Finite Automata

Question: What is a computer?

- real computers too complex for any theory

Finite Automata

Question: What is a computer?

- real computers too complex for any theory
- need manageable mathematical abstraction

Finite Automata

Question: What is a computer?

- real computers too complex for any theory
- need manageable mathematical abstraction
- idealized models: accurate in some ways, but not in all details

Finite Automata – Important ideas

- formal definition of finite automata

Finite Automata – Important ideas

- formal definition of finite automata
- deterministic vs. non-deterministic finite automata

Finite Automata – Important ideas

- formal definition of finite automata
- deterministic vs. non-deterministic finite automata
- regular languages

Finite Automata – Important ideas

- formal definition of finite automata
- deterministic vs. non-deterministic finite automata
- regular languages
- operations on regular languages

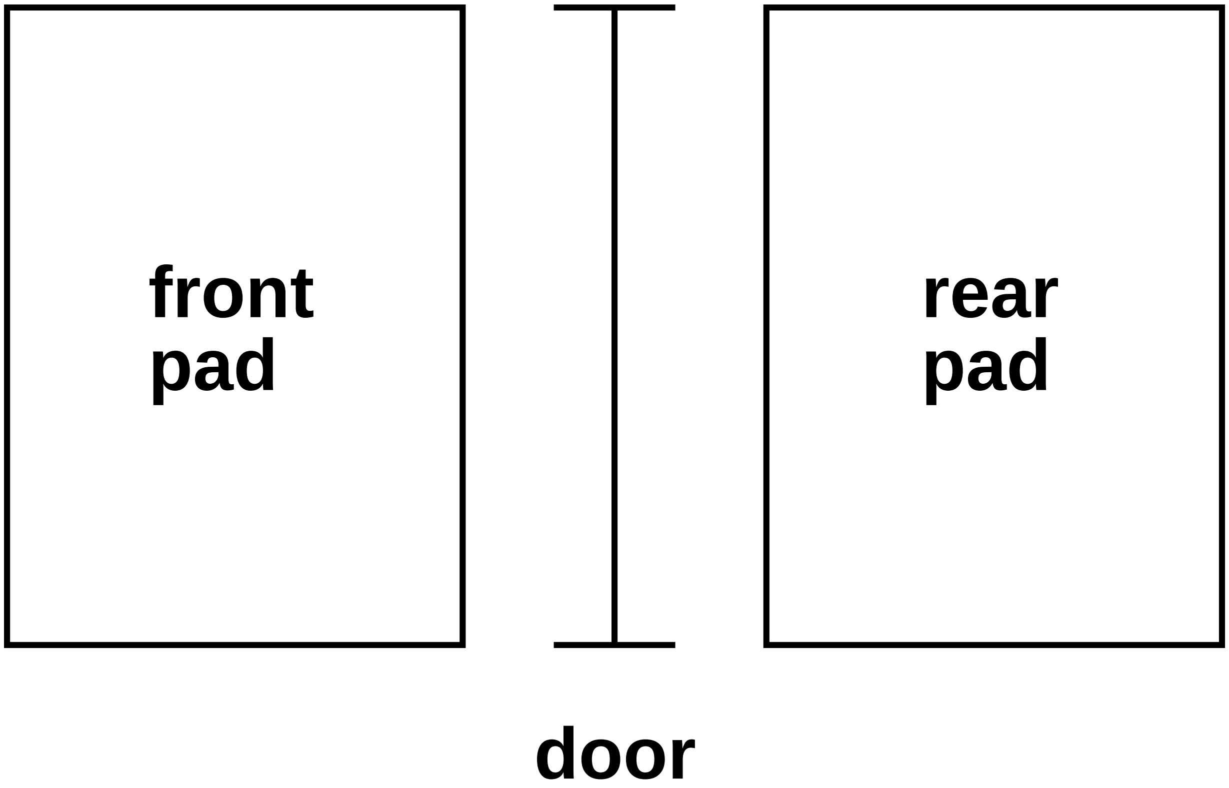
Finite Automata – Important ideas

- formal definition of finite automata
- deterministic vs. non-deterministic finite automata
- regular languages
- operations on regular languages
- regular expressions

