AUMKAR M GADEKAR

2017130023

TE COMPS

Experiment No.5

Aim:- Write a program to find first and follow sets for the given grammar. Program should accept the grammar from the user and output the first and follow sets for each of the grammar symbols.

Theory:-

First(α) is a set of terminals that begins strings derived from α. If α=>ε then ε is also in First(ε).

In predictive parsing when we have A-> α|β, if First(α) and First(β) are disjoint sets then we can select appropriate A-production by looking at the next input.

Follow(A), for any nonterminal A, is set of terminals a that can appear immediately after A in some sentential form

If we have S => αAaβ for some α and β then a is in Follow(A)

If A can be the rightmost symbol in some sentential form, then $ is in Follow(A)

Finding First:-

First(x) for all grammar symbols X
Apply following rules:

1. If X is terminal, FIRST(X) = {X}.
2. If X → ε is a production, then add ε to FIRST(X).
3. If X is a non-terminal, and X→ Y1Y2… Yk is a production, and ε is in all of FIRST(Y1), …, FIRST(Yk), then add ε to FIRST(X).
4. If X is a non-terminal, and X→ Y1Y2… Yk is a production, then add a to FIRST(X) if for some i, a is in FIRST(Yi), and ε is in all of FIRST(Y1), …, FIRST(Yi-1).

Applying rules 1 and 2 is obvious. Applying rules 3 and 4 for FIRST(Y1Y2… Yk) can be done as follows:

Add all the non-ε symbols of FIRST(Y1) to FIRST(Y1 Y2 … Yk). If ε ∈ FIRST(Y1), add all the non-ε symbols of FIRST(Y2). If ε ∈ FIRST(Y1) and ε ∈ FIRST(Y2), add all the non-ε symbols of FIRST(Y3), and so on. Finally, add ε to FIRST(Y1 Y2 … Yk) if ε ∈ FIRST(Yi), for all $1 \leq i \leq k$.

Example:
Consider the following grammar.
E → E + T | T

T → T * F | F
F → (E) | id
Grammar after removing left recursion:
E → TX
X → +TX | ε
T → FY
Y → *FY | ε
F → (E) | id
For the above grammar, following the above rules, the FIRST sets could be computed as follows:
FIRST(E) = FIRST(T) = FIRST(F) = {(, id}
FIRST(X) = {+, ε}
FIRST(Y) = {*, ε}

Finding Follow:-
To compute Follow(A) for all nonterminals A, apply following rules until nothing can be added to any follow set:
1. Place $ in Follow(S) where S is the start symbol
2. If there is a production A-> αBβ then everything in First(β) except ε is in Follow(B).
3. If there is a production A->B or a production A->αBβ where First(β) contains ε, then everything in Follow(A) is in Follow(B)

Example:-
E → TX
X → +TX | ε
T → FY
Y → *FY | ε
F → (E) | id

FOLLOW(E) = {$,) }
FOLLOW(X) = {$,) }
FOLLOW(T) = {+, $,) }
FOLLOW(Y) = {+, $,) }
FOLLOW(F) = {*, +, $,) }
.

