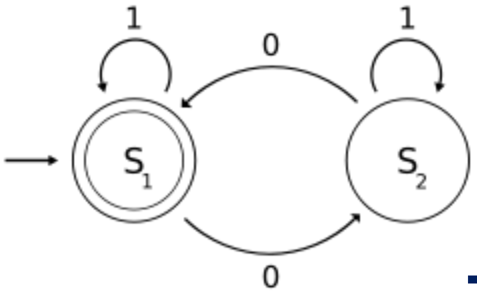
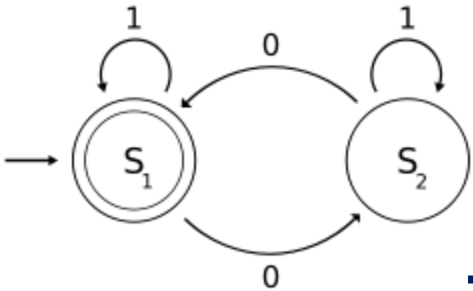


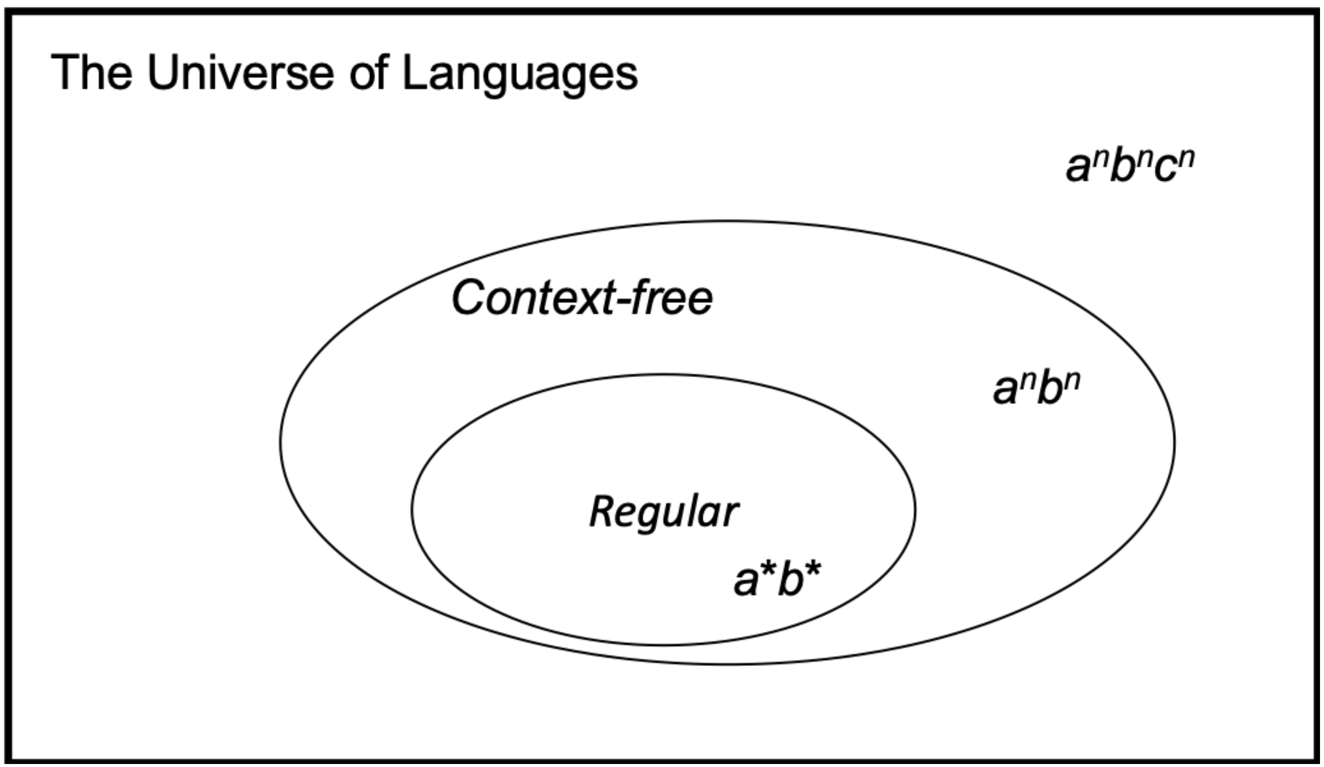
Turing Machines

Sipser: Section 3.1 pages 165 - 175