Finite Automata – Important ideas

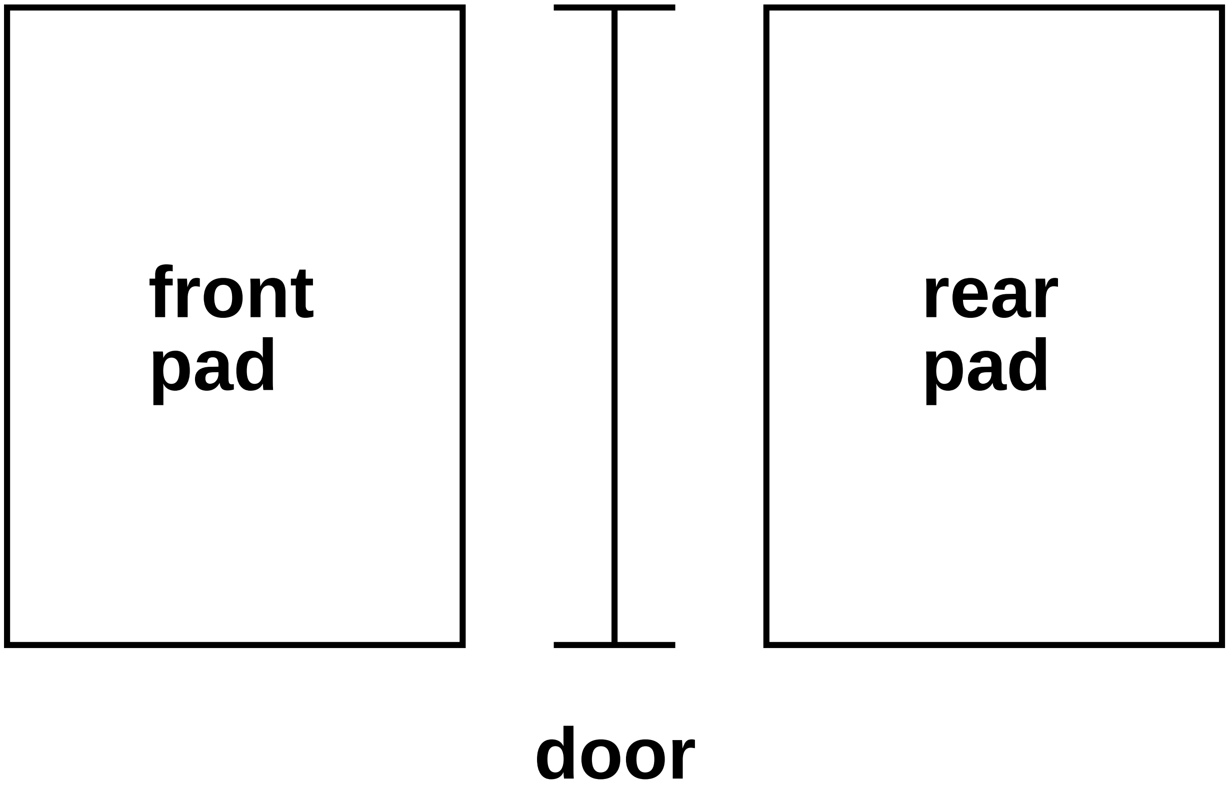
- formal definition of finite automata
- deterministic vs. non-deterministic finite automata
- regular languages
- operations on regular languages
- regular expressions
- pumping lemma

