《编译原理》实验报告

学校： 安徽工业大学

姓名： 杨颂

学号： 169074209

**实验一 First集**

实验目的：

求产生式的First集。

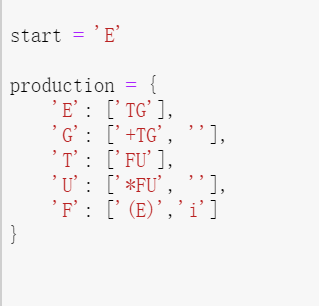
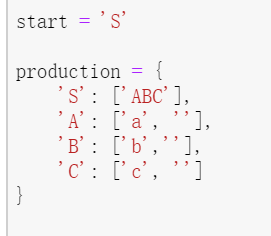
实验内容：

用高级语言实现对产生式First集的求解。

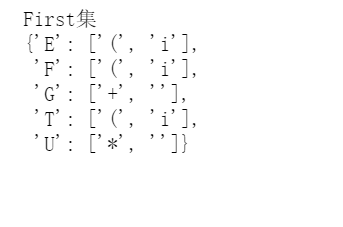
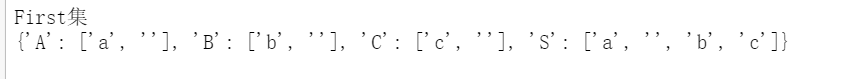
我使用了python语言实现了求产生式First集。

实验结果：

测试用例：

测试结果：

实验结果与预期的相符。

**实验二 Follow集**

实验目的：

求产生式的Follow集。

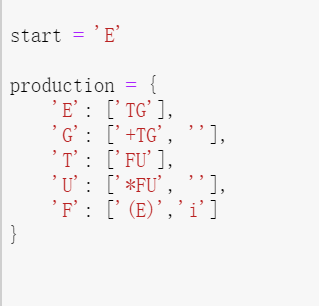
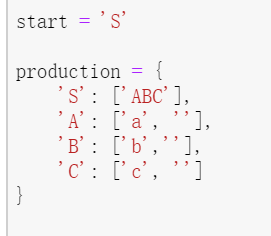
实验内容：

用高级语言实现对产生式Follow集的求解。

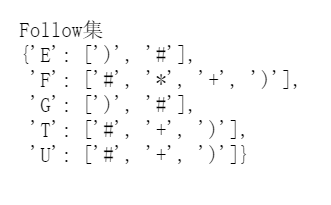
我使用了python语言实现了求产生式Follow集。

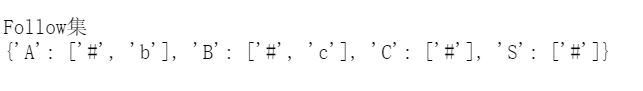
实验结果：

测试用例：

测试结果：





实验结果与预期的相符。

**实验三 Select**

实验目的：

求产生式的Select集。

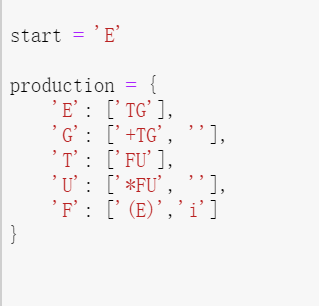
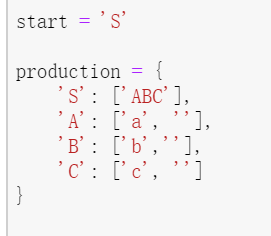
实验内容：

用高级语言实现对产生式Select集的求解。

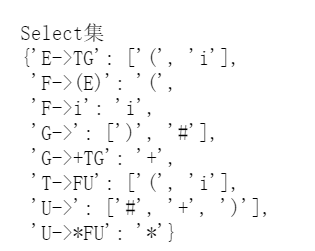
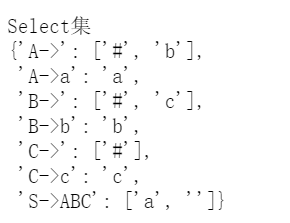
我使用了python语言实现了求产生式Select集。

实验结果：

测试用例：

测试结果：

实验结果与预期的相符。

实验心得

**本次实验让我深刻理解了产生式求First,Follow,Select集的求解。**

**First集：**

**先将产生式右部开始的终结符加入到，产生式左部的First集中。下一步再根据已得的和相应的产生式将他们的First集加入到右部为此非终结符开始的左部非终结符First集中。若非终结符可以推导出为空，则需要考虑此非终结符后的非终结符的First集。**

