Parsing

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

**Review**
Consider the CFG $G$:

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

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:

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

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

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.

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

**Definition: FIRST**

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

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∪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) FIRST(\( w \)) = 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
      
      FIRST(\( w \)) = FIRST(\( w \)) \cup FIRST(\( x_i \)) - \{\( \lambda \}\)
   
   c) If \( x_i \rightarrow^* \lambda \) for all \( i \) from 1 to \( n \) then
      
      FIRST(\( w \)) = FIRST(\( w \)) \cup \{\( \lambda \}\)

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

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

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

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 \( A, B \in V \) and \( v, w \in (V \cup T)^* \).

1. $ is in \( \text{FOLLOW}(S) \).
2. For \( A \rightarrow vB \), \( \text{FOLLOW}(A) \) is in \( \text{FOLLOW}(B) \).
3. For \( A \rightarrow vBw \):
   (a) \( \text{FIRST}(w) - \{\lambda\} \) is in \( \text{FOLLOW}(B) \).
   (b) If \( \lambda \in \text{FIRST}(w) \), then \( \text{FOLLOW}(A) \) is in \( \text{FOLLOW}(B) \).
Example:

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

FOLLOW(S) =
FOLLOW(B) =

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