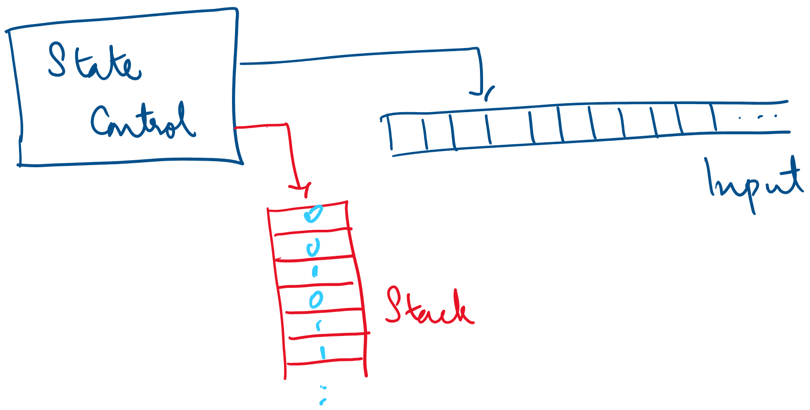
Pushdown Automata

10 February 2021 00:49

Exercise: Prove that Context-Free languages are closed under the regular operations.

Nondeterministic PDA (Pushdown Automata)

* Like NFA, but with a stack for computation.
* Stack: Simple Infinite Memory but restricted access.



In addition to moving between the states, the PDA can also choose to push/pop symbols to/from the stack. The state control also moves based on the stack symbols.

Def 2.13: A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ and F are finite sets.

1. Q is the set of states
2. Σ is the input alphabet

NFA: $Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$

2. Σ is the input alphabet

3. Γ is the stack alphabet

4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function.

NFA : $Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$

5. $q_0 \in Q$ is the start state.

6. $F \subseteq Q$ is the set of accepting states.

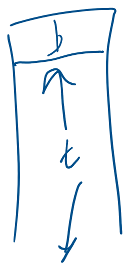
The PDA M computes as follows. It accepts input w if $w = w_1 w_2 \ldots w_m$ where $w_i \in \Sigma_\varepsilon$, and sequence of states $r_0, r_1, r_2, \ldots r_m \in Q$ and $s_0, s_1, s_2 \ldots s_m \in \Gamma^*$ exist

(1) $r_0 = q_0$ and $s_0 = \varepsilon$

Start state Empty stack

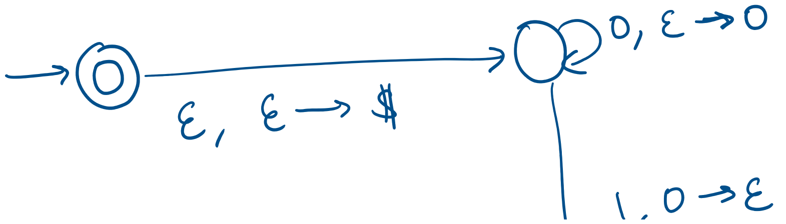
(2) For $i = 0, 1, \ldots m-1$, we have

$$(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$$

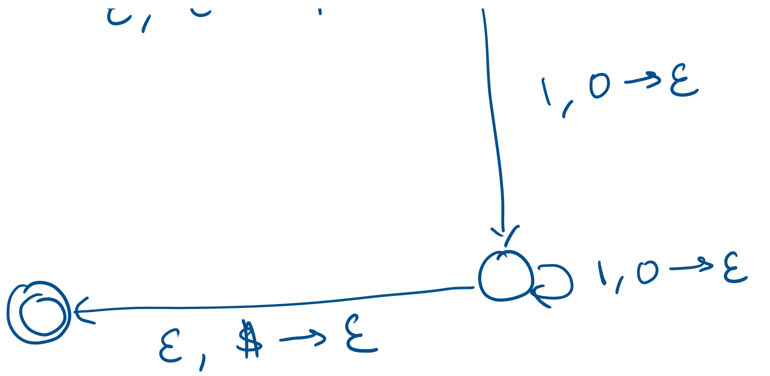


where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$.

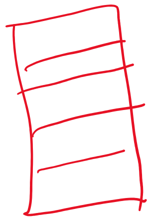
(3) $r_m \in F$. (Stack need not be empty)



L = ?