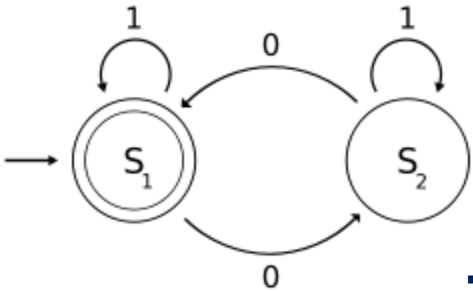


A Context-Free Grammar for $\{a^n b^n c^n : n \geq 0\}$?

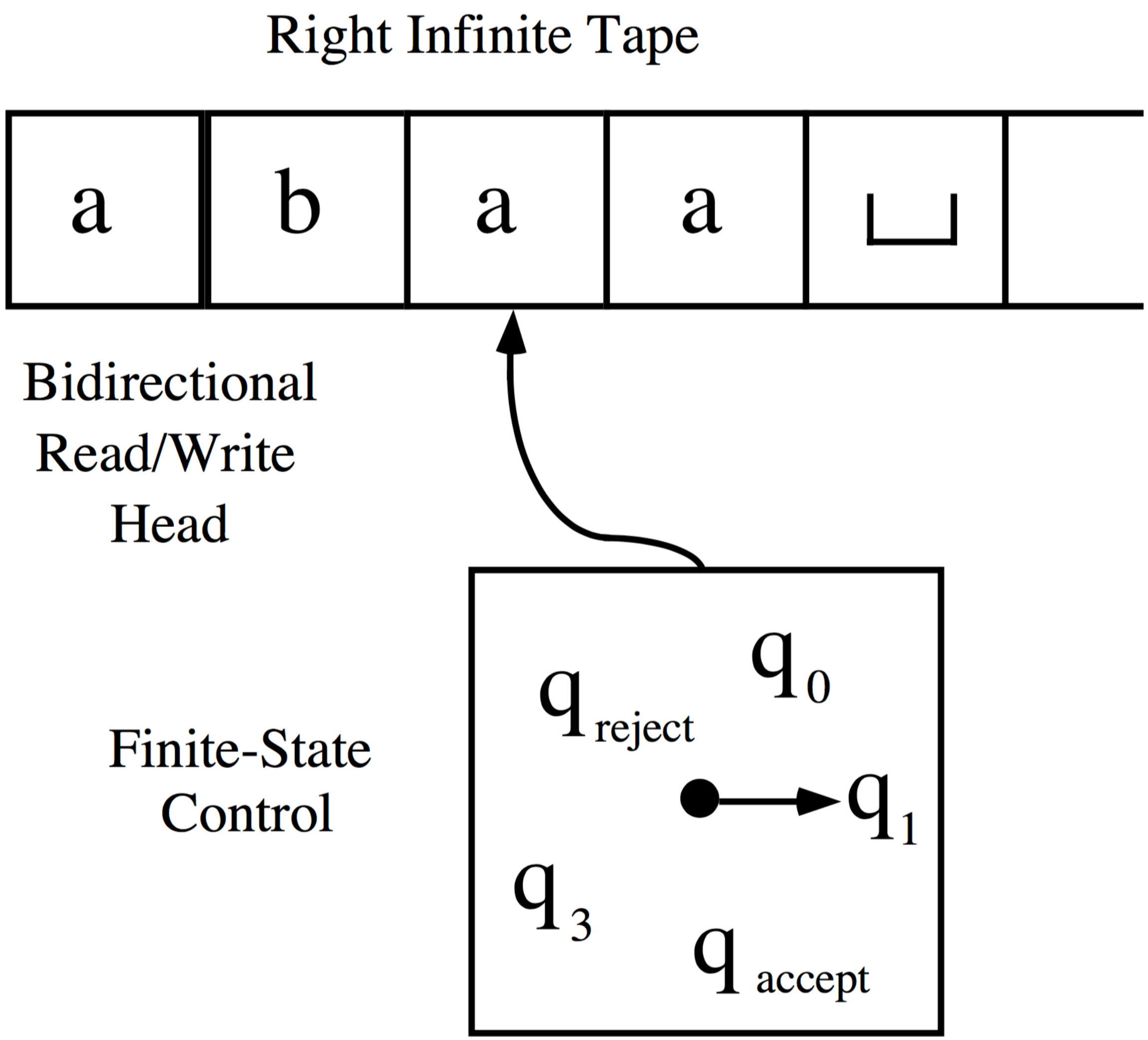
As it turns out, we can't generate one!*

