Section: Parsing

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

Consider the CFG $G$:

- $S \rightarrow Aa$
- $A \rightarrow AA \mid ABa \mid \lambda$
- $B \rightarrow BBa \mid b \mid \lambda$

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

New grammar $G'$ is:

- $S \rightarrow Aa \mid a$
- $A \rightarrow AA \mid ABa \mid Aa \mid Ba \mid a$
- $B \rightarrow BBa \mid Ba \mid a \mid b$

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

If \( w \xrightarrow{*} \lambda \) then
\[ \lambda \text{ is in FIRST}(w) \]
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 FIRST(A) then
       Everything in FIRST(w) is in FIRST(X)
3. In general, $\text{FIRST}(X_1X_2X_3..X_K) =$

- $\text{FIRST}(X_1)$
- $\cup \text{FIRST}(X_2)$ 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 $\text{FIRST}(X_{K-1})$
- $\setminus \{\lambda\}$ if $\lambda \notin \text{FIRST}(X_J)$ for all $J$
Example:

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

FIRST(B) =
FIRST(S) =
FIRST(Sc) =
Example

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

\text{FIRST}(S) = \]
\text{FIRST}(A) = \]
\text{FIRST}(B) = \]
\text{FIRST}(C) = \]
\text{FIRST}(D) = \]
\text{FIRST}(E) = \]
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. $\$ \text{ is in FOLLOW}(S)$

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:

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

\[\text{FOLLOW}(S) =\]
\[\text{FOLLOW}(B) =\]
Example:

\[
\begin{align*}
S & \rightarrow BCD \mid aD \\
A & \rightarrow CEB \mid aA \\
B & \rightarrow b \mid \lambda \\
C & \rightarrow dB \mid \lambda \\
D & \rightarrow cA \mid \lambda \\
E & \rightarrow e \mid fE
\end{align*}
\]

\[
\begin{align*}
\text{FOLLOW}(S) & = \\
\text{FOLLOW}(A) & = \\
\text{FOLLOW}(B) & = \\
\text{FOLLOW}(C) & = \\
\text{FOLLOW}(D) & = \\
\text{FOLLOW}(E) & =
\end{align*}
\]