18th February 2022

Experiment 5 : Computation of FIRST() and FOLLOW()

Aim: To compute the FIRST() and FOLLOW() sets using

Algorithm

1. Start
2. Get input from user for the number of terminals, non-terminals, and production rules.
3. Get input from user for all production rules in the form of A → B | cD | @ where @ denotes epsilon (∈)
4. Split the production rules along '|' and append to dictionary. For example, for above rule: { A : [B], [cD], [@] }

ALGORITHM FOR FIRST()

1. For every non terminal symbol, call FIRST() with that symbol as input string
2. In FIRST(), create a set to store the first set.
3. If input string is non terminal, get the alternate production rules from production dictionary and recursively call FIRST() for each alternative.
4. Append the results of recursive call to first set.

5. If the string is terminal or epsilon, append the string itself to the first set.
6. Else, call FIRST() for first character of input string.
7. If that leads to epsilon, append the results to original first set and continue calling FIRST() for the next character.
8. Break out of loop when there are no more recursive calls.
9. End FIRST().

ALGORITHM FOR FOLLOW()

1. Add $ to follow set for starting symbol.
2. For every non terminal symbol, call the FOLLOW() function.
3. Create an output follow set to store follow set.
4. Get all productions from production dict
5. For every non terminal, get right hand side (RHS) of ~~not~~ non terminal from production dict.
6. i) If character is non terminal, get the string which follows the character (For A → aBc, following string is c as it follows B)

① If following string is empty or epsilon, recursively call ~~NON~~ FOLLOW for the original non terminal and appends results ~~to~~ output set.

ii) If the following string is non empty, call FIRST() for the following string.
   ① If that results in epsilon, place the results without epsilon in output follow set and call FOLLOW for the original non terminal.
   ② Else, place results in output follow set.

7. Return the output follow set
8. END.

MANUAL WORKING

FIRST rules:

1. If X is a terminal, then FIRST(X) = {X}
2. If $X \to \epsilon$ is a production, then add ϵ to FIRST(X)
3. If X is a non-terminal and $X \to Y_1 Y_2 \dots Y_k$ is a production, then add FIRST(Y_1) to FIRST(X). If Y_1 derives ϵ, then add FIRST(Y_2) to FIRST(X).

Sample I/P and O/P:

Input production rules:

$E \to TB$

$B \to +TB \mid @$

$T \to FY$

$Y \to *FY \mid @$

$F \to (E) \mid a$

Working:

FIRST(a) = {a}

FIRST(+) = {+}

FIRST(*) = {*}

FIRST(() = {(}

FIRST()) = {)}

FIRST (E) = FIRST (T) = { (, a }

FIRST (B) = FIRST (+) ∪ {@} = { +, @ }

FIRST (T) = FIRST (F) = { (, a }

FIRST (Y) = FIRST (*) ∪ {@} = { *, @ }

FIRST (F) = FIRST (() ∪ FIRST (a) = { (, a }

FOLLOW rules :

1. For FOLLOW (starting symbol) place $ where $ is input end marker.
2. If there is a production A → ∝Bβ, then everything in FIRST (β) except ∈ is in FOLLOW (B)
3. If there is a production A → ∝B or A → ∝Bβ whe FIRST (β) contains ∈, then everything in FOLLOW (A) is in FOLLOW (B).

Working :

FOLLOW (E) = { $ } ∪ FIRST ()) = { $,) }

FOLLOW (B) = FOLLOW (E) = { $,) }

FOLLOW(T) = FIRST(B) ∪ FOLLOW(B) = {#, +,)}
- {@}

FOLLOW(Y) = FOLLOW(T) = {#, +,)}

FOLLOW(F) = FIRST(T') ∪ FOLLOW(T') = {#, *, +,}
- {@}

Sashrika Surya
RA1911027010092
Section: N2
Lab Batch: 1
Date: 18th February 2022

Compiler Design Lab
Experiment 5: Computation of FIRST() and FOLLOW()

Code:

