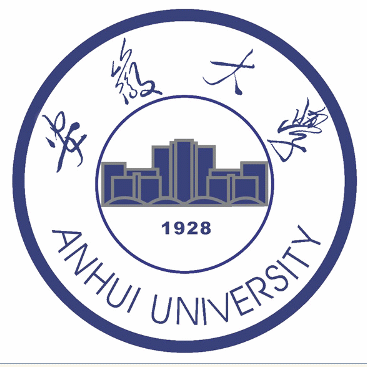
**《编译原理实验报告》**

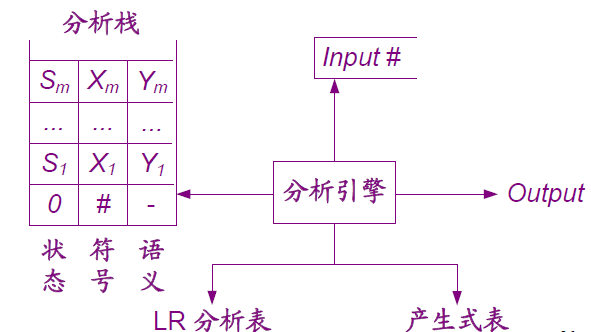
****

**论文名称：**　　　词法和语义分析程序

**年级专业：** 15级软件工程

**姓名学号：**何铭春 E21514033

## 一.实验内容



属性文法：

L →E

{print(E.val)}

E→E1+T

{ E.val :＝E1.val +T.val }

E→T

{ E.val :＝T.val }

T→T1\*digit

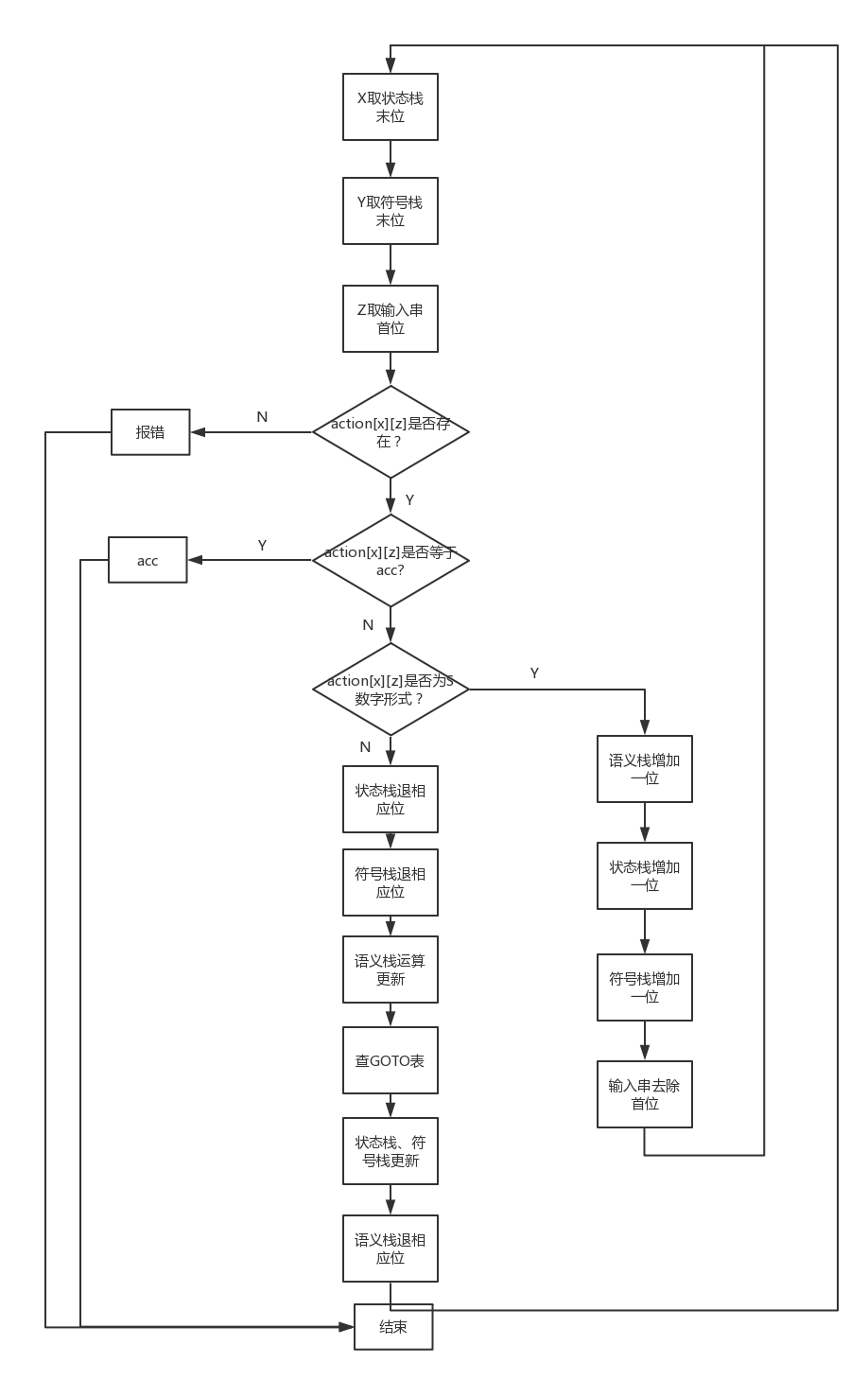
{ T.val :＝T1.val \* digit.lexval }

T→digit

{ T.val :＝digit.lexval }

