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?

$ba$ not in $L(G)$

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?

$ba \notin L(G)$
Top-down Parser:

- Start with S and try to derive the string.

\[ S \rightarrow aS \mid b \]

Example: 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 \( w \Rightarrow^* av \) then \( a \) is in FIRST(w)
- If \( w \Rightarrow^* \lambda \) then \( \lambda \) is in FIRST(w)
To compute FIRST:

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

2. \( \text{FIRST}(X) \)

   (a) If \( X \to aw \) then
       \( a \) is in \( \text{FIRST}(X) \)

   (b) IF \( X \to \lambda \) then
       \( \lambda \) is in \( \text{FIRST}(X) \)

   (c) If \( X \to Aw \) and \( \lambda \in \text{FIRST}(A) \) then
       Everything in \( \text{FIRST}(w) \) is in \( \text{FIRST}(X) \)
3. In general, 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:

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

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

Sc \Rightarrow aSc

Sc \Rightarrow Bc \Rightarrow bc

Sc \Rightarrow Bc \Rightarrow c

Sc \Rightarrow Bc \Rightarrow c
Example

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

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

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

To compute \( \text{FOLLOW} \):

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

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

\[ \text{FOLLOW}(S) = \{ \text{c}, \$ \} \]
\[ \text{FOLLOW}(B) = \{ \$ , \text{c} \} \]

\[ S \rightarrow aSc \rightarrow aBc \]

\[ S \rightarrow B \]

\[ \uparrow \]
Example:

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

\[
\text{FOLLOW}(S) = \{ \$, \# \} \\
\text{FOLLOW}(A) = \{ \$, d, c, e, \# \} \\
\text{FOLLOW}(B) = \{ \$, d, c, e, \# \} \\
\text{FOLLOW}(C) = \{ \$, d, c, e, \# \} \\
\text{FOLLOW}(D) = \{ \$, \# \} \\
\text{FOLLOW}(E) = \{ b, \$ \}
\]