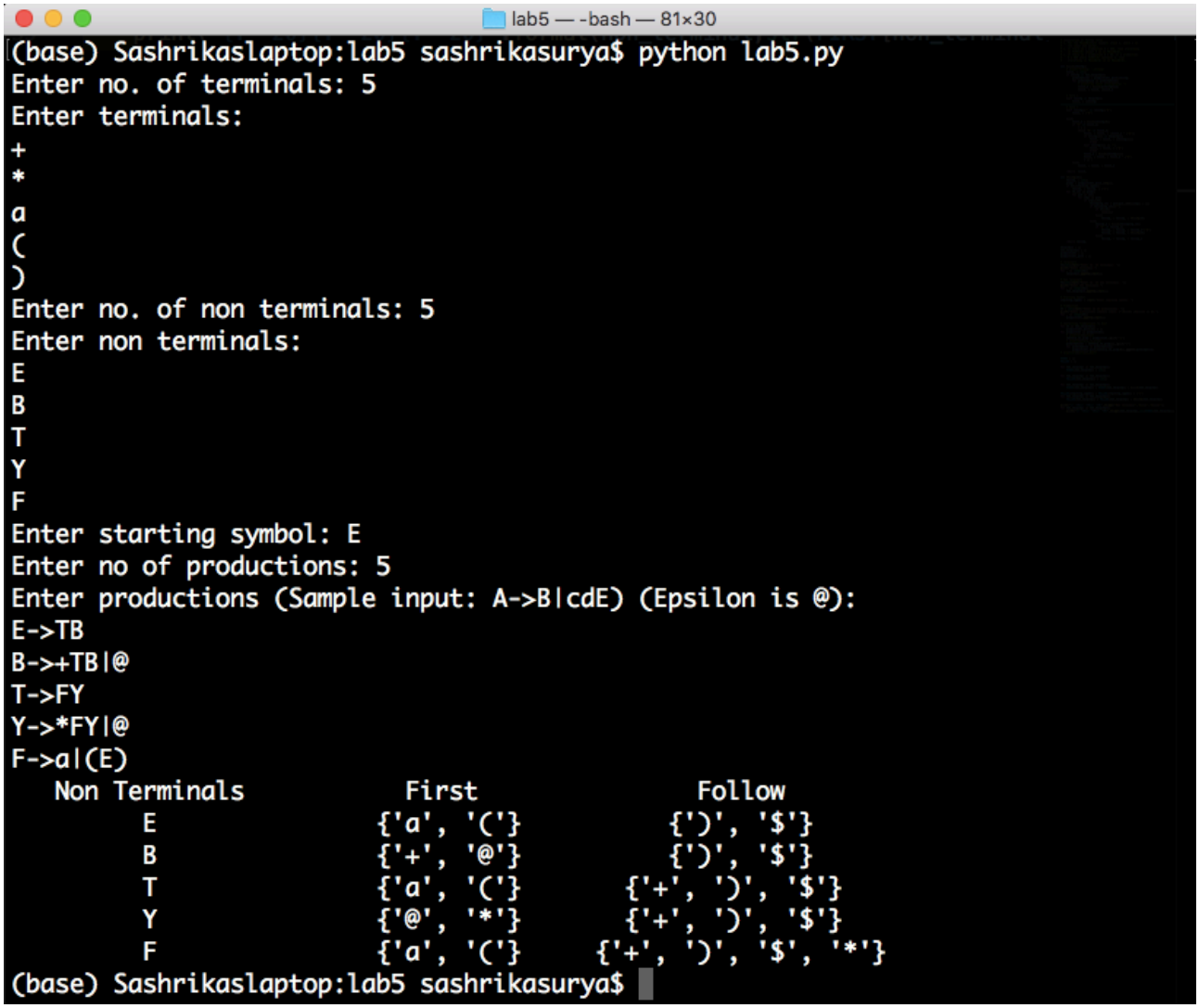
```
# Computation of First rules
# 1. If X is a terminal, then FIRST(X) = {X}.
# 2. If X → ε is a production, then add ε to FIRST(X).
# 3. If X is a non-terminal and X → Y1 Y2 ⋯ Yk is a
#    production, then add FIRST(Y1) to FIRST(X). If Y1
#    derives ε, then add FIRST(Y2) to FIRST(X).

# Computation of Follow rules
# 1. For the FOLLOW(start symbol) place $, where $ is
#    the input end marker.
# 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 → αB, or a production
#    A → αBβ where FIRST(β) contains ε, then
#    everything in FOLLOW(A) is in FOLLOW(B).

def first(string):
    first_ = set()
    if string in non_terminals:
        alternatives = productions_dict[string]
        for alternative in alternatives:
            first_2 = first(alternative)
            first_ = first_ |first_2

    # Rule 1
    elif string in terminals:
        first_ = {string}

    # Rule 2
    elif string=='' or string=='@':
        first_ = {'@'}

    else:
        first_2 = first(string[0])
```

```
        if '@' in first_2:
            i = 1
            while '@' in first_2:
                first_ = first_ | (first_2 - {'@'})
                if string[i:] in terminals:
                    first_ = first_ | {string[i:]}
                    break
                elif string[i:] == '':
                    first_ = first_ | {'@'}
                    break
                first_2 = first(string[i:])
                first_ = first_ | first_2 - {'@'}
                i += 1
        else:
            first_ = first_ | first_2

    return  first_

def follow(nT):
    follow_ = set()
    prods = productions_dict.items()
    # Rule 1
    if nT==starting_symbol:
        follow_ = follow_ | {'$'}
    for nt,rhs in prods:
        for alt in rhs:
            for char in alt:
                if char==nT:
                    following_str = alt[alt.index(char) + 1:]
                    if following_str=='':
                        if nt==nT:
                            continue
                        else:
