Push Down Automata

\[ P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F). \]

- \( Q \): Finite set of states; \( q_0 \) is the start state; \( F \) is the set of final/accepting states.
- \( \Sigma \): Alphabet set; \( \Gamma \): Stack alphabet
- \( Z_0 \) is the (only) initial symbol on the stack.
- \( \delta \): transition function.

\( \delta \) takes as input a state \( q \), an input letter \( a \) (or \( \epsilon \)), and a stack symbol (top of stack) \( X \). \( \delta(q, a, X) \) is then a finite subset of \( Q \times \Gamma^* \).
$(p, \gamma) \in \delta(q, a, X)$ denotes that when in state $q$, reading symbol $a$ (or $\epsilon$), with top of stack being $X$, the machine’s new state is $p$, $X$ at the top of stack is popped and $\gamma$ is pushed to the stack. (By convention, if $\gamma = RS$, then $S$ is pushed first, and then $R$ is pushed on the stack).
Instantaneous Descriptions

\((q, w, \alpha)\): denotes that current state is \(q\), input left to read is \(w\), and \(\alpha\) is on the stack (first symbol of \(\alpha\) is top of stack).

\((q, aw, X\alpha) \vdash (p, w, \beta \alpha)\), if \((p, \beta) \in \delta(q, a, X)\) (here \(a\) can be \(\epsilon\)).

One can similarly define \(\vdash_P^*\) (or simply \(\vdash^*\), where \(P\) is understood).

1. \(I \vdash^* I\)
2. \(I \vdash^* J\) and \(J \vdash K\), then \(I \vdash^* K\)
Language accepted by PDA

Acceptance by final state.
\[{w \mid (q_0, w, Z_0) \vdash^*_P (q_f, \epsilon, \alpha), \text{ for some } q_f \in F}\}.

Acceptance by empty stack.
\[{w \mid (q_0, w, Z_0) \vdash^*_P (q, \epsilon, \epsilon), \text{ for some } q \in Q}\}.\]
From Acceptance using empty stack to Acceptance using Final State

Intuition: Initially put a special symbol onto the stack. If ever see the top of stack as that symbol, then go to final state.

\[ P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F). \]
\[ P_F = (Q \cup \{ p_0, p_f \}, \Sigma, \Gamma \cup \{ X_0 \}, \delta_F, p_0, X_0, \{ p_f \}). \]

1. \( \delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}. \)

2. For all \( Z \in \Gamma, a \in \Sigma \cup \{ \epsilon \}: \delta_F(p, a, Z) \) contains all \( (q, \gamma) \) which are in \( \delta(p, a, Z) \).

3. \( \delta_F(p, \epsilon, X_0) \) contains \( (p_f, \epsilon) \), for all \( p \in Q. \)
From Acceptance using final state to Acceptance using empty Stack

Place a transition from final state to a special state which empties the stack.

\( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \).

\( P_E = (Q \cup \{ p_0, p_f \}, \Sigma, \Gamma \cup \{ X_0 \}, \delta_E, p_0, X_0, \{ p_f \}). \)

1. \( \delta_E(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}. \)

2. \( \delta_E(p, a, Z) \) contains all \((q, \gamma)\) which are in \( \delta(p, a, Z) \), for all \( Z \in \Gamma \) and \( a \in \Sigma \cup \{ \epsilon \} \).

3. \( \delta_E(p, \epsilon, Z) \) contains \((p_f, \epsilon)\), for all \( p \in F \), and \( Z \in \Gamma \cup \{ X_0 \} \).

4. \( \delta_E(p_f, \epsilon, Z) \) contains \((p_f, \epsilon)\), for all \( Z \in \Gamma \cup \{ X_0 \} \).
First we show how to accept a CFG. We use the accepting by empty stack model. Intuitively, do left-most derivation. Use stack to keep track of “what is left to derive”. Each time there is a non-terminal on the top of stack, guess a production to be used and push it on the stack. Terminal symbols can be matched as it is.

Details:

\[ G = (V, T, P, S) \].

Then, construct \( PDA = (\{q_0\}, \Sigma, \Gamma, \delta, q_0, S, F) \), where, \( \Sigma = T \),
\[ \Gamma = V \cup T \].

For all \( a \in \Sigma \), \( \delta(q_0, a, a) = \{(q_0, \epsilon)\} \)

For all \( A \in V \), \( \delta(q_0, \epsilon, A) = \{(q_0, \gamma) : A \rightarrow \gamma \text{ in } P\} \).
Now we show that each language accepted by a PDA (using empty stack) can be accepted by a CFG. Suppose PDA is \((Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)\).

We define grammar \(G = (V, \Sigma, R, S)\) as follows.

\[
V = \{S\} \cup \{[qZp] : q, p \in Q, Z \in \Gamma\}.
\]

\(S \rightarrow [q_0Z_0p]\), for each \(p \in Q\).

If \(\delta(q, a, X)\) contains \((r, Y_1 \ldots Y_k)\), then we have productions of the form:

\[
[qXr_k] \rightarrow a[rY_1r_1][r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k],
\]

for all \(r_1, r_2, \ldots, r_k \in Q\).

Here, if \(k = 0\), then \([qXr] \rightarrow a\) (think of \(r = r_0\)).

Intuitively, \([qXr]\) generates \(w\) iff \((q, w, X) \vdash^* (r, \epsilon, \epsilon)\).

By induction on number of steps of PDA/derivation in the CFG.
Deterministic PDA

1. For all $a \in \Sigma \cup \{\epsilon\}$, $Z \in \Gamma$ and $q \in Q$, there is at most one element in $\delta(q, a, Z)$.
2. If $\delta(q, \epsilon, X)$ is non-empty, then $\delta(q, a, X)$ is empty for all $a \in \Sigma$.

Theorem: There exists a language which is accepted by PDA (NPDA) but not by any DPDA.
Deterministic PDA

Theorem: If we consider acceptance by final state, then every regular language can be accepted by a DPDA.

Theorem: If we consider acceptance by empty stack, then \( \{a, aa\} \) is not accepted by a DPDA.

Any language accepted by a DPDA accepting by empty stack has prefix property: For every \( x, y \) in the language, \( x \) is not a prefix of \( y \).