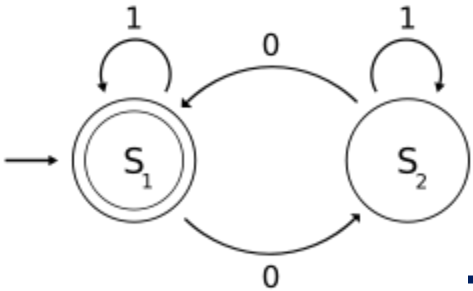


*We can prove this using an analog of the pumping lemma for context-free grammars.

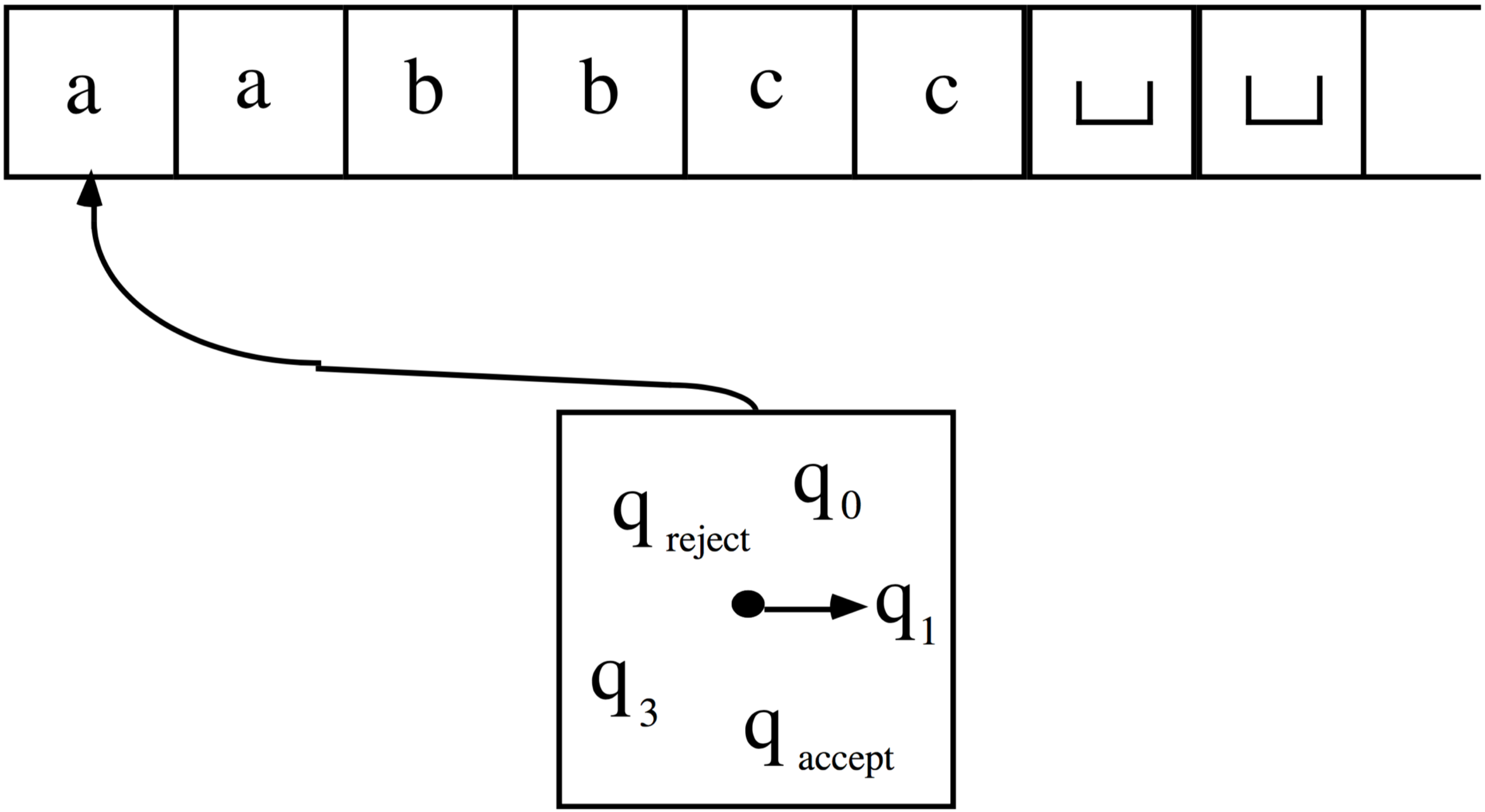


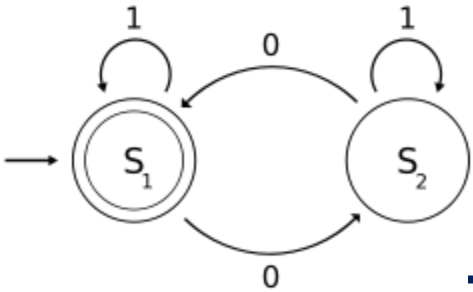
Turing Machines





Recognizing $\{a^n b^n c^n : n \geq 0\}$





Boring …

Definition. A ***Turing Machine*** is a 7-tuple,
$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, Q, Σ, Γ are finite sets,

1. Q is the set of states,

2. Σ is the input alphabet not containing the special ***blank*** symbol $\sqcup$,

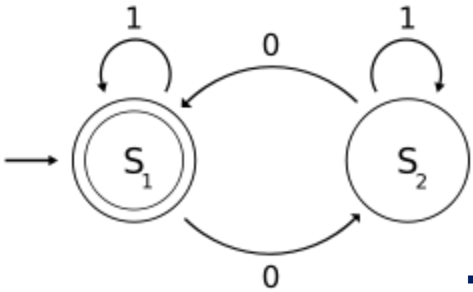
3. Γ is the tape alphabet, where $\{\sqcup\} \in \Gamma$ and $\Sigma \subseteq \Gamma$,

