Calculating First and Follow sets of given Grammar

24 Samihan Inamdar
27 Pranit Jadhao
43 Mayur Jadhav
57 Adesh Ramgude

Contents

Syntax Analysis

Syntax Analysis or Parsing is the second phase, i.e. after lexical analysis. It checks the syntactic structure of the given input, i.e. whether the given input is in the correct syntax (of the language in which the input has been written) or not.

It does so by building a data structure, called a Parse tree or Syntax tree. The parse tree is constructed by using the predefined Grammar of the language and the input string. If the given input string can be produced with the help of the syntax tree (in the derivation process), the input string is found to be in the correct syntax. if not, error is reported by syntax analyzer.

Why 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.

Example:

S -> cAd
A -> bc|a
And the input string is “cad”.

Why 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” . But the parser can apply it only when it knows that the character that follows B in the production rule is same as the current character in the input.

FIRST set in syntax analysis

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 of FIRST set

Production Rules of Grammar
S -> ACB | Cbb | Ba
A -> da | BC
B -> g | Є
C -> h | Є

FIRST sets
FIRST(S) = FIRST(A) U FIRST(B) U FIRST(C)
= { d, g, h, Є, b, a}
FIRST(A) = { d } U FIRST(B) = { d, g , h, Є }
FIRST(B) = { g , Є }
FIRST(C) = { h , Є }

FOLLOW set in syntax analysis

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

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 of FOLLOW set

Production Rules:

S -> ACB|Cbb|Ba

A -> da|BC

B-> g|Є

C-> h| Є

FIRST set

FIRST(S) = FIRST(A) U FIRST(B) U FIRST(C) = { d, g, h, Є, b, a}

FIRST(A) = { d } U FIRST(B) = { d, g, Є }

FIRST(B) = { g, Є }

FIRST(C) = { h, Є }

FOLLOW Set

FOLLOW(S) = { $ }

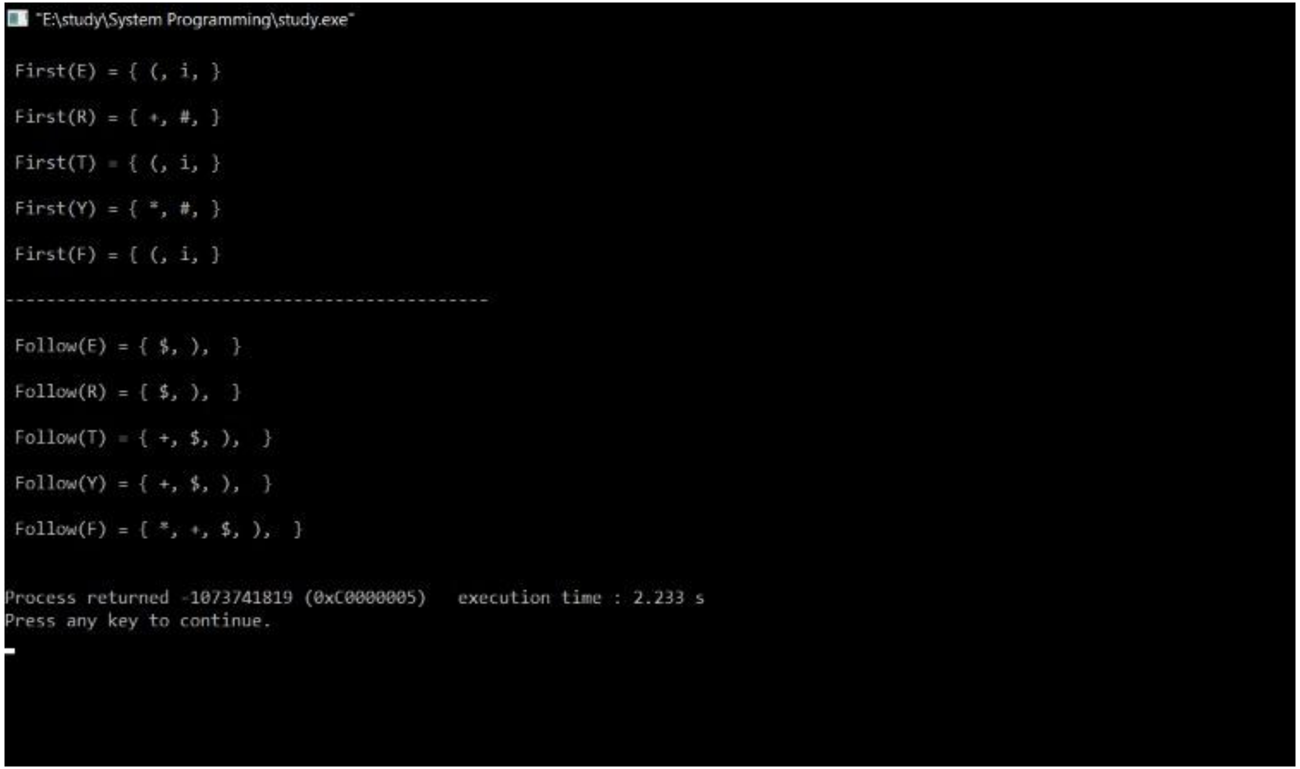
FOLLOW(A) = { h, g, $ }

FOLLOW(B) = { a, $, h, g }

FOLLOW(C) = { b, g, $, h }

Result

Input :
E -> TR
R -> +T R| #
T -> F Y
Y -> *F Y | #
F -> (E) | i



```
"E:\study\System Programming\study.exe"

 First(E) = { (, i, }

 First(R) = { +, #, }

 First(T) = { (, i, }

 First(Y) = { *, #, }

 First(F) = { (, i, }

-----------------------------------------------

 Follow(E) = { $, ),  }

 Follow(R) = { $, ),  }

 Follow(T) = { +, $, ),  }

 Follow(Y) = { +, $, ),  }

 Follow(F) = { *, +, $, ),  }


Process returned -1073741819 (0xC0000005)   execution time : 2.233 s
Press any key to continue.
```

Points to be remembered

1. The grammar used is Context-Free Grammar (CFG).
2. Є as a FOLLOW doesn’t mean anything (Є is an empty string).

Advantages

1. FOLLOW can be applied to a single non-terminal only, and returns a set of terminals.
2. FIRST and FOLLOW help us to pick a rule when we have a choice between two or more r.h.s. by predicting the first symbol that each r.h.s. can derive.
3. Even if there are only one r.h.s. we can still use them to tell us whether or not we have an error - if the current input symbol cannot be derived from the only r.h.s. available, then we know immediately that the sentence does not belong to the grammar, without having to (attempt to) finish the parse.
4. Backtracking is not needed to get the correct syntax tree.

Conclusion

The conclusions is, we need to find FIRST and FOLLOW sets for a given grammar, so that the parser can properly apply the needed rule at the correct position.

Thank you