Name: Varun Khadayate	Subject: Compiler Design
Roll No: A016	Date of Submission: 3rd September 2021

Aim

To find the First and Follow of the given grammar

Program logic

First Function-

First(α) is a set of terminal symbols that begin in strings derived from α.

Example-

Consider the production rule-

$$A \rightarrow abc / def / ghi$$

Then, we have-

$$First(A) = \{ a , d , g \}$$

Rules For Calculating First Function-

Rule-01:

For a production rule $X \rightarrow \in$,

$$First(X) = \{ \in \}$$

Rule-02:

For any terminal symbol 'a',

$$First(a) = \{ a \}$$

Rule-03:

For a production rule $X \rightarrow Y_1Y_2Y_3$,

Calculating First(X)

- If $\in \notin First(Y_1)$, then $First(X) = First(Y_1)$
- If $\in \in First(Y_1)$, then $First(X) = \{ First(Y_1) - \in \} \cup First(Y_2Y_3)$

Calculating First(Y2Y3)

- If $\in \notin First(Y_2)$, then $First(Y_2Y_3) = First(Y_2)$
- If $\in \in First(Y_2)$, then $First(Y_2Y_3) = \{ First(Y_2) - \in \} \cup First(Y_3)$

Similarly, we can make expansion for any production rule $X \rightarrow Y_1Y_2Y_3.....Y_n$.

Follow Function-

Follow(α) is a set of terminal symbols that appear immediately to the right of α.

Rules For Calculating Follow Function-

Rule-01:

For the start symbol S, place $ in Follow(S).

Rule-02:

For any production rule A → αB,

$$Follow(B) = Follow(A)$$

Rule-03:

For any production rule A → αBβ,

- If ∈ ∉ First(β), then Follow(B) = First(β)
- If ∈ ∈ First(β), then Follow(B) = { First(β) – ∈ } ∪ Follow(A)

Important Notes-

NOTE-01:

- ∈ may appear in the first function of a non-terminal.
- ∈ will never appear in the follow function of a non-terminal.

NOTE-02:

- Before calculating the first and follow functions, eliminate **Left Recursion** from the grammar, if present.

NOTE-03:

- We calculate the follow function of a non-terminal by looking where it is present on the RHS of a production rule.

PRACTICE PROBLEMS BASED ON CALCULATING FIRST AND FOLLOW-

Problem-01:

Calculate the first and follow functions for the given grammar-

$$S \rightarrow aBDh$$

$$B \rightarrow cC$$

$$C \rightarrow bC / \in$$

$$D \rightarrow EF$$

$$E \rightarrow g / \in$$

$$F \rightarrow f / \in$$

Solution-

The first and follow functions are as follows-

First Functions-

- First(S) = { a }
- First(B) = { c }
- First(C) = { b , ∈ }
- First(D) = { First(E) – ∈ } ∪ First(F) = { g , f , ∈ }
- First(E) = { g , ∈ }
- First(F) = { f , ∈ }

Follow Functions-

- Follow(S) = { $ }
- Follow(B) = { First(D) – ∈ } ∪ First(h) = { g , f , h }
- Follow(C) = Follow(B) = { g , f , h }
- Follow(D) = First(h) = { h }
- Follow(E) = { First(F) – ∈ } ∪ Follow(D) = { f , h }
- Follow(F) = Follow(D) = { h }

Lab Assignment

What is First and Follow?

First(α) is a set of terminal symbols that begin in strings derived from α.

Follow(α) is a set of terminal symbols that appear immediately to the right of α.

Specify rules for first and follow?

Rules For Calculating First Function-

Rule-01:

For a production rule $X \rightarrow \in$,

$$First(X) = \{ \in \}$$

Rule-02:

For any terminal symbol ‘a’,

$$First(a) = \{ a \}$$

Rule-03:

For a production rule $X \rightarrow Y_1Y_2Y_3$,

Calculating First(X)

- If $\in \notin First(Y_1)$, then $First(X) = First(Y_1)$
- If $\in \in First(Y_1)$, then $First(X) = \{ First(Y_1) - \in \} \cup First(Y_2Y_3)$

Calculating First(Y2Y3)

- If $\in \notin First(Y_2)$, then $First(Y_2Y_3) = First(Y_2)$
- If $\in \in First(Y_2)$, then $First(Y_2Y_3) = \{ First(Y_2) - \in \} \cup First(Y_3)$

Similarly, we can make expansion for any production rule $X \rightarrow Y_1Y_2Y_3.....Y_n$.

Rules For Calculating Follow Function-

Rule-01:

For the start symbol S, place $ in Follow(S).

Rule-02:

For any production rule $A \rightarrow \alpha B$,

$$Follow(B) = Follow(A)$$

Rule-03:

For any production rule $A \rightarrow \alpha B\beta$,

- If $\in \notin First(\beta)$, then $Follow(B) = First(\beta)$
- If $\in \in First(\beta)$, then $Follow(B) = \{ First(\beta) - \in \} \cup Follow(A)$

Define algorithm for first and follow?

Algorithm for calculating First set

- if α is a terminal, then FIRST(α) = { α }.
- if α is a non-terminal and α → Ɛ is a production, then FIRST(α) = { Ɛ }.
- if α is a non-terminal and α → γ1 γ2 γ3 … γn and any FIRST(γ) contains t then t is in FIRST(α).

Algorithm for calculating Follow set:

- if α is a start symbol, then FOLLOW() = $
- if α is a non-terminal and has a production α → AB, then FIRST(B) is in FOLLOW(A) except Ɛ.
- if α is a non-terminal and has a production α → AB, where B Ɛ, then FOLLOW(A) is in FOLLOW(α).

Lab Assignment Program

Write a program to implement first and follow from given grammar.

Code

```
gram = {
    "S":["aBDh"],
    "B":["cC"],
    "C":["bC","e"],
    "D":["EF"],
    "E":["g","e"],
    "F":["f","e"]
}

def removeDirectLR(gramA, A):
    """gramA is dictonary"""
    temp = gramA[A]
    tempCr = []
    tempInCr = []
    for i in temp:
        if i[0] == A:
            tempInCr.append(i[1:]+[A+"'"])
        else:
            tempCr.append(i+[A+"'"])
    tempInCr.append(["e"])
    gramA[A] = tempCr
    gramA[A+"'"] = tempInCr
    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 of ({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 = {}
print("\n")
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 of ({i}):',follows[i])
```

Output

```
PS E:\TY\CD> & e:/TY/CD/venv/Scripts/python.exe "e:/TY/CD/Practical 5 and 6/prac_5_6_first_n_follow.py"
First of (S): ['a']
First of (B): ['c']
First of (C): ['b', 'e']
First of (D): ['g', 'e']
First of (E): ['g', 'e']
First of (F): ['f', 'e']


Follow of (S): ['$']
Follow of (B): ['h', '$']
Follow of (C): ['h', '$']
Follow of (D): ['h', '$']
Follow of (E): ['f', 'h', '$']
Follow of (F): ['h', '$']
PS E:\TY\CD>
```

Conclusion

Hence, we were able to implement first and follow of the given grammar.