4. $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,

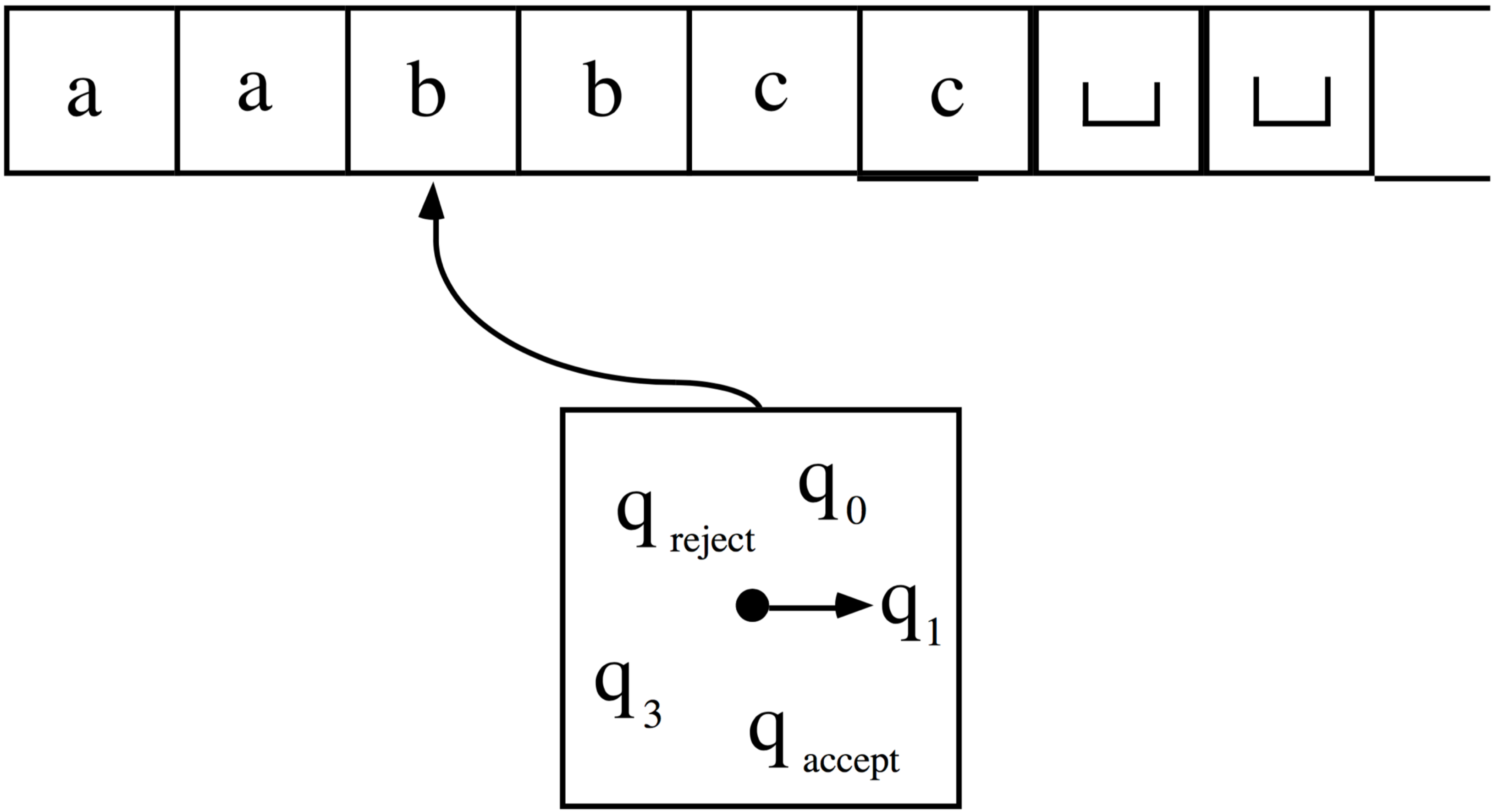
5. $q_0 \in Q$ is the start state,

6. $q_{accept} \in Q$ is the accept state, and

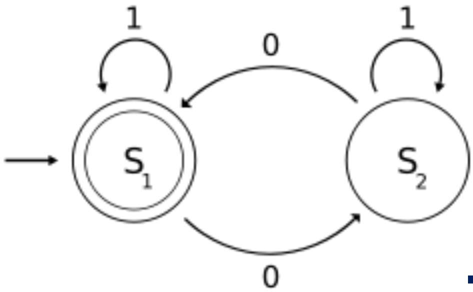
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.



