Finite Automata

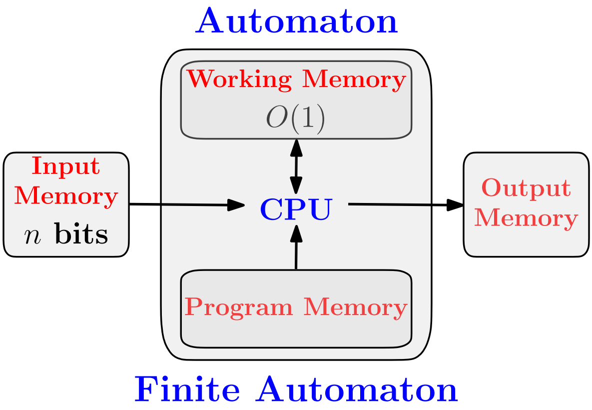
- Deterministic Finite Automata
- Languages decided by a DFA – Regular Languages
- Closure Properties of regular languages
- Non-Deterministic Finite Automata, DFA= NFA
- Regular Expression: Computation as Description
- DFA=NFA=RegExp, Generalized NFA
- Non-Regular Languages, The Pumping Lemma
- Minimizing DFA

IMDAD ULLAH KHAN

Deterministic Finite Automata

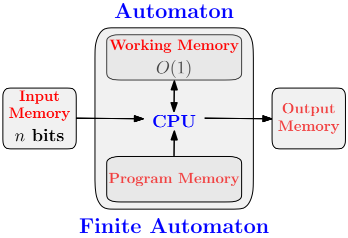
Automata are distinguished by type/amount of working memory

A Deterministic Finite Automata has constant working memory



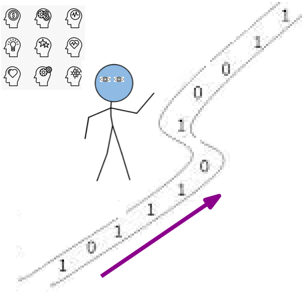
Anatomy of DFA

A Deterministic Finite Automata has constant working memory



A deterministic finite automaton or a finite state machine is a little creature

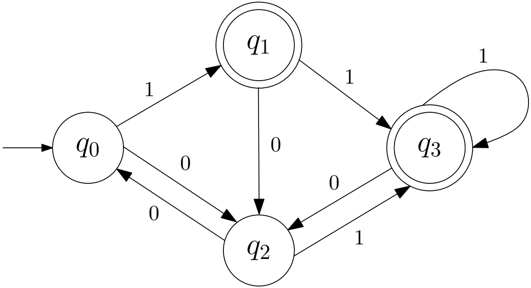
- it has tiny eyes – sees one symbol
- changes its state of mind according to the symbol it sees
- only remember its current state of mind



Anatomy of DFA

A DFA over alphabet Σ is depicted as a directed graph with self-loops

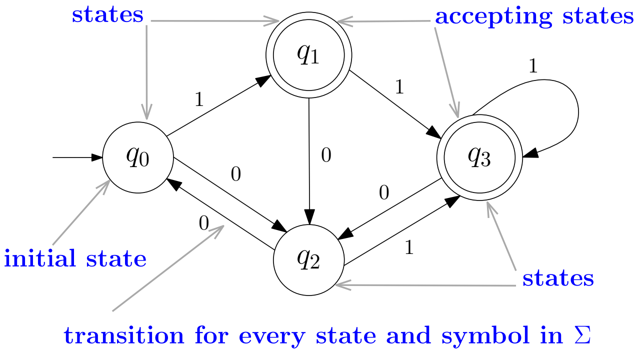
▷ called state diagram of the DFA



Anatomy of DFA

A DFA over alphabet Σ is depicted as a directed graph with self-loops

▷ called state diagram of the DFA

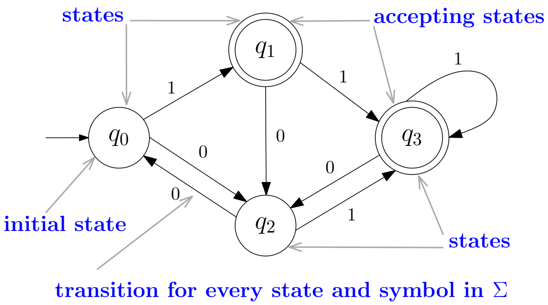


Simulation of DFA

DFA begins in the (unique) initial state and read the input left-to-right one character at a time

It transition to the next state according to transition rules (labeled edges)

The automaton **accepts** the input string if the last state is an accepting state (double-circled). Else, it **rejects** the input string.

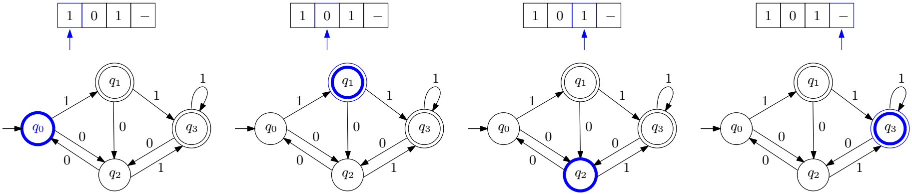


Simulation of DFA

DFA begins in the (unique) initial state and read the input left-to-right one character at a time

It transition to the next state according to transition rules (labeled edges)

The automaton **accepts** the input string if the last state is an accepting state (double-circled). Else, it **rejects** the input string.