Example: An Automatic Door



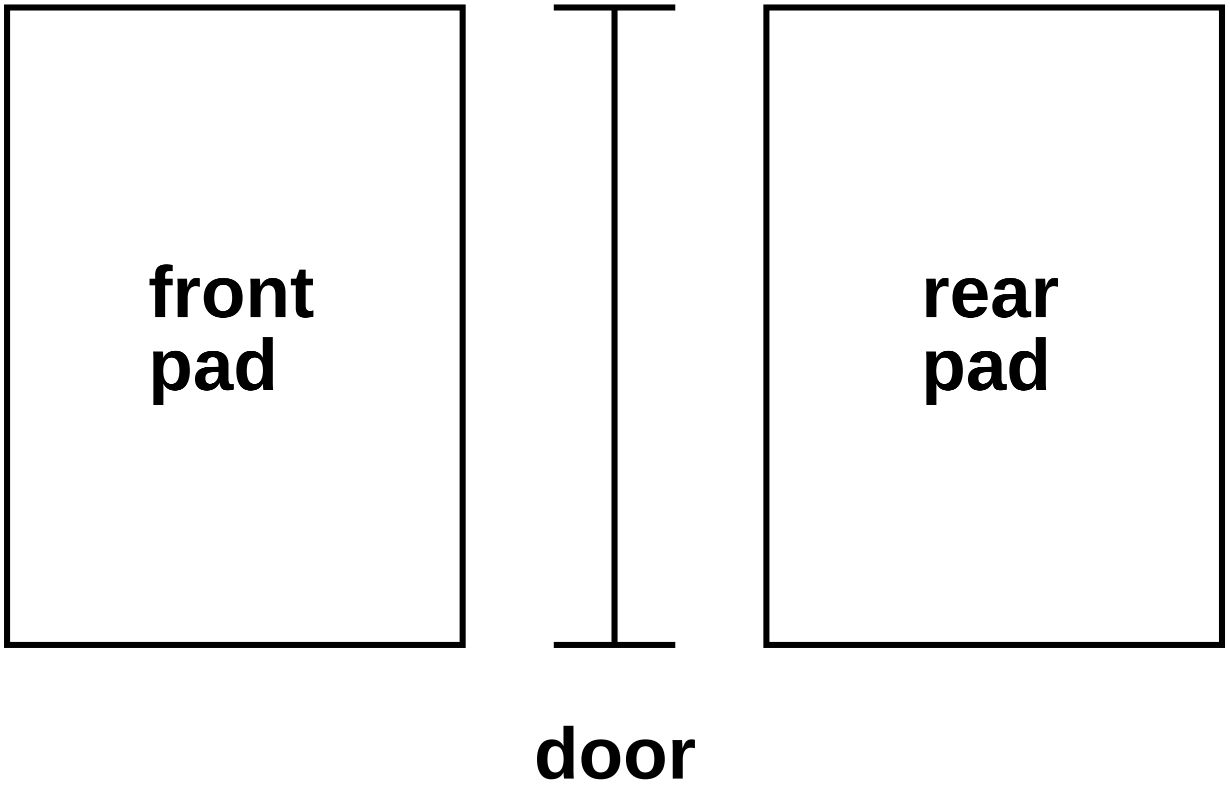
- open when person approaches

Example: An Automatic Door



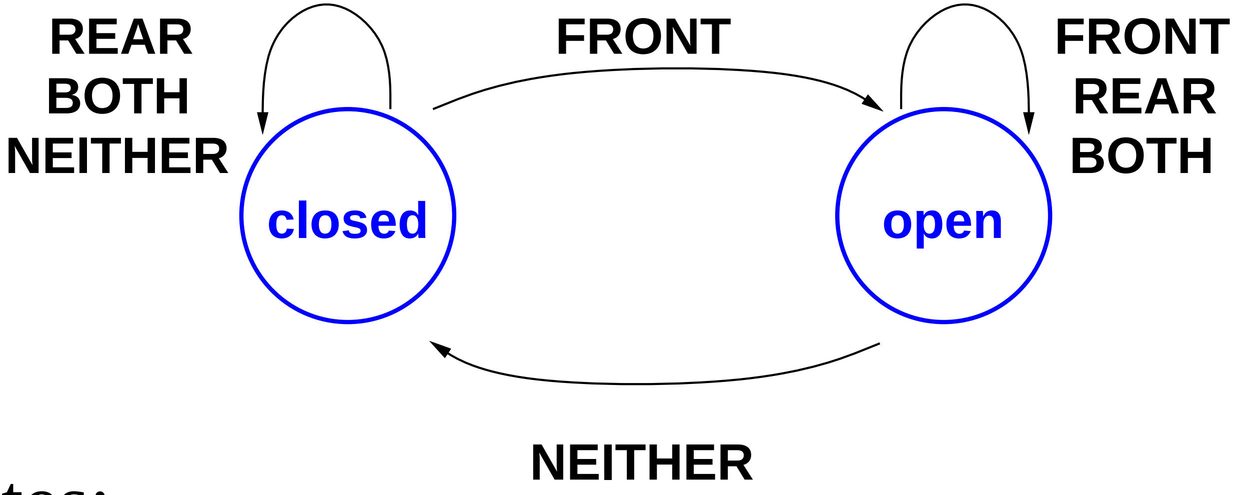
- open when person approaches
- hold open until person clears

Example: An Automatic Door



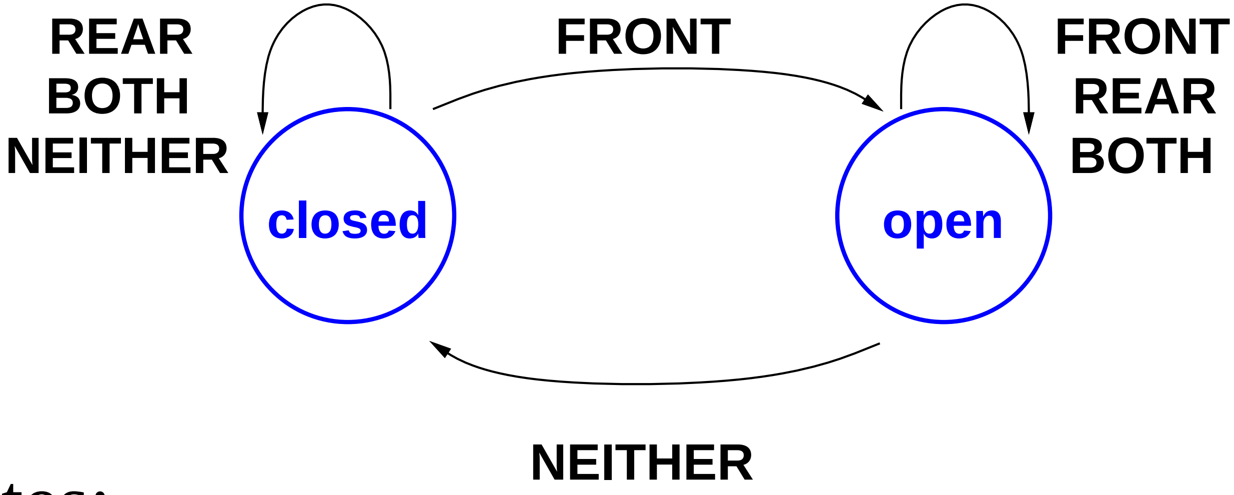
- open when person approaches
- hold open until person clears
- don’t open when someone standing behind door

