Pushdown Automata – Lecture 10

James Marshall

The Pushdown Automaton

Like a nondeterministic finite automaton (NDFA) but:

1. A PDA has a stack:

   a. The PDA can 'pop' (read) a symbol from the top of the stack, and 'push' (write) a symbol onto the stack

   b. The stack is infinite, but only the head can be operated on (see **a**)

2. Deterministic and nondeterministic PDAs are *not* equivalent

Definition

A (nondeterministic) PDA is a 6-tuple $(Q,\Sigma,\Gamma,\delta,q_0,F)$ where Q,Σ,Γ,F are finite sets and

1. Q is the set of *states*,
2. Σ is the *input alphabet*,
3. Γ is the *stack alphabet*,
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accept states

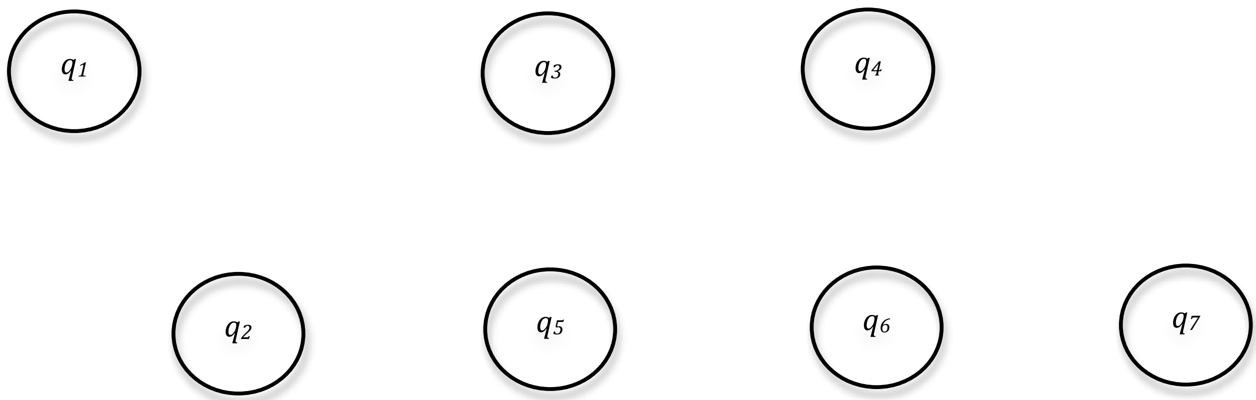
Behaviour

1. To follow a transition, the indicated input symbol must be the next input, and the indicated stack symbol must be at the head of the stack
2. On following a transition, the input is discarded, and the stack is popped. As part of the transition, a new symbol is pushed onto the stack
3. ε-transitions can be followed without reading any input, without popping from the stack, and/or without pushing onto the stack, depending on where the ε symbol appears
4. If more than one valid transition is found, each is followed concurrently in a separate branch
5. If there is no more input and a branch is in an accept state, that branch accepts. If input remains and no valid transitions are found, that branch rejects
6. If any branch accepts, the machine accepts, otherwise it rejects

Example

Design a PDA to recognise $\left\{a^i b^j c^k \mid i,j,k \geq 0 \text{ and } i = j \text{ or } i = k\right\}$

- $\Sigma =$
- $\Gamma =$



Q: *can you see why this language cannot be recognised by a deterministic PDA?*