根据以上属性文法判断是否符合语义

## 二.基本思想（流程图）



## 三.核心算法

终结符字典

terminal={"d":0,"+":1,"\*":2,"#":3}

非终结符字典

noterminal={"E":0,"T":1}

ACTION表和GO表

ACTION=[(["0"]\*4) for i in range(0,8)]

GOTO=[(["0"]\*2) for i in range(0,8)]

语法

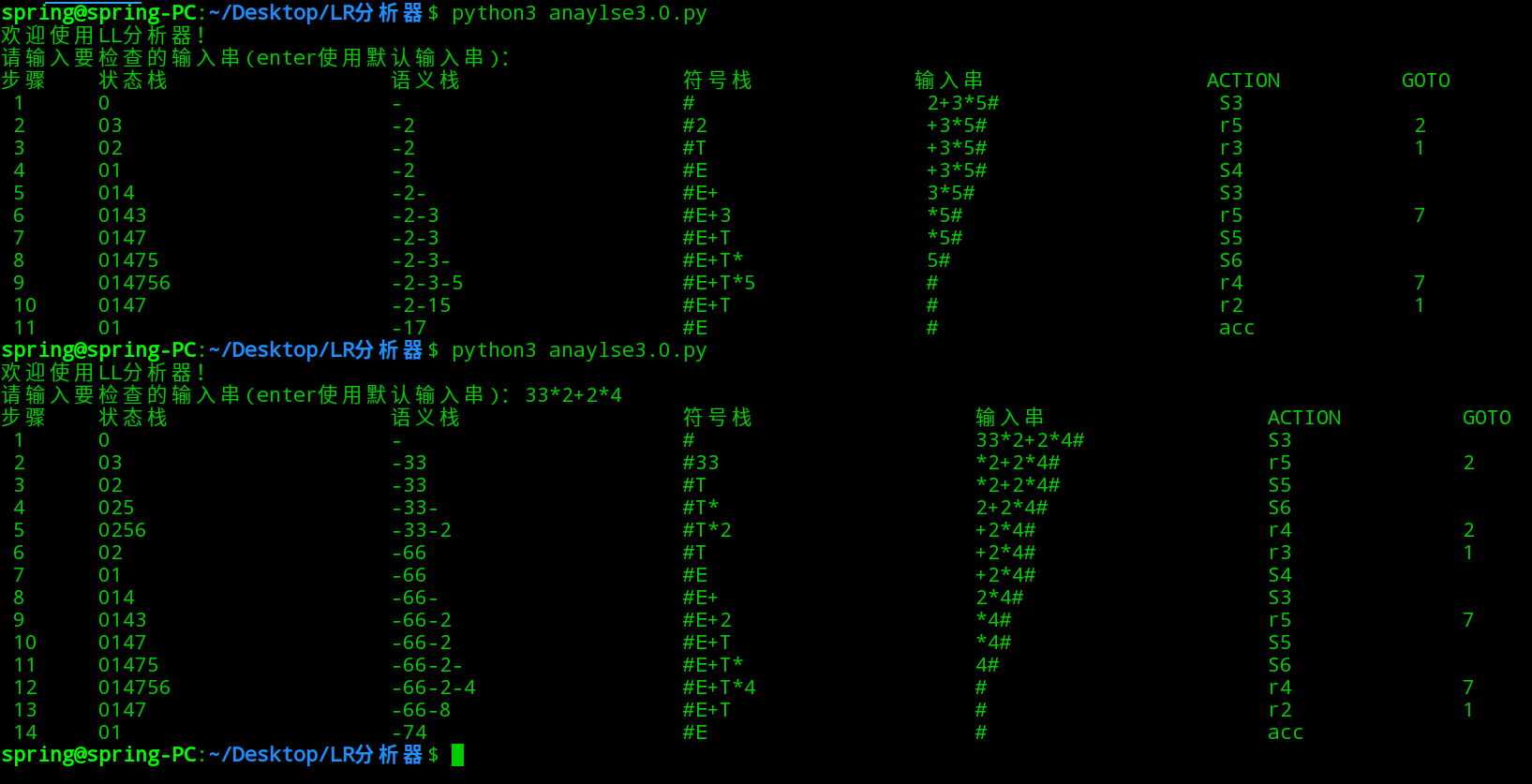
G={"r1":"L->E","r2":"E->E+T","r3":"E->T","r4":"T->T\*d","r5":"T->d"}

语义栈，一个用于显示，一个用于计算

mean=["-"]

extre=["-"]

