$LL(k)$ Grammars

Adding Lookahead

- Goal: Resolve the nondeterminism of NTAs without using backtracking by allowing the automaton to take the next $k \in \mathbb{N}$ input symbols into account when doing an expansion
- This is called a *lookahead* of k input symbols
- Requires that we give the NTA more information.

  For the CFG

  $$\begin{array}{ll} S \rightarrow A \mid B & (1, 2) \\ A \rightarrow b & (3) \\ B \rightarrow Cd & (4) \\ C \rightarrow a & (5) \end{array}$$

  the automaton needs to know that
  - expanding S to A only makes sense if the next input symbol is b
  - expanding S to B only makes sense if the next input symbol is a
- In that way, parsing the input ab becomes deterministic
- Fortunately, that information is already in the CFG!

$first_k$ set

- For $G =< \Sigma, N, P, S >$, the $first_k$ set of $\alpha \in (N \cup \Sigma)^*$ is given by:

$$first_k(\alpha) = \{v \in \Sigma^k | \exists w \in \Sigma^*: \alpha \Rightarrow^* vw\} \cup \{v \in \Sigma^{<k} | \alpha \Rightarrow^* v\}$$

- Example: For the CFE

$$S \to aSb \mid \varepsilon$$

  we have

$$first_1(ab) = \{a\}$$
$$first_2(ab) = \{ab\}$$
$$first_2(a) = \{a\}$$
$$first_1(S) = \{\varepsilon, a\}$$
$$first_2(S) = \{\varepsilon, ab, aa\}$$
$$first_3(S) = \{\varepsilon, ab, aab, aaa\}$$
$$first_3(Sa) = \{a, aba, aab, aaa\}$$

- Note that $|first_k(\alpha)| = 1$ for $\alpha \in \Sigma^*$

$LL(k)$ Grammar

- $LL(k)$ = "reading input from Left to right with k-lookahead and Leftmost derivation"
- The CFE $G = < \Sigma, N, P, S >$ is an $LL(k)$ grammar for a given $k \in \mathbb{N}$ if for all leftmost derivations of the form

$$S \Rightarrow_l^* wA\alpha \begin{cases} \Rightarrow_l w\beta\alpha \Rightarrow_l^* wx \\ \Rightarrow_l w\gamma\alpha \Rightarrow_l^* wy \end{cases} \text{such that } \beta \neq \gamma$$

  it follows that $first_k(x) \neq first_k(y)$
- This means that in an $LL(k)$ grammar different productions x and y do not have the same k-lookahead
- This is interesting because it means that the decision to expand $A \rightarrow \beta$ or $A \rightarrow \gamma$ is determined by the next k symbols following w
- Problem: If we want to verify whether G is a $LL(k)$ do we have to verify all possible productions $\beta\alpha \Rightarrow_l^* x$ and $\gamma\alpha \Rightarrow_l^* y$? (there could be an infinite number)

$LL(k)$ Grammar, part 2

- There is a useful lemma that we will not prove here:
- $G =< \Sigma, N, P, S >$ is an $LL(k)$ grammar if and only if for all leftmost derivations of the form

$$S \Rightarrow_l^* wA\alpha \begin{cases} \Rightarrow_l w\beta\alpha \Rightarrow_l^* wx \\ \Rightarrow_l w\gamma\alpha \Rightarrow_l^* wy \end{cases} \text{ such that } \beta \neq \gamma$$

  it follows that $first_k(\beta\alpha) \cap first_k(\gamma\alpha) = \emptyset$
- This means that $A \rightarrow \beta$ or $A \rightarrow \gamma$ is determined by the $first_k$ sets of $\beta\alpha$ and $\gamma\alpha$

Example with $k = 1$ *lookahead*

- CFG

$$
\begin{array}{ll}
S \rightarrow A \mid B & (1, 2) \\
A \rightarrow b & (3) \\
B \rightarrow Cd & (4) \\
C \rightarrow a & (5)
\end{array}
$$

- Initial state (ad, S, ε) of NTA for input ad
- Should we use rule 1 or rule 2?
  - $first_1(A) = \{b\}$
  - $first_1(B) = \{a\}$
- Because the next input symbol is a, we know that rule 2 is the correct one. Next state will be $(ad, B, 2)$