Page No.
Class
Roll No.
Date: / /202

Practical No. 05

Title : Simulate first and follow of a Grammer

Theory :

Need for first :

If the compiler would have come to know in advance that what is the "First character of the string produced when a production rule is applied" and comparing it to the current character or token in the input string it sees it can wisely take decision on which production rule to apply

let's take the following grammer :

S → cAd

A → b/ε

And the input string is "cad"

thus in the ex above if it knew that after reading character 'c' in the input string and applying S → cAd.

Hence it is validated that if the compiler/parser knows about first character of the string that can be obtained by applying a production rule then it can wisely apply the correct production rule to get the syntax tree for the given input string

FIRST (X) for a grammer symbol X is the set of terminals that being the strings derivable from X.

Page No.
Class
Roll No.
Date: /

Rule to complete FIRST set :-

1. If X is a terminal then FIRST (X) = {, X"}.
2. If X → ∈ is a production rule then add ∈ to FIRST
3. If X → Y1, Y2, Y3 ..... Yn is a production.
   1. FIRST (X) = FIRST (Y1)
   2. If FIRST (Y1) contains ∈ then FIRST (X) = {FIRST
      ∪ {FIRST (Y2)}
   3. If FIRST (Yi) contains ∈ for all i=1 to n, then add
      to FIRST (X).

Example 1:-

production Rule of Grammer

E → TE'

E' → +TE' / ∈T → FT'

T' → *FT' / ∈

F → (E) / id

FIRST sets

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

FIRST (E') = {+, ∈}

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

FIRST (T') = {*, ∈}

FIRST (F) = {(, id}

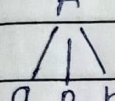
Page No.
Class
Roll No.
Date: / /202

Need for Follow:-

- The parser faces one more problem

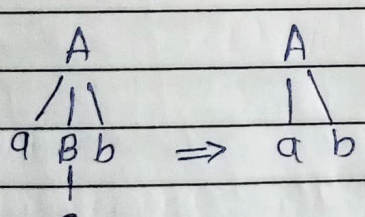
A → aBb

B → c/ϵ



Now the parser checks for the second character of the input string which is b, and the non-terminals to drive B, but the parser can't get any string derivable from B that contains b as first character.

In RHS of A → aBb, b follows Non-terminals B i.e FOLLOW(B) = {b} and the current input character read is also b hence the parser applied this rule and it is able to get the string "ab" from the given grammer



So follow can make a non-terminal to vanish out if needed to generate the string from the parse tree.

Rules to complete FOLLOW set:-

1) FOLLOW(S) = {$} // where S is the starting Non-terminal
2) If A → pBq is a production where p, B and q are any grammer symbol then everything in FIRST(q) except ϵ is in FOLLOW(B)

3. If A → pB is a production, then everything in FOLLO is in FOLLOW(B).
4. If A → pBq is a production and FIRST(q) contains ϵ FOLLOW(B) contains {FIRST(q) − ϵ} U FOLLOW(A).

Ex.

Production Rule

E → TE'
E' → +TE' / ϵT → FT'
T' → *FT' / ϵ
F → (E) / id

First set

first(E) = first(T) = {(, id)}, first(E') = {+, ϵ}.
first(T) = first(F) = {(, id}
first(F) = {(, id}

Follow set

follow = {$,)}
follow(E) = follow(E') = {$,)}
follow(T) = {first(E') − ϵ} U follow(E') = {+, $,)}

Result: thus the program for implementing the first a follow of a gramma was executed and the output was verified successfully.