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

Given a context-free grammar \( G = (V, T, S, P) \), \( a \in T \) and \( w, v \in (V \cup T)^* \), the FIRST\((w)\) is the set of terminals that can be the first terminal \( a \) in \( w \Rightarrow^* av \). \( \lambda \) is in FIRST\((w)\) if \( w \Rightarrow^* \lambda \).

We show how to calculate FIRST for variables and terminals in the grammar, for \( \lambda \) and for strings.
Algorithm for FIRST

Given a grammar $G = (V, T, S, P)$, calculate FIRST($w$) for $w$ in $(V \cup T)^*$,

1. For $a \in T$, $FIRST(a) = \{a\}$.
2. $FIRST(\lambda) = \{\lambda\}$.
3. For $A \in V$, set $FIRST(A) = \{}$.
4. Repeat these steps until no more terminals or $\lambda$ can be added to any FIRST set for variables.

For every production $A \rightarrow w$

$$FIRST(A) = FIRST(A) \cup FIRST(w)$$
5. For $w = x_1x_2x_3 \ldots x_n$ where 
$x_i \in (V \cup T)$

a) $\text{FIRST}(w) = \text{FIRST}(x_1)$
b) For $i$ from 2 to $n$ do:
   if $x_j \Rightarrow \lambda$ for all $j$ from 1 to $i - 1$ then
   $\text{FIRST}(w) = \text{FIRST}(w) \cup \text{FIRST}(x_i) - \{\lambda\}$
c) If $x_i \Rightarrow \lambda$ for all $i$ from 1 to $n$ then
   $\text{FIRST}(w) = \text{FIRST}(w) \cup \{\lambda\}$
Example:

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

FIRST(B) = ₃b, ₂c

FIRST(S) = FIRST(aSc) v FIRST(B)

FIRST(Sc) = ₃a, b, ₂c, ₃
Example

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

\[
\begin{align*}
\text{FIRST}(S) &= \{ b, d, c, a, \lambda \} \\
\text{FIRST}(A) &= \{ a, d, e, f \} \\
\text{FIRST}(B) &= \{ b, \lambda \} \\
\text{FIRST}(C) &= \{ d, \lambda \} \\
\text{FIRST}(D) &= \{ c, \lambda \} \\
\text{FIRST}(E) &= \{ e, f \}
\end{align*}
\]
Definition: FOLLOW

Given a context-free grammar $G = (V, T, S, P)$, $A \in V$, $a \in T$ and $w, v \in (V \cup T)^*$, $\text{FOLLOW}(A)$ is the set of terminals that can be the first terminal $a$ immediately following $A$ in some sentential form $vAaw$. \$ is always in $\text{FOLLOW}(S)$. 
Algorithm for FOLLOW

To calculate FOLLOW for the variables in G=(V,T,S,P). Let A, B ∈ V and v, w ∈ (V ∪ T)*.

1. $ is in FOLLOW(S).

2. For A → vB, FOLLOW(A) is in FOLLOW(B).

3. For A → vBw:
   (a) FIRST(w) − {λ} is in FOLLOW(B).
   (b) If λ ∈ FIRST(w), then FOLLOW(A) is in FOLLOW(B).
Example:

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

\[
\text{FOLLOW}(S) = \{ \$, \langle, ? \}
\]
\[
\text{FOLLOW}(B) = \{ \$, \langle \}
\]
Example:

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

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