Session 5: Use of CFGs for Parsing

Assignment:

- We can think of using CFGs to detect various language constructs in the token streams freed from simple syntactic and semantic errors, as it is easier to describe the constructs with CFGs. But CFGs are hard to apply practically. In this session we first exercise on simpler implementations of CFGs, and then implement the FIRST and FOLLOW functions that facilitate efficient parsing algorithms for practical problems.

  - Observe the C code segment that implements the nonterminals of the CFG given beside it.

```
void S() {
        if (str[i] == 'b'){
           i++;
           f=1;
           return;
   }
   else {
          A();
          if (f) { B(); return;}
   }
}

void A() {
        if (str[i] == 'a') {
                i++;
                f=1;
        }
        else {f=0; return;}
   if (i<l-1) A();
}
```

```
void B() {
        if (str[i] == 'b') {
                i++;
                f=1;
                return;
        }
        else {f=0; return;}
}
```

CFG for Exercise 1:

$S \rightarrow b \mid AB$

$A \rightarrow a \mid aA$

$B \rightarrow b$

Language generated:

{b, ab, aab, aaab, …}

1. Implement the grammar given above.

2. Implement the following CFG, in the way shown above or with the help of a user defined stack, that is, in the way it may work in an implementation of the transition function of a PDA.

$A \rightarrow aXd$

$X \rightarrow bbX$

$X \rightarrow bcX$

$X \rightarrow \varepsilon$

3. Implement the FIRST and FOLLOW functions described below.

- To Compute FIRST(X) for all grammar symbols X, apply the following rules until no more terminals or ε can be added to any FIRST set.
  1. If X is a terminal, then FIRST(X) is {X}.
  2. If X is a non-terminal and $X \rightarrow Y_1 Y_2 \ldots Y_k$ is a production for some $k \geq 1$, then place b in FIRST(X) if for some i, b is in FIRST(Y_i), and ε is in all of FIRST(Y_1), ..., FIRST(Y_{i-1}); that is, $Y_1, \ldots, Y_{i-1}$ derives ε.
  3. If ε is in FIRST(Y_j) for all j = 1, 2, ..., k then add ε to FIRST(X).

  Sample input and corresponding output:

$E \rightarrow TE'$ $E' \rightarrow +TE' \mid \varepsilon$ $T \rightarrow FT'$ $T' \rightarrow *FT' \mid \varepsilon$ $F \rightarrow (E) \mid id$	FIRST (E) = {(, id} FIRST (T) = {(, id} FIRST (E') = {+, ε} FIRST (T') = {*, ε} FIRST (F) = {(, id}

- To compute FOLLOW(A) for all non-terminals A, apply the following rules until nothing can be added to any FOLLOW set.
  1. Place $ in FOLLOW(S), where S is the start symbol and $ is the right end-marker of an input.
  2. If there is a production $A \rightarrow \alpha B \beta$, then everything in FIRST(β) except ε is in FOLLOW(B).
  3. If there is a production $A \rightarrow \alpha B$, then everything in FOLLOW(A) is in FOLLOW(B).
  4. If there is a production $A \rightarrow \alpha B \beta$ where FIRST(β) contains ε, then everything in FOLLOW(A) is in FOLLOW(B).

  Sample input and corresponding output:

$E \rightarrow TE'$ $E' \rightarrow +TE' \mid \varepsilon$ $T \rightarrow FT'$ $T' \rightarrow *FT' \mid \varepsilon$ $F \rightarrow (E) \mid id$	FOLLOW (E) = {$,)} FOLLOW (T) = {+, $,)} FOLLOW (E') = {), $} FOLLOW (T') = {+,),$} FOLLOW (F) = {*, +, $,)}