

24-Feb-22

Experiment 5 - FIRST and FOLLOW Computation

Dhawal Patil

RA1911003010575

CSE A2

Experiment 5

Aim:

To write a program to perform first and follow using Python language.

Algorithm:

For computing the first:

1. If X is a terminal then $\text{FIRST}(X) = \{X\}$

Example: $F \rightarrow I \mid id$

We can write it as $\text{FIRST}(F) \rightarrow \{ (, id)$

2. If X is a non-terminal like $E \rightarrow T$ then to get $\text{FIRST}(E)$ substitute T with other productions until you get a terminal as the first symbol
3. If $X \rightarrow \epsilon$ then add ϵ to $\text{FIRST}(X)$.

For computing the follow:

1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side).
2. (a) If that non-terminal (S,A,B...) is followed by any terminal (a,b...,*,+,(),...) , then add that terminal into the FOLLOW set.
(b) If that non-terminal is followed by any other non-terminal then add FIRST of other nonterminal into the FOLLOW set.

Code:

```
gram = {  
    "E":["E+T","T"],  
    "T":["T*F","F"],  
    "F":["(E)","id"]  
}
```

```
def removeDirectLR(gramA, A):  
    temp = gramA[A]  
    tempCr = []  
    tempInCr = []  
    for i in temp:  
        if i[0] == A:  
            tempInCr.append(i[1:]+[A+""])  
        else:
```

```

        tempCr.append(i+[A+""])
    templnCr.append(["e"])
    gramA[A] = tempCr
    gramA[A+""] = templnCr
    return gramA

```

```

def checkForIndirect(gramA, a, ai):
    if ai not in gramA:
        return False
    if a == ai:
        return True
    for i in gramA[ai]:
        if i[0] == ai:
            return False
        if i[0] in gramA:
            return checkForIndirect(gramA, a, i[0])
    return False

```

```

def rep(gramA, A):
    temp = gramA[A]
    newTemp = []
    for i in temp:
        if checkForIndirect(gramA, A, i[0]):
            t = []
            for k in gramA[i[0]]:
                t=[]
                t+=k
                t+=i[1:]
            newTemp.append(t)
        else:
            newTemp.append(i)
    gramA[A] = newTemp
    return gramA

```

```

def rem(gram):
    c = 1
    conv = {}
    gramA = {}
    revconv = {}
    for j in gram:
        conv[j] = "A"+str(c)
        gramA["A"+str(c)] = []
        c+=1

```

```

for i in gram:
    for j in gram[i]:
        temp = []
        for k in j:
            if k in conv:
                temp.append(conv[k])
            else:
                temp.append(k)
        gramA[conv[i]].append(temp)
for i in range(c-1,0,-1):
    ai = "A"+str(i)
    for j in range(0,i):
        aj = gramA[ai][0][0]
        if ai!=aj :
            if aj in gramA and checkForIndirect(gramA,ai,aj):
                gramA = rep(gramA, ai)
for i in range(1,c):
    ai = "A"+str(i)
    for j in gramA[ai]:
        if ai==j[0]:
            gramA = removeDirectLR(gramA, ai)
            break
op = {}
for i in gramA:
    a = str(i)
    for j in conv:
        a = a.replace(conv[j],j)
    revconv[i] = a
for i in gramA:
    l = []
    for j in gramA[i]:
        k = []
        for m in j:
            if m in revconv:
                k.append(m.replace(m,revconv[m]))
            else:
                k.append(m)
        l.append(k)
    op[revconv[i]] = l
return op
result = rem(gram)

def first(gram, term):

```

```

a = []
if term not in gram:
    return [term]
for i in gram[term]:
    if i[0] not in gram:
        a.append(i[0])
    elif i[0] in gram:
        a += first(gram, i[0])
return a

```

```

firsts = {}
for i in result:
    firsts[i] = first(result,i)
print(f'First({i}):',firsts[i])

```

```

def follow(gram, term):
    a = []
    for rule in gram:
        for i in gram[rule]:
            if term in i:
                temp = i
                indx = i.index(term)
                if indx+1!=len(i):
                    if i[-1] in firsts:
                        a+=firsts[i[-1]]
                    else:
                        a+=i[-1]]
                else:
                    a+=["e"]
            if rule != term and "e" in a:
                a+= follow(gram,rule)
    return a

```

```

follows = {}
for i in result:
    follows[i] = list(set(follow(result,i)))
    if "e" in follows[i]:
        follows[i].pop(follows[i].index("e"))
    follows[i]+=["$"]
print(f'Follow({i}):',follows[i])

