Section: Parsing

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

Consider the CFG G:

$$\begin{aligned} S &\rightarrow Aa \\ A &\rightarrow AA \mid ABa \mid \lambda \\ B &\rightarrow BBa \mid b \mid \lambda \end{aligned}$$

Is ba in L(G)? Running time?

New grammar G' is:

$$\begin{aligned} S &\rightarrow Aa \mid a \\ A &\rightarrow AA \mid ABa \mid Aa \mid Ba \mid a \\ B &\rightarrow BBa \mid Ba \mid a \mid b \end{aligned}$$

Is ba in L(G)? Running time?

Top-down Parser:

- **Start with S and try to derive the string.**

$$\mathbf{S \rightarrow aS \mid b}$$

- **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:

$$\mathbf{G=(V,T,S,P)}$$
$$\mathbf{w,v{\in}(V{\cup}T)^*}$$
$$\mathbf{a{\in}T}$$
$$\mathbf{X,A,B{\in}V}$$
$$\mathbf{X}_I \in\mathbf{(V{\cup}T)^+}$$

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** $w \stackrel{*}{\Rightarrow} av$**.** λ **is in FIRST(**w**) if** $w \stackrel{*}{\Rightarrow} \lambda$**.**

We show how to calculate FIRST for variables and terminals in the grammar, for λ **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(λ) = $\{\lambda\}$.**
3. **For $A \in V$, set FIRST(A) = $\{\}$.**
4. **Repeat these steps until no more terminals or λ can be added to any FIRST set for variables.**

   For every production $A \to w$
   FIRST(A) = FIRST(A) $\cup$ FIRST(w)

5. For $w = x_1 x_2 x_3 \ldots x_n$ where $x_i \in (V \cup T)$

a) FIRST(w) = FIRST(x_1)

b)

For i from 2 to n do:

  if $x_j \stackrel{*}{\Rightarrow} \lambda$ for all j from 1 to $i-1$ then

    FIRST(w) = FIRST(w) $\cup$ FIRST(x_i) - $\{\lambda\}$

c)

If $x_i \stackrel{*}{\Rightarrow} \lambda$ for all i from 1 to n then

  FIRST(w) = FIRST(w) $\cup$ $\{\lambda\}$

Example:

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{aSc} \mid \mathbf{B} \\ \mathbf{B} &\rightarrow \mathbf{b} \mid \lambda \end{aligned}$$

FIRST(B) =

FIRST(S) =

FIRST(Sc) =

Example

$$
\begin{aligned}
\mathbf{S} &\rightarrow \mathbf{BCD} \mid \mathbf{aD} \\
\mathbf{A} &\rightarrow \mathbf{CEB} \mid \mathbf{aA} \\
\mathbf{B} &\rightarrow \mathbf{b} \mid \lambda \\
\mathbf{C} &\rightarrow \mathbf{dB} \mid \lambda \\
\mathbf{D} &\rightarrow \mathbf{cA} \mid \lambda \\
\mathbf{E} &\rightarrow \mathbf{e} \mid \mathbf{fE}
\end{aligned}
$$

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** $w, v \in (V \cup T)^*$**, 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 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** $FOLLOW(S)$.
2. **For** $A \to vB$, $FOLLOW(A)$ **is in** $FOLLOW(B)$.
3. **For** $A \to vBw$:
   - **(a)** $FIRST(w) - \{\lambda\}$ **is in** $FOLLOW(B)$.
   - **(b) If** $\lambda \in FIRST(w)$, **then** $FOLLOW(A)$ **is in** $FOLLOW(B)$.

Example:

$$\mathbf{S} \rightarrow \mathbf{aSc} \mid \mathbf{B}$$
$$\mathbf{B} \rightarrow \mathbf{b} \mid \lambda$$

FOLLOW(S) =

FOLLOW(B) =

Example:

$$\begin{aligned}
\mathbf{S} &\rightarrow \mathbf{BCD} \mid \mathbf{aD} \\
\mathbf{A} &\rightarrow \mathbf{CEB} \mid \mathbf{aA} \\
\mathbf{B} &\rightarrow \mathbf{b} \mid \lambda \\
\mathbf{C} &\rightarrow \mathbf{dB} \mid \lambda \\
\mathbf{D} &\rightarrow \mathbf{cA} \mid \lambda \\
\mathbf{E} &\rightarrow \mathbf{e} \mid \mathbf{fE}
\end{aligned}$$

FOLLOW(S) =

FOLLOW(A) =

FOLLOW(B) =

FOLLOW(C) =

FOLLOW(D) =

FOLLOW(E) =