Configurations and Yields



aa$q_1$bbcc ***yields*** aax$q_3$bcc

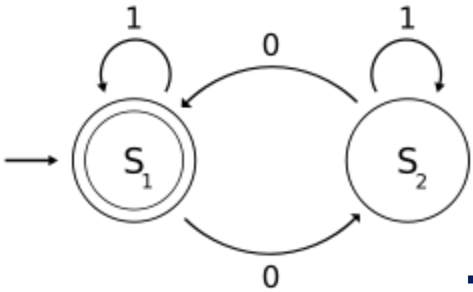


Recursively Enumerable Languages

Definition. A Turing machine M accepts input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists where

1. C_1 is the start configuration of M on input w,
2. each C_i yields C_{i+1}, and
3. C_k is an accepting configuration.

Definition. Call a language ***Turing-recognizable*** if it is the language accepted by some Turing machine.



Recognizing $A = \{0^{2^n} \mid n \geq 0\}$

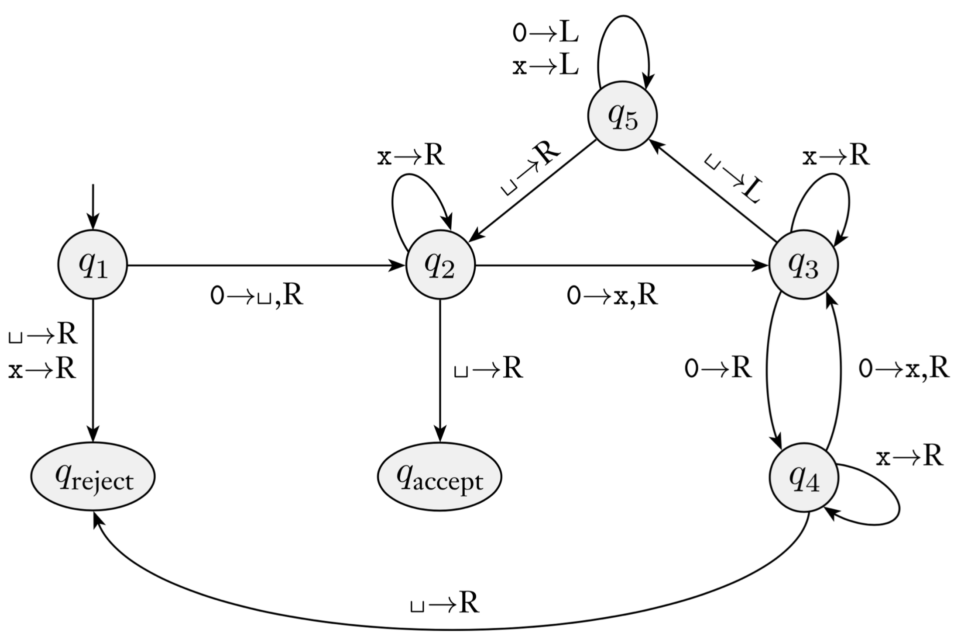
Define $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where

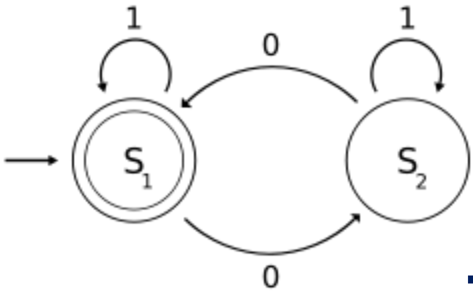
$Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,

$\Sigma = \{0\}$,

$\Gamma = \{0, x, \sqcup\}$, and

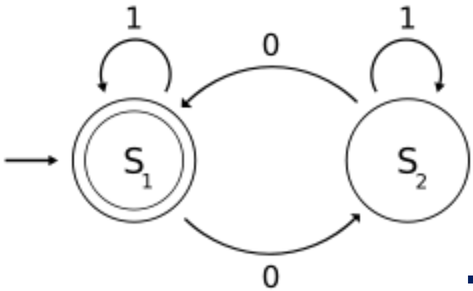
$\delta = Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is given by





Recognizing $B = \{w\#w \mid w \in \{0,1\}^*\}$

Practice!



