Lesson 6: Pumping lemma and push-down automata

Theme: Pumping lemma and push-down automata as a model of computation for context-free languages.

1 Pumping lemma

Like regular languages, context-free languages also have property that their words can be “pumped.”

Lemma 6.1 (pumping lemma) Let \( G = (\Sigma, V, R, S) \) be a CFG. Suppose \( M \) is the maximum of all the \(|w|\)'s that appear on the right hand side of the rules in \( R \), i.e.,

\[
M = \max_{A \rightarrow w \in R} |w|.
\]

For every \( u \in L(G) \) such that \(|u| \geq M |R| + 1\), \( u \) can be partitioned into:

\[
u = s x y z t \quad \text{such that } |x| + |z| \geq 1
\]

and for every \( i \geq 0 \):

\[
sx^iyzt \in L(G).
\]

Using Lemma 6.1, we can show that the language \( L = \{a^kb^kc^k \mid k \geq 0\} \) is not a CFL. Since \( L \) is the intersection of two CFL's \( \{a^mb^nc^n \mid m, n \geq 0\} \) and \( \{a^n b^n c^m \mid m, n \geq 0\} \), it also shows that CFL's are not closed under intersection.

2 Push-down automata

In the following we will have two alphabets \( \Sigma \) and \( \Gamma \).

A push-down automaton (PDA) is a system \( A = (\Sigma, \Gamma, Q, q_0, F, \delta) \), where each of the component is as follows.

- \( \Sigma \) is a finite alphabet, called the input alphabet, whose elements are called input symbols.
- \( \Gamma \) is a finite alphabet, called the stack alphabet, whose elements are called stack symbols.
- \( Q \) is a finite set of states.
- \( q_0 \in Q \) is the initial state.
- \( F \subseteq Q \) is the set of final states.
- \( \delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q \times (\Gamma \cup \{\epsilon\}) \) is the transition relation.

We will usually write a transition \((p, x, y, q, z) \in \delta\) as:

\[(p, x, \text{pop}(y)) \rightarrow (q, \text{push}(z))\]

Intuitively, such transition means that when a PDA is in state \( p \) reading \( x \) from the input and the top of the stack is \( y \), it can “pop” \( y \) from the top of the stack and moves to state \( q \) and push \( z \) onto the stack. Here it is possible that \( x, y \), and \( z \) are the empty string \( \epsilon \).
Note that the fashion a symbol is written into and taken out of the stack is “Last In First Out” (LIFO), i.e., the last symbol that gets written into the stack has to come out first. It is also important to note that while the input is a word over $\Sigma$, its stack contains symbols from $\Gamma$.

We will now describe formally how PDA computes. Let $A = (\Sigma, \Gamma, Q, q_0, F, \delta)$ be a PDA. A configuration of $A$ is a pair $(q, u) \in Q \times \Gamma^*$, where $q$ is the state of $A$ and $u$ is the content of the stack. The initial configuration is $(q_0, \epsilon)$. A configuration is final, if the state component is one of the final states.

On input $w = a_1 \ldots a_m$, a run of a PDA from a configuration $(q, u)$ is a sequence:

$$(p_0, v_0) \vdash_{b_1} (p_1, v_1) \vdash_{b_2} \cdots \vdash_{b_n} (p_n, v_n),$$

where

- $(p_0, v_0) = (q, u)$,
- $b_1 \cdots b_n = a_1 \cdots a_m$, i.e., some of the $b_i$’s can be $\epsilon$,
- for each $i = 1, \ldots, n$, there is $(p_i, x, \text{pop}(y)) \rightarrow (p_{i+1}, \text{push}(z)) \in \delta$ such that
  - $x = b_i$,
  - $v_i = ys$ and $v_{i+1} = zs$, for some $s \in \Gamma^*$.

Note: Here $v_i = ys$ denotes that the content of the stack is $ys$, where the top of the stack is $y$. When the transition $(p_i, x, \text{pop}(y)) \rightarrow (p_{i+1}, \text{push}(z))$ is applied, the PDA is in state $p_i$ reads $x$ from the input, “pop” $y$ from the stack, and moves to state $p_{i+1}$ and at the same time “push” $z$ onto the stack. Thus, the subsequent content $v_{i+1}$ of the stack is $zs$.

A run is accepting, if it starts from the initial configuration and ends with a final configuration. The language accepted by $A$, denoted by $L(A)$, consists of all the words for which it has an accepting run. Formally,

$$L(A) = \{ \ w \ | \ \text{there is an accepting run of } A \text{ on } w \}.$$