```

```
main.py
1- gram = {
2-     "E":["E+T","T"],
3-     "T":["T*F","F"],
4-     "F":["(E)","id"]
5- }
6-
7- def removeDirectLR(gramA, A):
8-     temp = gramA[A]
9-     tempCr = []
10-    tempInCr = []
11-    for i in temp:
12-        if i[0] == A:
13-            tempInCr.append(i[1:]+[A+""])
14-        else:
15-            tempCr.append(i+[A+""])
16-    tempInCr.append(["e"])
17-    gramA[A] = tempCr
18-    gramA[A+""] = tempInCr
19-    return gramA
20-
21- def checkForIndirect(gramA, a, ai):
22-     if ai not in gramA:
23-         return False
24-     if a == ai:
25-         return True
26-     for i in gramA[ai]:
27-         if i[0] == ai:
28-             return False
29-         if i[0] in gramA:
30-             return checkForIndirect(gramA, a, i[0])
31-     return False
32-
33- def rep(gramA, A):
34-     temp = gramA[A]
35-     newTemp = []
36-     for i in temp:
37-         if checkForIndirect(gramA, A, i[0]):
38-             t = []
39-             for k in gramA[i[0]]:
40-                 t=[]
41-                 t+=k
42-                 t+=i[1:]
43-                 newTemp.append(t)
44-         else:
45-             newTemp.append(i)
46-     gramA[A] = newTemp
47-     return gramA
48-
49- def rem(gram):
50-     c = 1
51-     conv = {}
52-     gramA = {}
53-     revconv = {}
54-     for j in gram:
55-         conv[j] = "A"+str(c)
56-         gramA["A"+str(c)] = []
57-         c+=1
58-     for i in gram:
59-         for j in gram[i]:
60-             temp = []
61-             for k in j:
62-                 if k in conv:
63-                     temp.append(conv[k])
64-                 else:
65-                     temp.append(k)
66-             gramA[conv[i]].append(temp)
67-     for i in range(c-1,0,-1):
68-         ai = "A"+str(i)
69-         for j in range(0,i):
70-             aj = gramA[ai][0][0]
71-             if ai!=aj :
72-                 if aj in gramA and checkForIndirect(gramA,ai,aj):
```

```

73         gramA = rep(gramA, ai)
74     for i in range(1,c):
75         ai = "A"+str(i)
76         for j in gramA[ai]:
77             if ai==j[0]:
78                 gramA = removeDirectLR(gramA, ai)
79                 break
80     op = {}
81     for i in gramA:
82         a = str(i)
83         for j in conv:
84             a = a.replace(conv[j],j)
85         revconv[i] = a
86     for i in gramA:
87         l = []
88         for j in gramA[i]:
89             k = []
90             for m in j:
91                 if m in revconv:
92                     k.append(m.replace(m,revconv[m]))
93                 else:
94                     k.append(m)
95             l.append(k)
96     op[revconv[i]] = l
97     return op
98 result = rem(gram)
99
100 def first(gram, term):
101     a = []
102     if term not in gram:
103         return [term]
104     for i in gram[term]:
105         if i[0] not in gram:
106             a.append(i[0])
107         elif i[0] in gram:
108             a += first(gram, i[0])
109     return a
110
111 firsts = {}
112 for i in result:
113     firsts[i] = first(result,i)
114     print(f'First({i}):',firsts[i])
115
116 def follow(gram, term):
117     a = []
118     for rule in gram:
119         for i in gram[rule]:
120             if term in i:
121                 temp = i
122                 indx = i.index(term)
123                 if indx+1==len(i):
124                     if i[-1] in firsts:
125                         a+=firsts[i[-1]]
126                 else:
127                     a+=i[indx+1]
128             else:
129                 a+="'e'"
130             if rule != term and "e" in a:
131                 a+= follow(gram,rule)
132     return a
133
134 follows = {}
135 for i in result:
136     follows[i] = list(set(follow(result,i)))
137     if "e" in follows[i]:
138         follows[i].pop(follows[i].index("e"))
139     follows[i]+="$"
140     print(f'Follow({i}):',follows[i])

```

Output:

```

First(E): ['(', 'i']
First(T): ['(', 'i']
First(F): ['(', 'i']
First(E): ['+', 'e']
First(T): ['*', 'e']
Follow(E): [')', '$']
Follow(T): [')', '+', '$']
Follow(F): ['*', ')', '+', '$']
Follow(E): [')', '$']
Follow(T): [')', '+', '$']

...Program finished with exit code 0
Press ENTER to exit console.

```

Result:

A program for FIRST and FOLLOW computation was run successfully.