j])
                        CanBreak = True
                        for next in NextRound:
                            if next == "ε":
                                CanBreak = False
                                CountNone += 1
                            elif not (next in Ans):
                                Ans.append(next)
                        if CanBreak:
                            break

                    if CountNone == len(val[i][0]):
                        if not ("ε" in Ans):
                            Ans.append("ε")
        return Ans
    #求所有变量的first集
    def __getFirstSet(self):
        self.__getGrammar()
        for key in self.__GrammarTable:
            AnsList = self.__firstSet(key)
            self.__FirstSet[key] = copy.deepcopy(AnsList)
            for k in self.__GrammarTable:
                for i in range(len(self.__GrammarTable[k])):
                    self.__GrammarTable[k][i][1] = False
    #获取table
    def __Expand(self, Given):
        for s in Given:
            t = s.split("-")
            type = int(t[0])
            position = int(t[1])
            n = len(self.__GrammarList[type][1])

            if position == n:
                pass
            elif position == n - 1:
                key = self.__GrammarList[type][1][position]
                if self.__GrammarTable.get(key) == None:
                    continue
                if self.__GrammarTable.get(key) != None:
                    for K in self.__GrammarTable[key]:
                        result = str(K[2]) + "-O-" + t[2]
                        if not (result in Given):
                            Given.append(result)
            else:
                key = self.__GrammarList[type][1][position]
                if self.__GrammarTable.get(key) == None:
```

```
                continue
            FollowString = self.__GrammarList[type][1][position + 1:] + t[2]
            FollowSet = []
            for FS in range(len(FollowString)):
                FollowTemp = self.__firstSet(FollowString[FS])
                findEmpty = False
                for FT in FollowTemp:
                    if FT != "ε":
                        if not (FT in FollowSet):
                            FollowSet.append(FT)
                    else:
                        findEmpty = True
                for k in self.__GrammarTable:
                    for i in range(len(self.__GrammarTable[k])):
                        self.__GrammarTable[k][i][1] = False
                if not findEmpty:
                    break
            for K in self.__GrammarTable[key]:
                for c in FollowSet:
                    if len(c) == 0:
                        print(FollowSet)
                    result = str(K[2]) + "-0-" + c
                    if not (result in Given):
                        Given.append(result)

def __CheckBelong(self, Item, CheckResult):
    count = 0
    for status in self.__Status:
        AllIn = True
        if len(Item) == len(status):
            for i in Item:
                if not (i in status):
                    AllIn = False
                    break
            if AllIn:
                CheckResult[0] = count
                break
        count += 1
def getTable(self):#生成分析表
    self.__getFirstSet()
    NowItem = ["0-0-#"]  # 使用"-"区分产生式左边,右边和展望符
   self.__Expand(NowItem)
    ItemQueue = deque()
    ItemQueue.append(copy.deepcopy(NowItem))
    self.__Status.append(copy.deepcopy(NowItem))
    self.__Action[str(0)] = {}
    self.__Goto[str(0)] = {}
    NumberCount = 0
    while len(ItemQueue) != 0:
        NowItem = ItemQueue.popleft()
        NowPointer = {}
        NowPointer["keys"] = []
        if self.__Action.get(str(NumberCount)) == None:
            self.__Action[str(NumberCount)] = {}
        if self.__Goto.get(str(NumberCount)) == None:
            self.__Goto[str(NumberCount)] = {}
        for item in NowItem:
            t = item.split("-")
            type = int(t[0])
            position = int(t[1])
            n = len(self.__GrammarList[type][1])
            child = self.__GrammarList[type][1]
            if child == "ε":
                self.__Action[str(NumberCount)][t[2]] = "ε" + t[0]
            elif position == n:
                self.__Action[str(NumberCount)][t[2]] = "r" + t[0]
            else:
                NowChar = self.__GrammarList[type][1][position]
```