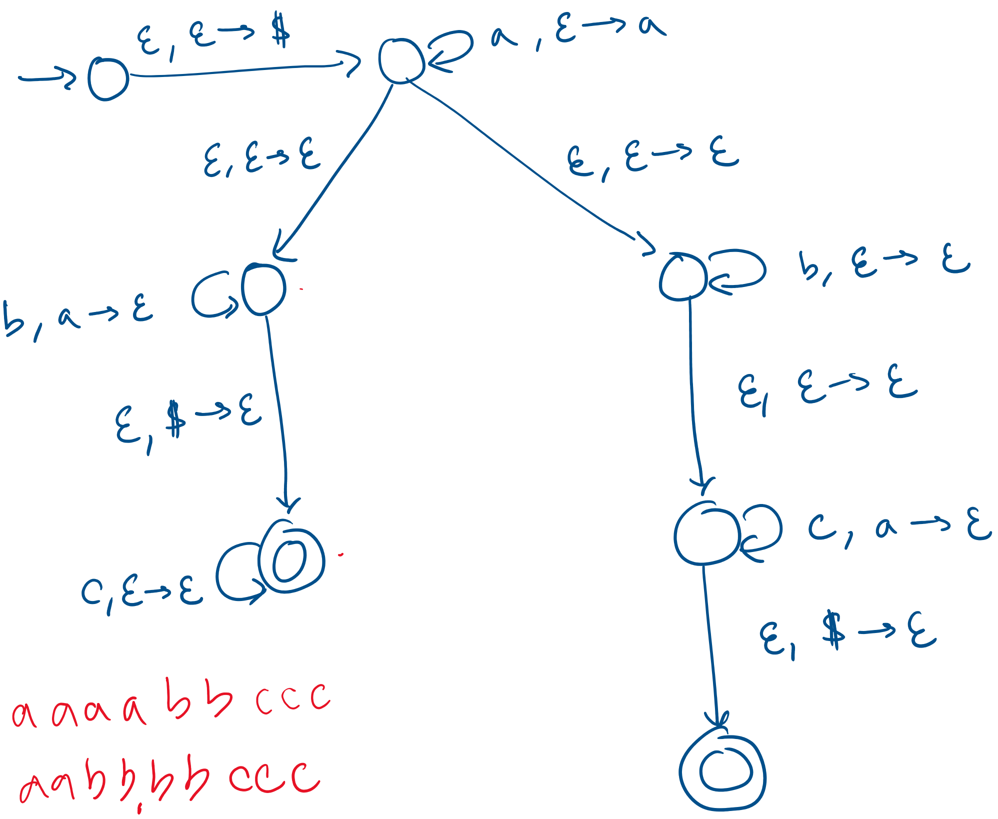
000111



Here we use the $ symbol to check if the stack is empty. The formal definition of PDA has no means to check if the stack is empty.

Exercise: Construct a PDA for all the strings that constitute properly nested parenthesis.

$$L = \{a^i b^j c^k \mid i, j, k \geq 0,\ i = j \text{ or } i = k\}$$

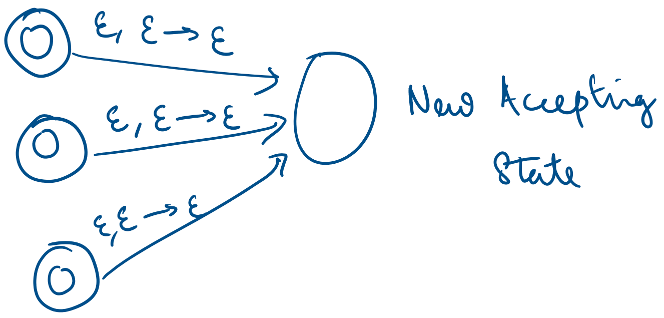


aaaabbccc

aabbbbccc

Some Normalizations

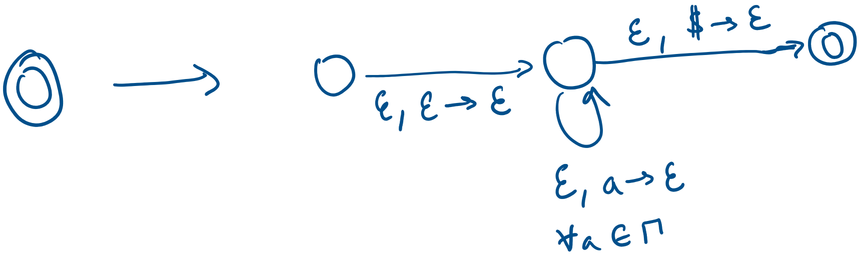
1. We can convert any PDA into a PDA with a single accepting state.



2. We can construct an equivalent PDA that empties stack before accepting.

   1. Put $ into stack initially.

   2. Empty all symbols after original accept. Accept only when $ is popped from the stack.



3. Each transition pushes or pops, but not both.

1. Push only. OK
2. Pop only. OK
3. Both push and pop
4. Neither push nor pop.

Exercise.