Lesson 4: Push-down automata

Theme: Push-down automata as a model of computation for context-free languages.

In the following we will have two alphabets $\Sigma$ and $\Gamma$. A push-down automaton (PDA) is a system $A = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$, 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 accepting 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 $A = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$ 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 accepting, if the state component is one of the accepting 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
  - $x = b_i$,
  - $v_i = sy$ and $v_{i+1} = sz$, for some $s \in \Gamma^*$.

Note: Here $v_i = sy$ denotes that the content of the stack, 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 $sz$. 

1/2
We will also write the run as:

\[(p_0, v_0) \xrightarrow{+} w (p_n, v_n).\]

In this case we will also say that there is a run of \( \mathcal{A} \) on \( w \) from \( (p_0, v_0) \) to \( (p_n, v_n) \).

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

\[ L(\mathcal{A}) = \{ w \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w \}. \]

The following theorem states that every CFL is accepted by a PDA.

**Theorem 4.1**

- For every CFG \( \mathcal{G} \), there is a PDA \( \mathcal{A} \) such that \( L(\mathcal{A}) = L(\mathcal{G}) \).
- Vice versa, for every PDA \( \mathcal{A} \), there is a CFG \( \mathcal{G} \) such that \( L(\mathcal{A}) = L(\mathcal{G}) \).

The proof is a bit technical. We will discuss it in our next lesson. One immediate consequence of Theorem 4.1 is that the intersection of a regular language and a CFL is a CFL.

**Theorem 4.2** If \( K \) is CFL and \( L \) is regular language, then their intersection \( K \cap L \) is CFL.

**Proof.** By Theorem 4.1, let \( \mathcal{A}_1 = (\Sigma, \Gamma, Q_1, q_0,1, F_1, \delta_1) \) be a PDA that accepts \( K \). Let \( \mathcal{A}_2 = (\Sigma, Q_2, q_0,2, F_2, \delta_2) \) be an NFA that accepts \( L \).

Construct the following PDA \( \mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \delta) \) that simulates both \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) simultaneously.

- \( Q = Q_1 \times Q_2 \).
- \( q_0 = (q_0,1, q_0,2) \).
- \( F = F_1 \times F_2 \).
- \( \delta \) is defined as follows.

  - For every \( (p_1, x, \text{pop}(y) \rightarrow (q_1, \text{push}(z))) \in \delta_1 \), where \( x \neq \epsilon \) and \( (p_2, x, q_2) \in \delta_2 \), the following transition is in \( \delta \):

    \[ ((p_1, p_2), x, \text{pop}(y) \rightarrow ((q_1, q_2), \text{push}(z)) \]

  - For every \( (p_1, x, \text{pop}(y) \rightarrow (q_1, \text{push}(z))) \in \delta_1 \), where \( x = \epsilon \), for every \( p_2 \in Q_2 \), the following transition is in \( \delta \):

    \[ ((p_1, p_2), x, \text{pop}(y) \rightarrow ((q_1, p_2), \text{push}(z)) \]

That \( \mathcal{A} \) accepts precisely \( L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \) can be proved in a similar manner as the proof that regular languages are closed under intersection.