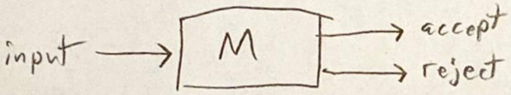
Deterministic Finite Automata

DFAs are computational machines which

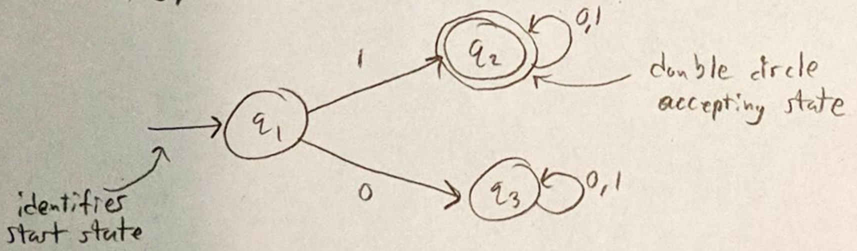
- have finite memory,
- compute in "real-time" in that they fully process an input upon recieving it,
- and accept/reject thier input string.

Visually, we can draw a DFA M as



A DFA <u>accepts</u> a string ω if after processing all of its input it is in an <u>accepting state</u>. It rejects ω if it is in any other state.

Visually, we can draw a DFA M



$\Sigma = \mathbb{Z}_2$

What does this DFA do?

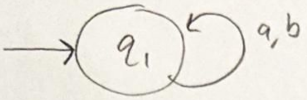
$L(M) = \{1\omega \mid \omega \in \Sigma^*\} = \{\omega \mid \omega \in \Sigma^*$ and starts with a $1\}$

Give a DFA for the following languages with $\Sigma = \{a, b\}$.

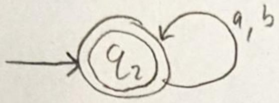
- $\emptyset$, Σ
- $\{xaybz \mid x, y, z \in \Sigma^*\}$
- $\{xabbay \mid x, y \in \Sigma^*\}$
- $\{\omega \in \Sigma^* \mid \omega$ has equal a's and b's$\}$

↖ not regular!

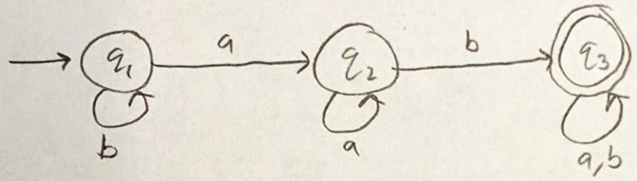
1)



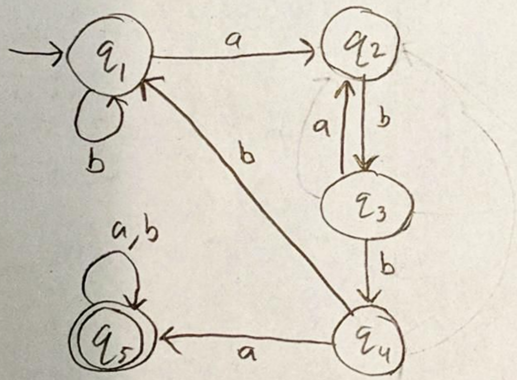
2)



3)



4)



5) No DFA can decide this.

To know $\#(a) = \#(b)$, you must track their difference. Since $\#(a) - \#(b)$ is unbounded, this cannot be done with finite memory.

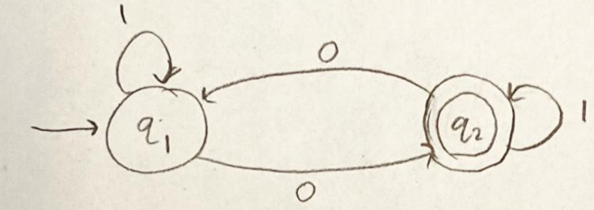
Formal DFA

A **deterministic finite automaton** M is a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F).$$

$$\underset{\text{states}}{Q} \quad \underset{\text{alphabet}}{\Sigma} \quad \underset{\text{transition function}}{\delta} \quad \underset{\text{start state}}{q_0} \quad \underset{\text{accepting states}}{F}$$

Consider the DFA



what does it do?

$Q = \{q_1, q_2\}$

$\Sigma = \mathbb{Z}_2$

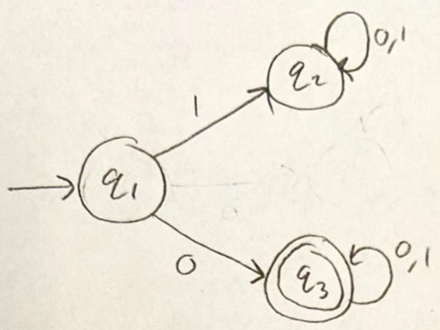
$\delta =$

	q_1	q_2
0	q_2	q_1
1	q_1	q_2

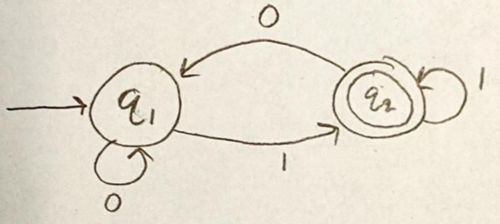
$q_0 = q_1$

$F = \{q_1\}$

Def A language A is regular if there is some DFA M with $L(M) = A$.



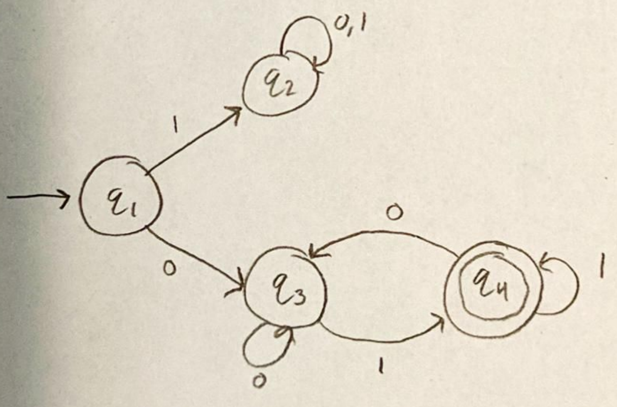
$A = \{0w \mid w \in \Sigma^*\}$



$B = \{w1 \mid w \in \Sigma^*\}$

What is the DFA for

$A \cap B = \{0w1 \mid w \in \Sigma^*\}$?


q1
q2
q3
q4
0,1
1
0
0
1
1
0