Recognizing $B = \{w\#w \mid w \in \{0,1\}^*\}$

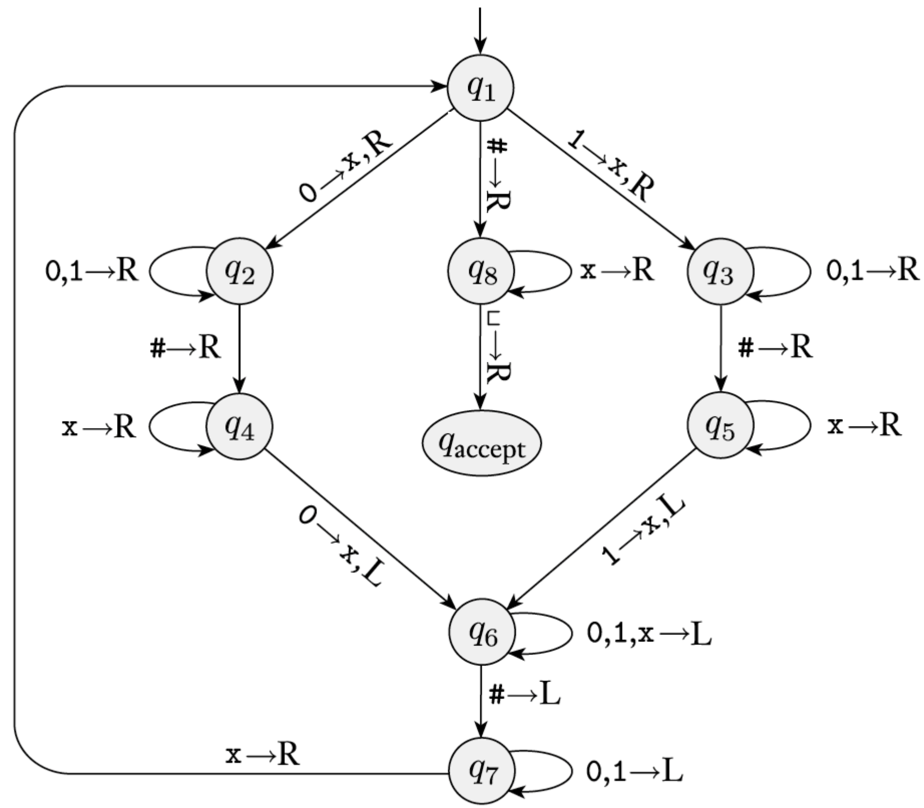
Define $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where

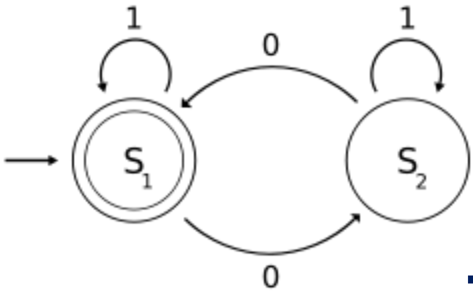
$Q = \{q_1,\dots,q_8, q_{accept}, q_{reject}\}$, q_{reject} *not in the picture for simplicity*

$\Sigma = \{0\}$,

$\Gamma = \{0, x, \sqcup\}$, and

$\delta = Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$





Recognizing $B = \{w\#w \mid w \in \{0,1\}^*\}$

At a higher level, we can also *describe* M, slightly differently, by *specifying its implementation*.

M_1 = "On input string w :

1. Check if the string is in the correct format (i.e.contains #). If not, *reject*.

2. Start on the left side of the #, at the first non-crossed symbol. Cross off a symbol.

3.Scan right, past the #, to the first non-crossed symbol. If it matches the one crossed off in step 2, cross this one off. Otherwise, *reject*.

4. Go back to the left-hand side of the tape. Go to step 2.

5. When all symbols to the left of the # have been crossed off, check for remaining symbols on the right of the #. If any remain, *reject*. Otherwise, *accept*."