## 四.实验结果



## 五.实验心得

1.在保留上次实验基础上，进行了改进。增加了语义栈的移入和归约处理。

2.采用函数归约，extra保存了运算符号，可以直接提取数字和符号进行运算。

例如第9步：语义栈为-2-3-5 extra保存为：-2+3\*5，遇到r4，直接提取出3\*5进行运算。

3.在语义栈的基础上，再次改进输入栈，不再使用字符串做栈，避免了33识别为两个3的问题。同步改进了符号栈，解决了33问题。

## 六．源代码

#一下进行数据结构初始化

terminal={"d":0,"+":1,"\*":2,"#":3}

noterminal={"E":0,"T":1}

ACTION=[(["0"]\*4) for i in range(0,8)]

GOTO=[(["0"]\*2) for i in range(0,8)]

G={"r1":"L->E","r2":"E->E+T","r3":"E->T","r4":"T->T\*d","r5":"T->d"}

ACTION[0][0]="S3"

ACTION[1][1]="S4"

ACTION[1][3]="acc"

ACTION[2][1]="r3"

ACTION[2][2]="S5"

ACTION[2][3]="r3"

ACTION[3][1]="r5"

ACTION[3][2]="r5"

ACTION[3][3]="r5"

ACTION[4][0]="S3"

ACTION[5][0]="S6"

ACTION[6][1]="r4"

ACTION[6][2]="r4"

ACTION[6][3]="r4"

ACTION[7][1]="r2"

ACTION[7][2]="S5"

ACTION[7][3]="r2"

GOTO[0][0]="1"

GOTO[0][1]="2"

GOTO[2][1]="3"

GOTO[4][1]="7"

#状态栈

# state="0"

#符号栈

# symbal="#"

#输入串

# inputStr="abbcde#"

# counter=1

def strToList(inputStr):

tempList=[]

word=""

for i in inputStr:

if i.isdigit():

word=word+i

else:

tempList.append(word)

word=""

tempList.append(i)

return tempList

def listToStr(inputList):

tempStr=""

for i in inputList:

tempStr=tempStr+str(i)

return tempStr

def calculate(list):

calculateStr='';

for i in list:

calculateStr=calculateStr+str(i)

return eval(calculateStr)

def LR(terminal,noterminal,ACTION,GOTO,inputList):

#计数变量

counter=1

#状态栈

state="0"

#符号栈

symbal=["#"]

mean=["-"]

extre=["-"]

print("步骤"+'\t'+"状态栈"+'\t\t\t'+"语义栈"+'\t\t\t'+"符号栈"+'\t\t\t'+"输入串"+'\t\t\t'+"ACTION"+'\t\t'+"GOTO")

while 1:

print("%- 4d"%(counter),end='\t')

print("%- 20s"%(state),end='\t')

print("%- 20s"%(listToStr(mean)),end='\t')

print("%- 20s"%(listToStr(symbal)),end='\t')

print("%- 20s"%(listToStr(inputList)),end='\t')

X=state[-1:]

Y=symbal[-1]

Z=inputList[0]

if Z.isdigit():

tempZ=Z

Z="d"

else:

tempZ=Z

#当action表中查不到值的时：

if ACTION[int(X)][int(terminal[Z])]=="0":

print("错误的输入串")

break

#字符串可以被接受时：

elif ACTION[int(X)][int(terminal[Z])]=="acc":

print("acc")

break

#遇到移入操作

elif "S" in ACTION[int(X)][int(terminal[Z])]:

state=state+ACTION[int(X)][int(terminal[Z])][-1:]

symbal.append(tempZ)

inputList=inputList[1:]

if tempZ.isdigit():

mean.append(tempZ)

else:

mean.append("-")

extre.append(tempZ)

print("%- 12s"%(ACTION[int(X)][int(terminal[Z])]))

#遇到归约操作

else:

print("%- 12s"%(ACTION[int(X)][int(terminal[Z])]),end='\t')

tempList=G[ACTION[int(X)][int(terminal[Z])]].split("->")

state=state[:len(state)-len(tempList[1])]

symbal=symbal[:len(symbal)-len(tempList[1])]

tempForMean=extre[-len(tempList[1]):]

calculateInt=calculate(tempForMean)

extre=extre[:len(mean)-len(tempList[1])]

mean=mean[:len(mean)-len(tempList[1])]

mean.append(calculateInt)

extre.append(calculateInt)

temp=state[-1:]

state=state+GOTO[int(temp)][int(noterminal[tempList[0]])]

print("%- 4s"%(GOTO[int(temp)][int(noterminal[tempList[0]])]))

symbal.append(tempList[0])

counter=counter+1

print("欢迎使用LL分析器！")

spring=input("请输入要检查的输入串(enter使用默认输入串)：")

if spring=="":

LR(terminal,noterminal,ACTION,GOTO,["2","+","3","\*","5","#"])

else:

spring=spring+"#"

spring=strToList(spring)

LR(terminal,noterminal,ACTION,GOTO,spring)