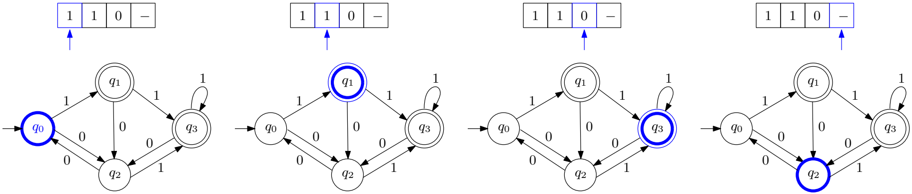
This DFA accepts the input string 101

Simulation of DFA

DFA begins in the (unique) initial state and read the input left-to-right one character at a time

It transition to the next state according to transition rules (labeled edges)

The automaton **accepts** the input string if the last state is an accepting state (double-circled). Else, it **rejects** the input string.



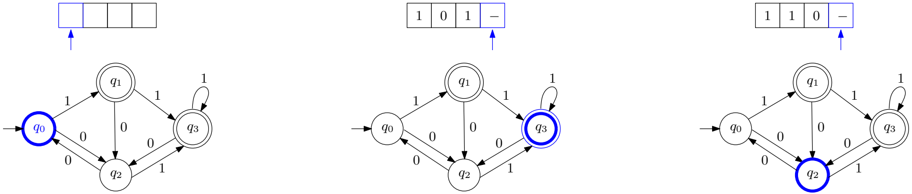
This DFA rejects the input string 110

Simulation of DFA

DFA begins in the (unique) initial state and read the input left-to-right one character at a time

It transition to the next state according to transition rules (labeled edges)

The automaton **accepts** the input string if the last state is an accepting state (double-circled). Else, it **rejects** the input string.



Which strings are accepted by this DFA?

DFA: Formal Definition

A DFA M is a 5-tuple $M = (Q, \Sigma, q_0, \delta, F)$

- Q: A finite set of states
- Σ: Alphabet ▷ A finite set of characters
- $q_0 \in Q$ Start or initial state
- $\delta : Q \times \Sigma \mapsto Q$ Transition function
- $F \subseteq Q$ Set of accept/final states

M accepts $w = w_1, w_2, \ldots, w_n \in \Sigma^n$

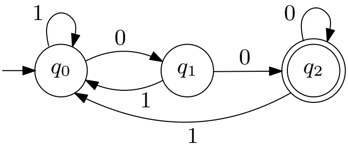
if there is a sequence of states $r_0, r_1, \ldots, r_n$ such that

$r_0 = q_0, \quad \delta(r_i, w_{i+1}) = r_{i+1}, \quad r_n \in F$

DFA: Formal Definition

A DFA M is a 5-tuple $M = (Q, \Sigma, q_0, \delta, F)$

- Q: A finite set of states
- Σ: Alphabet ▷ A finite set of characters
- $q_0 \in Q$ Start or initial state
- $\delta : Q \times \Sigma \mapsto Q$ Transition function
- $F \subseteq Q$ Set of accept/final states



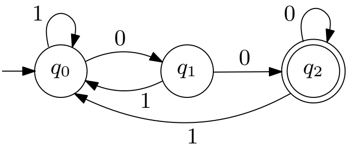
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- Initial State: q_0
  - $\delta(q_0, 0) = q_1, \quad \delta(q_0, 1) = q_0$
  - $\delta(q_1, 0) = q_2, \quad \delta(q_1, 1) = q_0$
  - $\delta(q_2, 0) = q_2, \quad \delta(q_2, 1) = q_0$
- $F = \{q_2\}$

DFA: Formal Definition

A DFA M is a 5-tuple $M = (Q, \Sigma, q_0, \delta, F)$

- Q: A finite set of states
- Σ: Alphabet ▷ A finite set of characters
- $q_0 \in Q$ Start or initial state
- $\delta : Q \times \Sigma \mapsto Q$ Transition function
- $F \subseteq Q$ Set of accept/final states

Transition function δ can be depicted in a transition table (lookup table)



	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

DFA as Programming Code

A DFA M is a 5-tuple $M = (Q, \Sigma, q_0, \delta, F)$

- Q: A finite set of states
- Σ: Alphabet ▷ A finite set of characters
- $q_0 \in Q$ Start or initial state
- $\delta : Q \times \Sigma \mapsto Q$ Transition function
- $F \subseteq Q$ Set of accept/final states

Algorithm DFA as Programming Code

```
STATE ← q_0
i ← 0
while input[i] ≠ EOF do
    STATE ← δ(STATE, input[i])
    i ← i + 1
if STATE ∈ F then
    return Accept
else
    return Reject
```