

Deterministic push-down automata

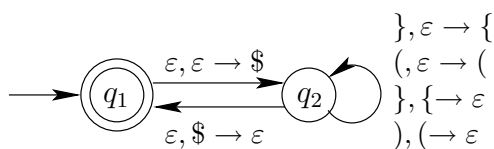
When we talk about deterministic push-down automata (DPDA), our definition will be different from that of finite automata, but the idea remains the same: at any point of execution we shouldn't have more than one choice for the machine to proceed. This means that we might permit transitions on ε as long as no other choices exist for the execution to proceed.

In case of PDAs, we now have two pieces of information that define how we are going to proceed: an input character and a character on the top of the stack. We may choose to ignore one of them or both when we transition from a particular state (i.e., use ε for the input or stack character), but then for the machine to be deterministic no other transitions from that particular state can use something other than ε in other transitions. For example, if we have a transition from q_1 on a/ε , no other transitions from q_1 are allowed to use the stack.

Definition 1 A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is deterministic if there is no configuration for which M has choice of more than one move. That is, it must satisfy the following conditions:

1. For any $q \in Q$, $a \in \Sigma_\varepsilon$, and $s \in \Gamma_\varepsilon$, the set $\delta(q, a, s)$ has at most one element.
2. For any $q \in Q$ and $s \in \Gamma$, if $\delta(q, \varepsilon, s) \neq \emptyset$, then $\delta(q, a, s) = \emptyset$ for every $a \in \Sigma$ and $\delta(q, a, \varepsilon) = \emptyset$ for all $a \in \Sigma_\varepsilon$.
3. For any $q \in Q$ and $a \in \Sigma$, if $\delta(q, a, \varepsilon) \neq \emptyset$, then $\delta(q, a, s) = \emptyset$ for all $s \in \Gamma$ and $\delta(q, \varepsilon, t) = \emptyset$ for all $t \in \Gamma_\varepsilon$.
4. For any $q \in Q$, if $\delta(q, \varepsilon, \varepsilon) \neq \emptyset$, then $\delta(q, a, t) = \emptyset$ for all $a \in \Sigma_\varepsilon$ and $t \in \Gamma_\varepsilon$ (except when $a = \varepsilon$ and $t = \varepsilon$).

Example: The language of balanced strings¹ consisting of brackets $\{\}$ and $()$, which can be represented by the grammar $S \rightarrow SS \mid (S) \mid \{S\} \mid \varepsilon$. We construct a PDA that will store all left brackets on the stack. When a right bracket is observed that matches the left bracket on the stack, we pop it.



The rules $\{, \varepsilon \rightarrow \{$, $(, \varepsilon \rightarrow ($, and $\varepsilon, \$ \rightarrow \varepsilon$ introduce non-determinism because it is not clear if we should read the stack or input. Furthermore, how should we proceed if the top of the stack contains $\$$ and the input is $($? To solve this, we will always read stack in state q_2 . And to ensure that we don't have transitions on $(\{, \$)$ and $(\varepsilon, \$)$, we introduce a new state.

¹A string is balanced if there is a mate of the same type for each bracket and any prefix of the string does not contain more right brackets than left.

