FIRST and FOLLOW sets – a necessary preliminary to constructing the LL(1) parsing table

Remember: A predictive parser can only be built for an LL(1) grammar. A grammar is not LL(1) if it is:

1. Left recursive, or
2. Not left factored.

However, grammars that are not left recursive and are left factored may still not be LL(1). To see if a grammar is LL(1), try to build the parse table for the predictive parser by the method we are about to describe. If any element in the table contains more than one grammar rule right-hand side, then the grammar is not LL(1).

To build the table, we must must compute FIRST and FOLLOW sets for the grammar.

FIRST Sets

Ultimately, we want to define FIRST sets for the right-hand sides of each of the grammar's productions. The idea is that for a sequence α of symbols, FIRST(α) is the set of terminals that begin the strings derivable from α, and also, if α can derive ϵ, then ϵ is in FIRST(α). By allowing ϵ to be in FIRST(α), we can avoid defining the notion of "nullable" as Appel does in the book. More formally:

$$FIRST(\alpha) = \{t \mid (t\ is\ a\ terminal\ and\ \alpha \Rightarrow* t\beta)\ or\ (t = \epsilon\ and\ \alpha \Rightarrow* \epsilon)\}$$

To define $FIRST(\alpha)$ for arbitrary α, we start by defining $FIRST(X)$, for a single symbol X (a terminal, a nonterminal, or ϵ):

- X is a terminal: $FIRST(X) = X$
- X is ϵ: $FIRST(X) = \epsilon$

- X is a nonterminal. In this case, we must look at all grammar productions with X on the left, i.e., productions of the form:

$$X \rightarrow Y_1 Y_2 Y_3 \ldots Y_k$$

  where each Y_k is a single terminal or nonterminal (or there is just one Y, and it is ϵ). For each such production, we perform the following actions:
  - Put $FIRST(Y_1) - \{\epsilon\}$ into $FIRST(X)$.
  - If ϵ is in $FIRST(Y_1)$, then put $FIRST(Y_2) - \{\epsilon\}$ into $FIRST(X)$.
  - If ϵ is in $FIRST(Y_2)$, then put $FIRST(Y_3) - \{\epsilon\}$ into $FIRST(X)$.
  - . . .
  - If ϵ is in $FIRST(Y_i)$ for $1 \preceq i \preceq k$ (all production right- hand sides) then put ϵ into $FIRST(X)$.

For example, compute the FIRST sets of each of the non-terminals in the following grammar:

$$exp \rightarrow \ term \ exp'$$

$$exp' \rightarrow - \ term \ exp' \mid \epsilon$$
$$term \rightarrow factor \ term'$$
$$term' \rightarrow / \ factor \ term' \mid \epsilon$$
$$factor \rightarrow INTLITERAL \mid (\ exp \ )$$

Once we have computed FIRST(X) for each terminal and nonterminal X, we can compute FIRST(α) for every production's right-hand-side α. In general, alpha will be of the form:

$$X_1 X_2 \ldots X_n$$

where each X is a single terminal or nonterminal, or there is just one X_1 and it is ϵ. The rules for computing FIRST(α) are essentially the same as the rules for computing the first set of a nonterminal.

- Put $FIRST(X_1) - \{\epsilon\}$ into $FIRST(\alpha)$
- If ϵ is in $FIRST(X_1)$ put $FIRST(X_2) - \{\epsilon\}$ into $FIRST(\alpha)$.

- . . .
- If ϵ is in the FIRST set for every X_k, put ϵ into $FIRST(\alpha)$.

For the example grammar above, compute the FIRST sets for each production's right-hand side.

Why do we care about the FIRST(α) sets? During parsing, suppose the top-of- stack symbol is nonterminal A, that there are two productions $A \longrightarrow \alpha$ and $A \longrightarrow \beta$, and that the current token is a. Well, if FIRST(α) includes a . . .

FOLLOW sets

FOLLOW sets are only defined for single nonterminals. The definition is as follows:

For a nonterminal A, FOLLOW(A) is the set of terminals that can appear immediately to the right of A in some partial derivation; i.e., terminal t is in FOLLOW(A) if:

$$S \Rightarrow+ \ldots\ A\ t\ \ldots$$

where t is a terminal

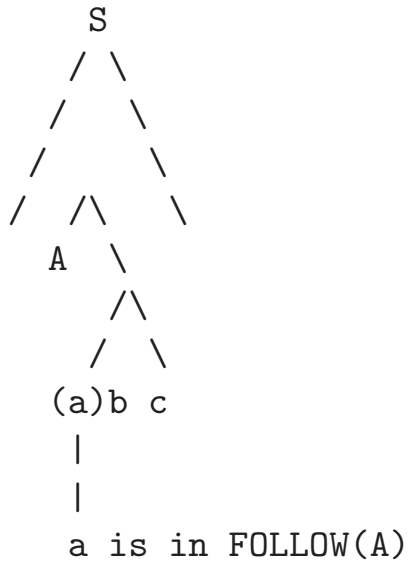
Furthermore, if A can be the rightmost symbol in a derivation, then EOF is in FOLLOW(A).

