Parsing

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

Review

Consider the CFG \(G:\)

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

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:

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

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.

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

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

If \( w \xrightarrow{*} av \) then

- \( a \) is in FIRST(w)

If \( w \xrightarrow{*} \lambda \) then

- \( \lambda \) 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 \( \lambda \in \text{FIRST(A)} \) then
       - Everything in FIRST(w) is in FIRST(X)

3. In general, FIRST(X_1X_2X_3..X_K) =
   
   - FIRST(X_1)
   - \( \cup \) FIRST(X_2) if \( \lambda \) is in FIRST(X_1)
   - \( \cup \) FIRST(X_3) if \( \lambda \) is in FIRST(X_1) and \( \lambda \) is in FIRST(X_2)
   
   ...  
   - \( \cup \) FIRST(X_K) if \( \lambda \) is in FIRST(X_1) and \( \lambda \) is in FIRST(X_2) ...
   
   - \( \cup \) FIRST(X_2) if \( \lambda \) is in FIRST(X_1) and \( \lambda \) is in FIRST(X_2) ...
   
   - \{\lambda\} if \( \lambda \notin \text{FIRST}(X_J) \) for all J
Example: \( L = \{a^n b^m c^n : n \geq 0, 0 \leq m \leq 1\} \)

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

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

Example

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

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

Definition: FOLLOW(X) = set of terminals that can appear to the right of X in some derivation.

If \( S \xrightarrow{} wAav \) then 
\( a \) is in FOLLOW(A)

(where \( w \) and \( v \) are strings of terminals and variables, \( a \) is a terminal, and \( A \) is a variable)
To compute FOLLOW:

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

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) =