University of Asia Pacific
Department of Computer Science & Engineering

Course Title: Compiler Design Lab
Course Code: CSE 430

Lab Exercise-04 Solution

Submitted by:-
Junnatul Mawa
Section:-B1
Registration:- 20101070

Description of Exercise-

Introduction:

The construction of a predictive parser is aided by two functions associated with a grammar G.
These functions, FIRST and FOLLOW, allow us to fill in the entries of a predictive parsing table
for G, whenever possible. First and Follow sets are needed so that the parser can properly apply
the needed production rule at the correct position.
FIRST(α)
First(α) is a set of terminal symbols that begin in strings derived from α.
Rule 1:
If FIRST (α) is the set of terminals that begin the strings derived from α.
Example 1:

Consider the production rule-
A → abc | def ghi

Then, we have-
First (A) = {a, d, g}

Rule 2:
If α → ∈, then ∈, is also in FIRST (α).
Example 2:
For a production rule X → ∈, First(X) = {∈}
Rule 3:
If FIRST (α) is the set of non-terminals:
If α → Y1 Y2 Y3.....Yk is a rule then
If a is in First (Y1) then
Add a to First (α)

If ∈ is in First (Y1) and a is in First (Y2) then

Add a to First (α)

If ∈ is in First (Y1) and ∈ is in First (Y2) and a is in First (Y3)

then Add a to First (α)

.....
If ∈ is in First (Yi) for all Yi then
Add ∈ to First (α)
FOLLOW (A):
FOLLOW (A), for nonterminal A, to be the set of terminals that can appear immediately to the
right of A in some sentential form. That is, the set of terminals such that there exists a derivation
of the form S aAαβ for some α and β.
To compute FOLLOW (A) for all nonterminal A, apply the following rules until nothing can be
added to any FOLLOW set:
Rule 1:
Place $ in FOLLOW(S), where S is the start symbol and $ is the input right end marker.
Rule 2:
If there is a production A αBβ, then everything in FIRST (β), except for ∈, is placed in
FOLLOW (B).
Rule 3:
If there is a production A →αB, or a production A → aBβ where FIRST (β) contains e (i.e., β□∈),
Then everything in FOLLOW (A) is in FOLLOW (B).
Example 1:
S → aSe | B FIRST(S) = {a, b, c, d, ε}
B → bBCf | C FIRST (B) = {b, c, d, ε}
C → cCg | d | ε FIRST(C) = {c, d, ε}

According to Rule 1:
FOLLOW (S) = {$}
According to Rule 2:
FOLLOW(C) = {f, g}
FOLLOW (B) = {c, d, f}
FOLLOW(S) = {$, e}
According to Rule 3:

FOLLOW (C) = {f, g} ∪ FOLLOW (B) = {c, d, e, f, g, $}
FOLLOW (B) ={c, d, f} ∪ FOLLOW(S) = {c, d, e, f, $}
FOLLOW (S) = {$, e}

Here is the code-

first_follow-20101070.py > ...

```python
import sys
sys.setrecursionlimit(60)

def first(string):
    #print("first({})".format(string))
    first_ = set()
    if string in non_terminals:
        alternatives = productions_dict[string]

        for alternative in alternatives:
            first_2 = first(alternative)
            first_ = first_ |first_2

    elif string in terminals:
        first_ = {string}

    elif string=='' or string=='@':
        first_ = {'@'}

    else:
        first_2 = first(string[0])
        if '@' in first_2:
            i = 1
            while '@' in first_2:
                #print("inside while")

                first_ = first_ | (first_2 - {'@'})
                #print('string[i:]=', string[i:])
                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 - {'@'}
```

first_follow-20101070.py > ...

```
4   def first(string):
36                  first_ = first_ | first_2 - {'@'}
37                  i += 1
38          else:
39              first_ = first_ | first_2
40
41
42      #print("returning for first({})".format(string),first_)
43      return  first_
44
45
46  def follow(nT):
47      #print("inside follow({})".format(nT))
48      follow_ = set()
49      #print("FOLLOW", FOLLOW)
50      prods = productions_dict.items()
51      if nT==starting_symbol:
52          follow_ = follow_ | {'$'}
53      for nt,rhs in prods:
54          #print("nt to rhs", nt,rhs)
55          for alt in rhs:
56              for char in alt:
57                  if char==nT:
58                      following_str = alt[alt.index(char) + 1:]
59                      if following_str=='':
60                          if nt==nT:
61                              continue
62                          else:
63                              follow_ = follow_ | follow(nt)
64                      else:
65                          follow_2 = first(following_str)
66                          if '@' in follow_2:
67                              follow_ = follow_ | follow_2-{'@'}
68                              follow_ = follow_ | follow(nt)
```

```
46  def follow(nT):
68                              follow_ = follow_ | follow(nt)
69                          else:
70                              follow_ = follow_ | follow_2
71      #print("returning for follow({})".format(nT),follow_)
72      return follow_
73
74
75  no_of_terminals=int(input("Enter no. of terminals: "))
76
77  terminals = []
78
79  print("Enter the terminals :")
80  for _ in range(no_of_terminals):
81      terminals.append(input())
82
83  no_of_non_terminals=int(input("Enter no. of non terminals: "))
84
85  non_terminals = []
86
87  print("Enter the non terminals :")
88  for _ in range(no_of_non_terminals):
89      non_terminals.append(input())
90
91  starting_symbol = input("Enter the starting symbol: ")
92
93  no_of_productions = int(input("Enter no of productions: "))
94
95  productions = []
96
97  print("Enter the productions:")
98  for _ in range(no_of_productions):
99      productions.append(input())
100
101
102 #print("terminals", terminals)
```

```
#print("terminals", terminals)

#print("non terminals", non_terminals)

#print("productions",productions)


productions_dict = {}

for nT in non_terminals:
    productions_dict[nT] = []


#print("productions_dict",productions_dict)

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)

#print("productions_dict",productions_dict)

#print("nonterm_to_prod",nonterm_to_prod)
#print("alternatives",alternatives)


FIRST = {}
FOLLOW = {}

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

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

```
122
123 #print("productions_dict",productions_dict)
124
125 #print("nonterm_to_prod",nonterm_to_prod)
126 #print("alternatives",alternatives)
127
128
129 FIRST = {}
130 FOLLOW = {}
131
132 for non_terminal in non_terminals:
133     FIRST[non_terminal] = set()
134
135 for non_terminal in non_terminals:
136     FOLLOW[non_terminal] = set()
137
138 #print("FIRST",FIRST)
139
140 for non_terminal in non_terminals:
141     FIRST[non_terminal] = FIRST[non_terminal] | first(non_terminal)
142
143 #print("FIRST",FIRST)
144
145
146 FOLLOW[starting_symbol] = FOLLOW[starting_symbol] | {'$'}
147 for non_terminal in non_terminals:
148     FOLLOW[non_terminal] = FOLLOW[non_terminal] | follow(non_terminal)
149
150 #print("FOLLOW", FOLLOW)
151
152 print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))
153 for non_terminal in non_terminals:
154     print("{: ^20}{: ^20}{: ^20}".format(non_terminal,str(FIRST[non_terminal]),str(FOLLOW[non_terminal])))
```