Code:-

```
n=int(input("Enter Number of productions "))
Sym=[]
V=[]
T=[]
P=[]
```

```
for i in range(n):
    prod=input("Enter production of the form V->(V U T) ")
    v=prod[0]
    if v not in V:
        V.append(v)
        P.append([])
    if(v not in Sym):
        Sym.append(v)
    ind=V.index(v)
    for j in range(3,len(prod)):
        if(prod[j] not in Sym and prod[j]!='|' and prod[j]!='ε'):
            Sym.append(prod[j])
    lst=prod[3:].split('|')
    for z in lst:
        P[ind].append(z)
for j in Sym:
    if j not in V:
        T.append(j)
First=[]
for i in range(len(Sym)):
    First.append([])
Sym=[]
Sym.extend(T)
Sym.extend(V)
eps=[]
t=len(T)
for i in range(t):
    First[i].append(Sym[i])
for j in range(t,len(Sym)):
    for x in P[j-t]:
        if(x[0] in T):
            First[j].append(x[0])

    if('ε' in P[j-t]):
        First[j].append('ε')
        eps.append(V[j-t])
change=1
```

```
while(change!=0):
    change=0
    for j in range(t,len(Sym)):
        if(Sym[j] not in eps):
            for p in P[j-t]:
                flag=0
                for k in p:
                    if(k not in eps):
                        flag=1
                        break
                if(flag==0):
                    First[j].append('ε')
                    eps.append(V[j-t])
                    change+=1
                    break
change=1
while(change!=0):
    change=0
    for j in range(t,len(Sym)):
         for p in P[j-t]:
           if(p!='ε'):
            k=0
            idx=Sym.index(p[k])
            for z in First[idx]:
                if(z not in First[j] and z!='ε'):
                    First[j].append(z)
                    change+=1
            while('ε' in First[idx] and k<len(p)):
                k+=1
                idx=Sym.index(p[k])
                for z in First[idx]:
                    if (z not in First[j] and z!='ε'):
                        First[j].append(z)
                        change+=1
print("Symbol   First-Pos")
for i in range(len(V)):
    print(V[i],"      ",First[t+i])
```

```
Follow=[]
for i in range(len(V)):
    Follow.append([])
Follow[0].append('$')
for i in P:
    for j in i:
        for k in range(len(j)-1):
            var=j[k]
            if(var in V):
                idx=V.index(var)
                l=k+1
                while(l<len(j)):
                    if(j[l] in T):
                        Follow[idx].append(j[l])
                        break
                    else:
                        ind=Sym.index(j[l])
                        for x in First[ind]:
                            if(x not in Follow[idx] and x!='ε'):
                                Follow[idx].append(x)
                        if(j[l] not in eps):
                            break
                    l+=1
change=1
while(change!=0):
    change=0
    for i in range(len(P)):
        for j in P[i]:
            l=len(j)-1
            if(j[l] in V):
                ind=Sym.index(j[l])-t
                for x in Follow[i]:
                    if(x not in Follow[ind] and x!='ε'):
                        Follow[ind].append(x)
                        change+=1
                l-=1
                while(l>=0):
```

```
                    if(j[l+1] in eps):
                        ind=Sym.index(j[l])-t
                        for x in Follow[i]:
                            if(x not in Follow[ind] and x!='ε'):
                                Follow[ind].append(x)
                                change+=1
                        l-=1
                    else:
                        break
print("Symbol    Follow-Pos")
for i in range(len(Follow)):
    print(V[i],"       ",Follow[i])
```

Output:-

```
C:\Users\aumkar\Downloads>python exp5.py
Enter Number of productions 6
Enter production of the form V->(V U T) S->ABCDE
Enter production of the form V->(V U T) A->a|ε
Enter production of the form V->(V U T) B->b|ε
Enter production of the form V->(V U T) C->c
Enter production of the form V->(V U T) D->d|ε
Enter production of the form V->(V U T) E->e|ε
Symbol   First-Pos
S        ['a', 'b', 'c']
A        ['a', 'ε']
B        ['b', 'ε']
C        ['c']
D        ['d', 'ε']
E        ['e', 'ε']
Symbol    Follow-Pos
S         ['$']
A         ['b', 'c']
B         ['c']
C         ['d', 'e', '$']
D         ['e', '$']
E         ['$']
```

```
C:\Users\aumkar\Downloads>python exp5.py
Enter Number of productions 5
Enter production of the form V->(V U T) E->TG
Enter production of the form V->(V U T) G->+TG|ε
Enter production of the form V->(V U T) T->FH
Enter production of the form V->(V U T) H->*FH|ε
Enter production of the form V->(V U T) F->i|(E)
Symbol   First-Pos
E        ['i', '(']
G        ['+', 'ε']
T        ['i', '(']
H        ['*', 'ε']
F        ['i', '(']
Symbol    Follow-Pos
E         ['$', ')']
G         ['$', ')']
T         ['+', '$', ')']
H         ['+', '$', ')']
F         ['*', '+', '$', ')']
```

Conclusion:-

Thus the given problem can determine the first and follow pos for all non-terminals in a given grammar.