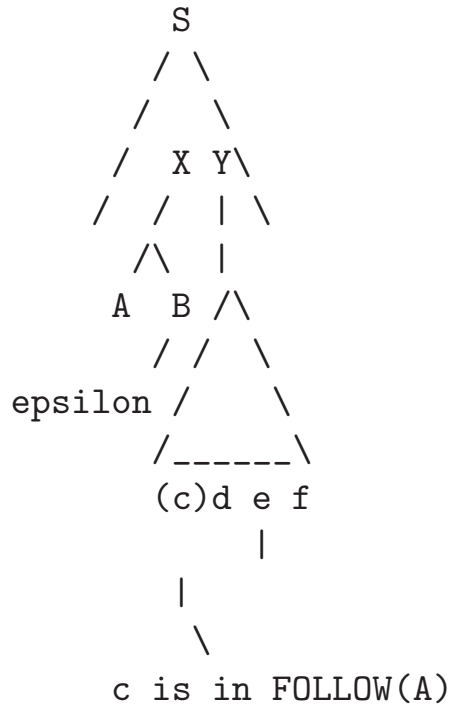
It is worth noting that epsilon is never in a FOLLOW set.

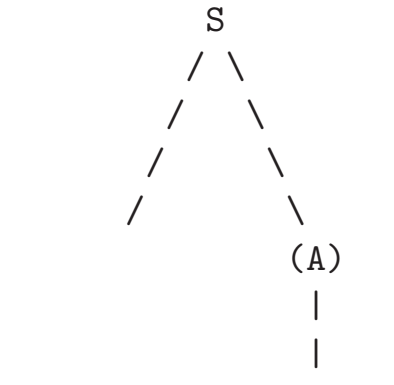
Notationally:

$$FOLLOW(A) = \{t \mid (t\ is\ a\ terminal\ and\ S \Rightarrow+\ \alpha\ A\ t\ \beta)\ or\ (t\ is\ EOF\ and\ S \Rightarrow*\ \alpha A)\}$$

Some conditions under which symbols a, c, and EOF are in the FOLLOW set of nonterminal A:







How to compute FOLLOW(A) for each nonterminal A:

- If A is the start nonterminal, put EOF in FOLLOW(A)

Find the productions with A on the right-hand-side:

- For each production $X \rightarrow \alpha A \beta$, put $FIRST(\beta) - \{\epsilon\}$ in FOLLOW(A)
- If ϵ is in $FIRST(\beta)$ then put $FOLLOW(X)$ into $FOLLOW(A)$
- For each production $X \rightarrow \alpha A$, put FOLLOW(X) into FOLLOW(A)

Note that

- To compute FIRST(A) you must look for A on a production's left-hand side.
- To compute FOLLOW(A) you must look for A on a production's right-hand side.
- FIRST and FOLLOW sets are always sets of terminals (plus, perhaps, ϵ for FIRST sets, and EOF for follow sets). Nonterminals are never in a FIRST or a FOLLOW set.

Example: Compute FOLLOW sets of the non-terminals in the following language:

$$S \rightarrow B\ c \mid D\ B$$

$$B \rightarrow a\ b \mid c\ S$$

$$D \rightarrow d \mid \epsilon$$

Why do we care about FOLLOW sets? Suppose during parsing we have X on top of the stack and a is the current token. What if X has two productions $X \rightarrow \alpha$ and $X \rightarrow \beta$ but a is not in the FIRST set of either α or β?

How to build parse tables – and tell if a grammar is LL(1)

To build the table, we fill in the rows one at a time for each nonterminal X as follows:

- for each production $X \rightarrow \alpha$
  - for each terminal t in First(α): put α in Table[X,t]
  - if ϵ is in First(α) then:
    - for each terminal t in Follow(X): put α in Table[X,t]

The grammar is not LL(1) if and only if there is more than one entry for any cell in the table. Try building a parse table for the following grammar:

$$S \rightarrow B\ c \mid D\ B$$

$$B \rightarrow a\ b \mid c\ S$$

$$D \rightarrow d \mid \epsilon$$