                            follow_ = follow_ | follow(nt)
                    else:
                        follow_2 = first(following_str)
                        if '@' in follow_2:
                            follow_ = follow_ | follow_2-{'@'}
                            follow_ = follow_ | follow(nt)
                        else:
                            follow_ = follow_ | follow_2
    return follow_

terminals = []
non_terminals = []
```

```
productions = []
productions_dict = {}

# Terminals
n1=int(input("Enter no. of terminals: "))
print("Enter terminals:")
for _ in range(n1):
    terminals.append(input())

# Non Terminals
n2=int(input("Enter no. of non terminals: "))
print("Enter non terminals:")
for _ in range(n2):
    non_terminals.append(input())

# Starting Symbol
starting_symbol = input("Enter starting symbol: ")

# Production
n3 = int(input("Enter no of productions: "))
print("Enter productions (Sample input: A->B|cdE) (Epsilon is @):")
for _ in range(n3):
    productions.append(input())

# Converting Productions to dict
for nT in non_terminals:
    productions_dict[nT] = []
for production in productions:
    nonterm_to_prod = production.split("->")
    alternatives = nonterm_to_prod[1].split("|")
    for alternative in alternatives:
        productions_dict[nonterm_to_prod[0]].append(alternative)

FIRST = {}
FOLLOW = {}

for non_terminal in non_terminals:
    FIRST[non_terminal] = set()

for non_terminal in non_terminals:
    FOLLOW[non_terminal] = set()

for non_terminal in non_terminals:
    FIRST[non_terminal] = FIRST[non_terminal] | first(non_terminal)
```

```
FOLLOW[starting_symbol] = FOLLOW[starting_symbol] | {'$'}
for non_terminal in non_terminals:
    FOLLOW[non_terminal] = FOLLOW[non_terminal] | follow(non_terminal)

print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))
for non_terminal in non_terminals:
    print("{: ^20}{: ^20}{:
^20}".format(non_terminal,str(FIRST[non_terminal]),str(FOLLOW[non_terminal])))
```

Output:



Result:

Hence, FIRST and FOLLOW was successfully computed for the given productions.