**Follow集：**

**先将产生式的开始符的Follow集加入#，再寻找出所有产生式右部可以直接得出的可能的Follow集。再根据左部的非终结符是否可以推导为空将相应的产生式的左部的Follow集加入相应的右部非终结符的First集或是Follow集。**

**再次循环完善每个Follow集，直到没有新的元素出现位置。**

**Select集：**

**根据所有可能的产生式一步一步推出，寻找产生式左部可能推导出的式子的第一个终结符，将此终结符加入Select集中。如果左部为空，则把左部的Follow集加入Select集中。否则将右部第一个非终结符的First集加入Select集中。**

附录

核心代码：

from pprint import pprint

from copy import deepcopy

class RecAnalyzer:

def \_\_init\_\_(self, start, over, production):

self.start = start

self.over = over

self.production = production

self.nontermainals = production.keys()

for nontermainal in self.nontermainals:

for right in self.production[nontermainal]:

if right != '' and right[0] == nontermainal:

exit('left recursive!')

self.first = {nontermainal: {} for nontermainal in self.nontermainals}

self.follow = {nontermainal: set() for nontermainal in self.nontermainals}

self.get\_first\_follow()

# 求first的函数

def get\_first(self, nontermainal):

ret\_dict = {}

for right in self.production[nontermainal]:

if (nontermainal, right) in self.first\_first:

ret\_dict = self.first[nontermainal]

continue

if right != '':

if right[0] in self.over:

ret\_dict.update({right[0]: right})

else:

for sign in right:

if sign in self.nontermainals:

first\_ = self.first[sign]

ret\_dict.update({key: right for key in first\_.keys()})

if '' not in first\_.keys():

break

else:

ret\_dict.update({'': ''})

return ret\_dict

# 求first集和follow集

def get\_first\_follow(self):

# 求first第一轮，产生式右部首字符为终结符号

self.first\_first = set()

for nontermainal in self.nontermainals:

for right in self.production[nontermainal]:

if right != '' and right[0] in self.over:

self.first[nontermainal][right[0]] = right

self.first\_first.add((nontermainal, right))

# 求first第二轮

while True:

old\_first = deepcopy(self.first)

for nontermainal in self.nontermainals:

self.first[nontermainal].update(self.get\_first(nontermainal))

if old\_first ==self.first:

break

# 起始符号follow集

self.follow[self.start].add('#')

# 循环直到follow集不再变化

while True:

old\_follow = deepcopy(self.follow)

for nontermainal in self.nontermainals:

for right in self.production[nontermainal]:

for i, sign in enumerate(right):

if sign in self.over:

continue

if i == len(right) - 1:

self.follow[sign] |= self.follow[nontermainal]

elif right[i + 1] in self.over:

self.follow[sign].add(right[i + 1])

else:

next\_set = {key for key in self.first[right[i + 1]].keys()}

next\_set\_without\_null = {key for key in self.first[right[i + 1]].keys() if key != ''}

self.follow[sign] |= next\_set\_without\_null

if '' in next\_set:

self.follow[sign] |= self.follow[nontermainal]

if old\_follow == self.follow:

break

for nontermainal in self.nontermainals:

if '' in self.first[nontermainal]:

self.first[nontermainal][''] = self.follow[nontermainal] - {key for key in self.first[nontermainal].keys()}

print('First集')

true\_first={}

for key,values in self.first.items():

true\_first[key]=list(values.keys())

pprint(true\_first)

print('\n\n\nFollow集')

true\_follow={}

for key,values in self.follow.items():

true\_follow[key]=list(values)

pprint(true\_follow)

select={}

for key,values in production.items():

for value in values:

i=key+'->'+value

if value=='':

select[i]=true\_follow[key]

elif value[0] in over:

select[i]=value[0]

elif value[0] in production.keys():

select[i]=true\_first[value[0]]

print('\n\n\nSelect集')

pprint(select)

start = 'S'

production = {

'S': ['ABC'],

'A': ['a', ''],

'B': ['b',''],

'C': ['c', '']

}

a=set()

for i in production.values():

for j in i:

for k in list(j):

a.add(k)

b=set(production.keys())

over=(list)(a-b)

rec\_analyzer = RecAnalyzer(start=start, over=over, production=production)