Government College of Engineering, Jalgaon

(An Autonomous Institute of Govt. of Maharashtra)

Name of Student:……
PRN:……. …
Course Teacher: Sharayu Bonde
Date of Performance: **Date of completion:**

Experiment No.: Group A: 05

Title: Simulate First and Follow of a Grammar

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 grammar:

S -> cAd

A -> bc|a

And the input string is "cad".

Thus, in the example above, if it knew that after reading character „c" in the input string and applying S->cAd, next character in the input string is „a", then it would have ignored the production rule A->bc (because „b" is the first character of the string produced by this production rule, not „a"), and directly use the production rule A->a (because „a" is the first character of the string produced by this production rule, and is same as the current character of the input string which is also „a").

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 correct syntax tree for the given input string.

FIRST(X) for a grammar symbol X is the set of terminals that begin the strings derivable from X.
Rules to compute FIRST set:

1. If x is a terminal, then FIRST(x) = { „x" }
2. If x-> Є, is a production rule, then add Є to FIRST(x).
3. If X->Y1 Y2 Y3….Yn is a production,
    1. FIRST(X) = FIRST(Y1)
    2. If FIRST(Y1) contains Є then FIRST(X) = { FIRST(Y1) – Є } U { FIRST(Y2) }
    3. If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X).

Example 1:

Production Rules of Grammar
E -> TE‟
E‟ -> +T E‟|Є
-> F T‟
T‟ -> *F T‟ | Є
F -> (E) | id
FIRST sets
FIRST(E) = FIRST(T) = { (, id }
FIRST(E‟) = { +, Є }
FIRST(T) = FIRST(F) = { (, id }
FIRST(T‟) = { *, Є }
FIRST(F) = { (, id }

Need for Follow :

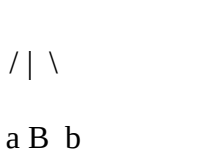
The parser faces one more problem. Let us consider below grammar to understand this problem.

A -> aBb

B -> c | ε

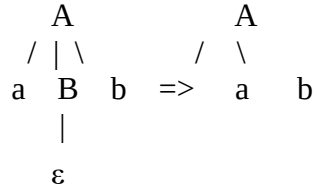
And suppose the input string is “ab” to parse.

As the first character in the input is a, the parser applies the rule A->aBb.



Now the parser checks for the second character of the input string which is b, and the Non-Terminal to derive is B, but the parser can‟t get any string derivable from B that contains b as first character. But the Grammar does contain a production rule B -> ε, if that is applied then B will vanish, and the parser gets the input “ab” , as shown below. But the parser can apply it only when it knows that the character that follows B is same as the current character in the input.

In RHS of A -> aBb, b follows Non-Terminal B, i.e. FOLLOW(B) = {b}, and the current input character read is also b. Hence the parser applies this rule. And it is able to get the string “ab” from the given grammar.



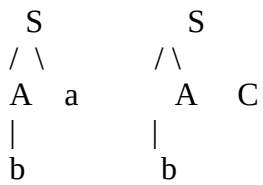
So FOLLOW can make a Non-terminal to vanish out if needed to generate the string from the parse tree.

Follow(X) to be the set of terminals that can appear immediately to the right of Non-Terminal X in some sentential form.

Example:

S ->Aa | Ac

A ->b



Here, FOLLOW (A) = {a, c}

Rules to compute 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 grammar symbols, then everything in FIRST(q) except Є is in FOLLOW(B.
3) If A->pB is a production, then everything in FOLLOW(A) is in FOLLOW(B).
4) If A->pBq is a production and FIRST(q) contains Є, then FOLLOW(B) contains { FIRST(q) – Є } U FOLLOW(A)

Example 1:

Production Rules:

E -> TE‟

E‟ -> +T E‟|Є T

-> F T‟

T‟ -> *F T‟ | Є

F -> (E) | id

FIRST set

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

FIRST(E‟) = { +, Є }

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

FIRST(T‟) = { *, Є }

FIRST(F) = { (, id }

FOLLOW Set

FOLLOW(E) = { $,) } // Note ')' is there because of 5th rule

FOLLOW(E‟) = FOLLOW(E) = { $,) } // See 1st production rule

FOLLOW(T) = { FIRST(E‟) – Є } U FOLLOW(E‟) = { + , $,) }

FOLLOW(T‟) = FOLLOW(T) = { + , $,) }

FOLLOW(F) = { FIRST(T‟) – Є } U FOLLOW(T‟) = { *, +, $,) }

Result:

Thus the program for implementing the first and follow of a grammar was executed and the output was verified successfully.

Course Teacher

Sharayu N. Bonde