Parsing

**Parsing:** Deciding if \( x \in \Sigma^* \) is in \( L(G) \) for some CFG \( G \).

**Review**

Consider the CFG \( G \):

\[
S \rightarrow Aa \\
A \rightarrow AA | ABa | \lambda \\
B \rightarrow BBa | b | \lambda
\]

Is \( ba \) in \( L(G) \)? Running time?

Remove \( \lambda \)-rules, then unit productions, and then useless productions from the grammar \( G \) above. New grammar \( G' \) is:

\[
S \rightarrow Aa | a \\
A \rightarrow AA | ABa | Aa | Ba | a \\
B \rightarrow BBa | Ba | a | b
\]

Is \( ba \) in \( L(G) \)? Running time?

**Top-down Parser:**

- Start with \( S \) and try to derive the string.

\[
S \rightarrow aS \mid b
\]

- Examples: LL Parser, Recursive Descent
Bottom-up Parser:

- Start with string, work backwards, and try to derive S.

- Examples: Shift-reduce, Operator-Precedence, LR Parser

We will use the following functions FIRST and FOLLOW to aid in computing parse tables.

The function FIRST:

Some notation that we will use in defining FIRST and FOLLOW.

\[ G=(V,T,S,P) \]

\[ w,v \in (V \cup T)^* \]

\[ a \in T \]

\[ X,A,B \in V \]

\[ X_I \in (V \cup T)^+ \]

**Definition**: FIRST(w) = the set of terminals that begin strings derived from w.

- If \( w \xrightarrow{*} av \) then a is in FIRST(w)
- If \( w \xrightarrow{*} \lambda \) then \( \lambda \) is in FIRST(w)

To compute FIRST:

1. FIRST(a) = \{a\}
2. FIRST(X)
   - (a) If \( X \to aw \) then a is in FIRST(X)
   - (b) If \( X \to \lambda \) then \( \lambda \) is in FIRST(X)
   - (c) If \( X \to Aw \) and \( \lambda \in \text{FIRST}(A) \) then Everything in FIRST(w) is in FIRST(X)
3. In general, FIRST(X_1X_2X_3..X_K) =
   - FIRST(X_1)
   - \( \cup \) FIRST(X_2) if \( \lambda \) is in FIRST(X_1)
   - \( \cup \) FIRST(X_3) if \( \lambda \) is in FIRST(X_1) and \( \lambda \) is in FIRST(X_2)
   - ...
   - \( \cup \) FIRST(X_K) if \( \lambda \) is in FIRST(X_1) and \( \lambda \) is in FIRST(X_2) and \( \lambda \) is in FIRST(X_{K-1})
   - \{-\} if \( \lambda \notin \text{FIRST}(X_J) \) for all J
**Example:** $L = \{a^n b^m c^n : n \geq 0, 0 \leq m \leq 1\}$

\[
S \rightarrow aSc | B \\
B \rightarrow b | \lambda
\]

**FIRST(B) =**

**FIRST(S) =**

**FIRST(Sc) =**

**Example**

\[
S \rightarrow BCD | aD \\
A \rightarrow CEB | aA \\
B \rightarrow b | \lambda \\
C \rightarrow dB | \lambda \\
D \rightarrow cA | \lambda \\
E \rightarrow e | fE
\]

**FIRST(S) =**

**FIRST(A) =**

**FIRST(B) =**

**FIRST(C) =**

**FIRST(D) =**

**FIRST(E) =**

**Definition:** \(\text{FOLLOW}(X) = \text{set of terminals that can appear to the right of X in some derivation.}\)

If \(S \Rightarrow wAav\) then
\[
a \text{ is in } \text{FOLLOW}(A)
\]

(where \(w\) and \(v\) are strings of terminals and variables, \(a\) is a terminal, and \(A\) is a variable)
To compute FOLLOW:

1. $ is in FOLLOW(S)
2. If $A → wBv and $v ≠ λ$ then
   \[ \text{FIRST}(v) - \{λ\} \] is in FOLLOW(B)
3. IF $A → wB OR$
   \[A → wBv \text{ and } λ \text{ is in FIRST}(v) \text{ then} \]
   FOLLOW(A) is in FOLLOW(B)
4. $λ$ is never in FOLLOW

Example:

\[
S → aSc \mid B \\
B → b \mid λ
\]

FOLLOW(S) =
FOLLOW(B) =

Example:

\[
S → BCD \mid aD \\
A → CEB \mid aA \\
B → b \mid λ \\
C → dB \mid λ \\
D → cA \mid λ \\
E → e \mid fE
\]

FOLLOW(S) =
FOLLOW(A) =
FOLLOW(B) =
FOLLOW(C) =
FOLLOW(D) =
FOLLOW(E) =