Section: Parsing

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

Consider the CFG $G$:

\[
\begin{align*}
S & \rightarrow Aa \\
A & \rightarrow AA | ABa | \lambda \\
B & \rightarrow BBa | b | \lambda
\end{align*}
\]

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

New grammar $G'$ is:

\[
\begin{align*}
S & \rightarrow Aa | a \\
A & \rightarrow AA | ABa | Aa | Ba | a \\
B & \rightarrow BBa | Ba | a | b
\end{align*}
\]

Is $ba$ in $L(G)$? Running time?
Top-down Parser:

- Start with S and try to derive the string. $aabb$

\[ S \rightarrow aS \mid b \]

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

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

![Diagram](image)

- Examples: Shift-reduce, Operator-Precedence, LR Parser
The function FIRST:

\[ G = (V, T, S, P) \]
\[ w, v \in (V \cup T)^* \]
\[ a \in T \]
\[ X, A, B \in V \]
\[ X_I \in (V \cup T)^+ \]

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

If \( w \Rightarrow^* av \) then 
\[ a \text{ is in } \text{FIRST}(w) \]

If \( w \Rightarrow^* \lambda \) then 
\[ \lambda \text{ is in } \text{FIRST}(w) \]
To compute FIRST:

1. FIRST(a) = \{a\}

2. FIRST(X)

   (a) If \( X \rightarrow aw \) then
       a is in FIRST(X)
   (b) IF \( X \rightarrow \lambda \) then
       \( \lambda \) is in FIRST(X)
   (c) If \( X \rightarrow Aw \) and \( \lambda \in \text{FIRST}(A) \)
       then
       Everything in FIRST(w) is in FIRST(X)
3. In general, \( \text{FIRST}(X_1 X_2 X_3 \ldots X_K) = \)

- \( \text{FIRST}(X_1) \)
- \( \cup \text{FIRST}(X_2) \) if \( \lambda \) is in \( \text{FIRST}(X_1) \)
- \( \cup \text{FIRST}(X_3) \) if \( \lambda \) is in \( \text{FIRST}(X_1) \)
  and \( \lambda \) is in \( \text{FIRST}(X_2) \)
  ...
- \( \cup \text{FIRST}(X_K) \) if \( \lambda \) is in \( \text{FIRST}(X_1) \)
  and \( \lambda \) is in \( \text{FIRST}(X_2) \)
  ...
  and \( \lambda \) is in \( \text{FIRST}(X_{K-1}) \)
- \( \{-\lambda\} \) if \( \lambda \notin \text{FIRST}(X_J) \) for all \( J \)
Example:

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

\[ \text{FIRST}(B) = \{b, \lambda\} \]
\[ \text{FIRST}(S) = \{a, b, c, \lambda\} \]
\[ \text{FIRST}(Sc) = \{a, b, c, \lambda\} \]

\[ \text{FIRST}(Sc) \neq \lambda \]
Example

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

\[
\begin{align*}
\text{FIRST}(S) &= \{ b, d, c, \lambda, a \} \\
\text{FIRST}(A) &= \{ a, d, e, f \} \\
\text{FIRST}(B) &= \{ b, \lambda \} \\
\text{FIRST}(C) &= \{ d, \lambda \} \\
\text{FIRST}(D) &= \{ c, \lambda \} \\
\text{FIRST}(E) &= \{ e, f \}
\end{align*}
\]
Definition: FOLLOW(X) = set of terminals that can appear to the right of X in some derivation.

If S \Rightarrow^* wAav then
a is in FOLLOW(A)

To compute FOLLOW:

1. $ is in FOLLOW(S)
2. If A \rightarrow wBv and v \neq \lambda then
   FIRST(v) - \{\lambda\} is in FOLLOW(B)
3. IF A \rightarrow wB OR
   A \rightarrow wBv and \lambda is in FIRST(v)
   then
   FOLLOW(A) is in FOLLOW(B)
4. \lambda is never in FOLLOW
Example:

\[ S \to aSc \mid B \]
\[ B \to b \mid \lambda \]

\text{FOLLOW}(S) = \{ \epsilon, \$, \% \}
\text{FOLLOW}(B) = \{ \epsilon, \% \}

everything in \text{FOL}(S) is in \text{FOL}(B)
Example:

S → BCD | aD
A → CEB | aA
B → b | λ
C → dB | λ
D → cA | λ
E → e | fE

FOLLOW(S) = \{3\}
FOLLOW(A) = FOLLOW(D) = \{3\}
FOLLOW(B) = FOLLOW(C) = \{3, e\}
FOLLOW(D) = FOLLOW(S) = \{3\}
FOLLOW(E) = \{b, \$\}