```
                    if NowPointer.get(NowChar) == None:
                        NowPointer[NowChar] = []
                    NowPointer[NowChar].append(t[0] + "-" + str(position + 1) + "-" +
t[2])
                    if not (NowChar in NowPointer["keys"]):
                        NowPointer["keys"].append(NowChar)
            for npKey in NowPointer["keys"]:
                CreatedItem = copy.deepcopy(NowPointer[npKey])
                self.__Expand(CreatedItem)
                CheckResult = [-1]
                self.__CheckBelong(CreatedItem, CheckResult)
                NeedGotoStatus = -1
                if CheckResult[0] == -1:
                    self.__Status.append(copy.deepcopy(CreatedItem))
                    ItemQueue.append(copy.deepcopy(CreatedItem))
                    NeedGotoStatus = len(self.__Status) - 1
                else:
                    NeedGotoStatus = CheckResult[0]
                if self.__GrammarTable.get(npKey) != None:
                    self.__Goto[str(NumberCount)][npKey] = str(NeedGotoStatus)
                else:
                    self.__Action[str(NumberCount)][npKey] = "s" + str(NeedGotoStatus)
            NumberCount += 1
        self.__Action[str(1)]["#"] = "acc"
        Table = {}
        Table["Action"] = self.__Action
        Table["Goto"] = self.__Goto
        Table["GrammarList"] = self.__GrammarList
        return Table
```

3.4 以下为运行结果:

(样例可以直接分析空串)

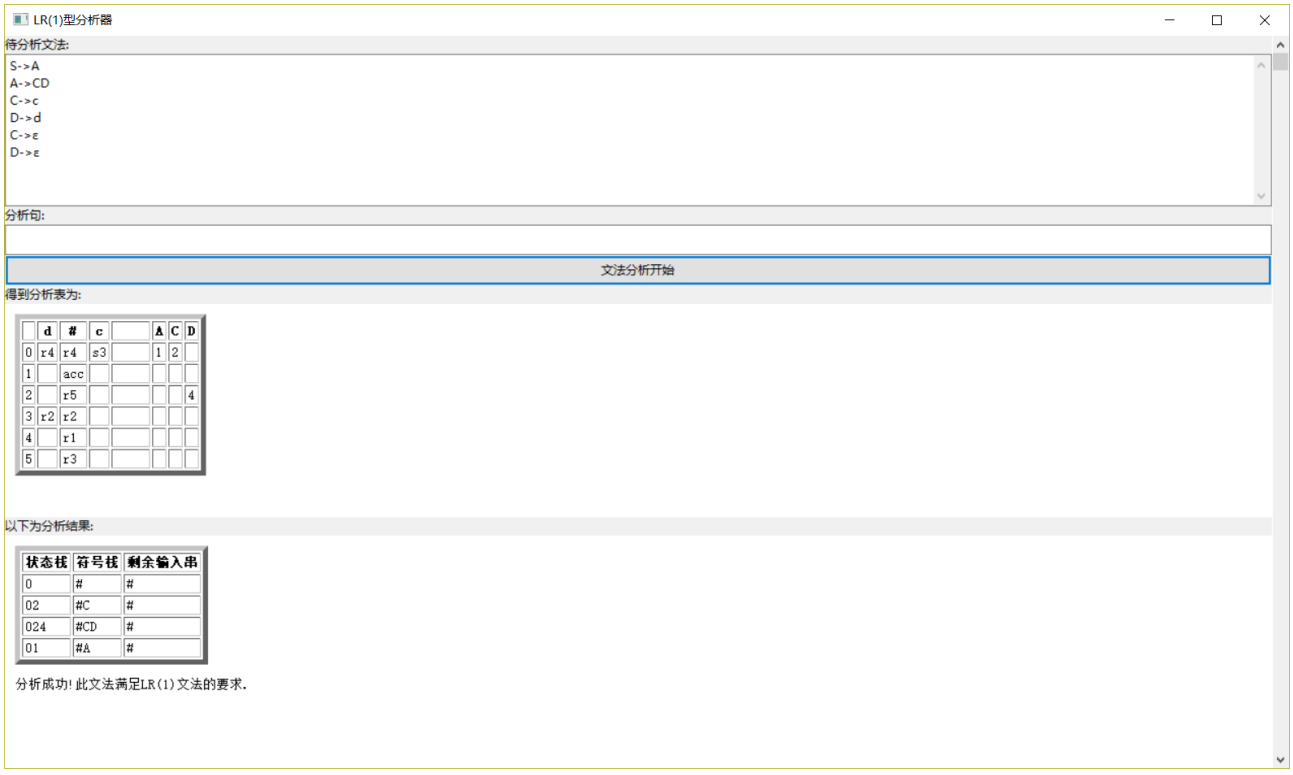

图 3.4-1 程序运行结果