The Automatic Door as DFA



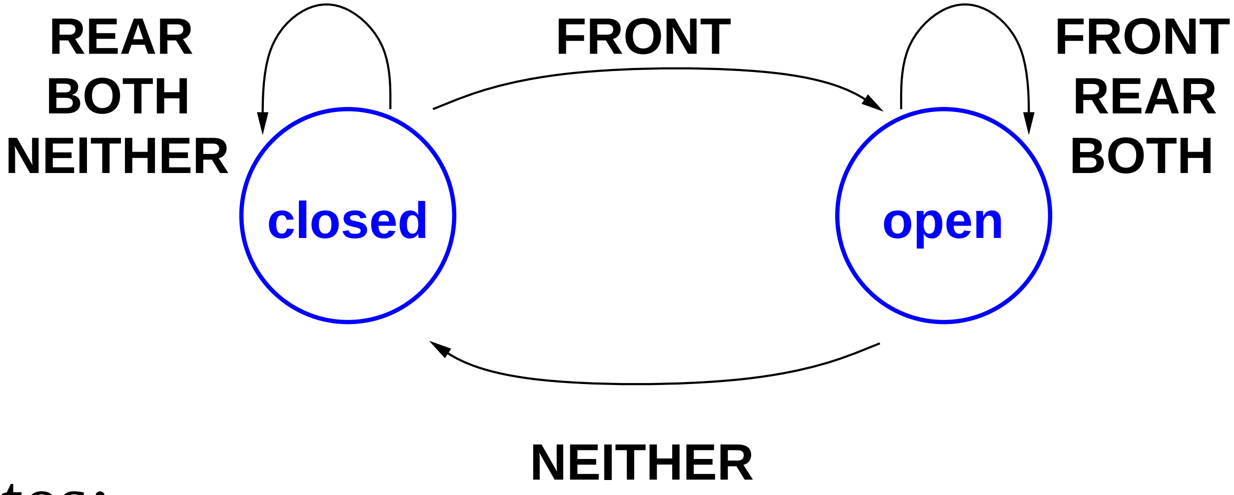
- States:

The Automatic Door as DFA



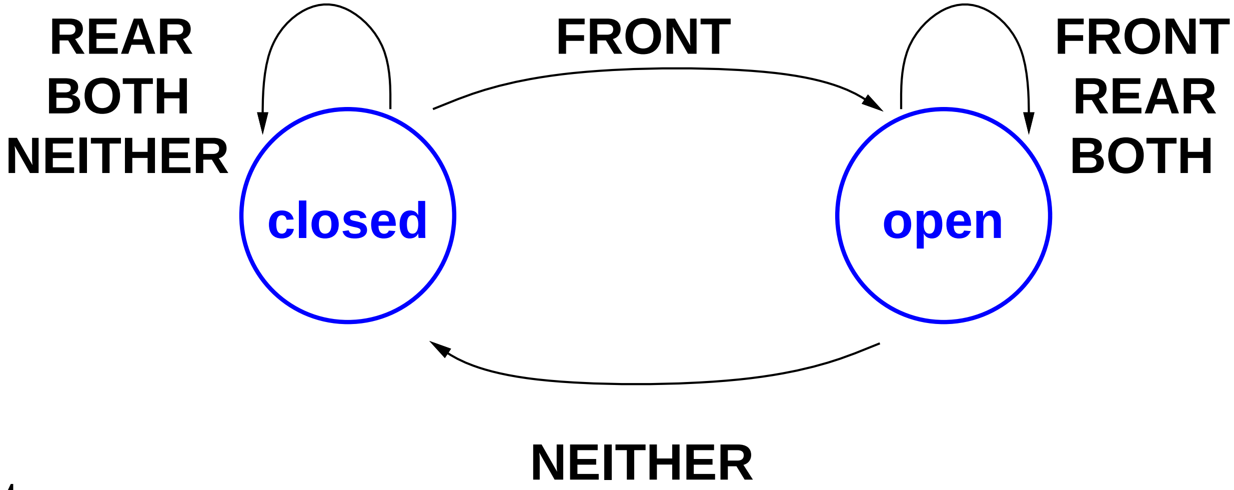
- States:
  - OPEN

The Automatic Door as DFA



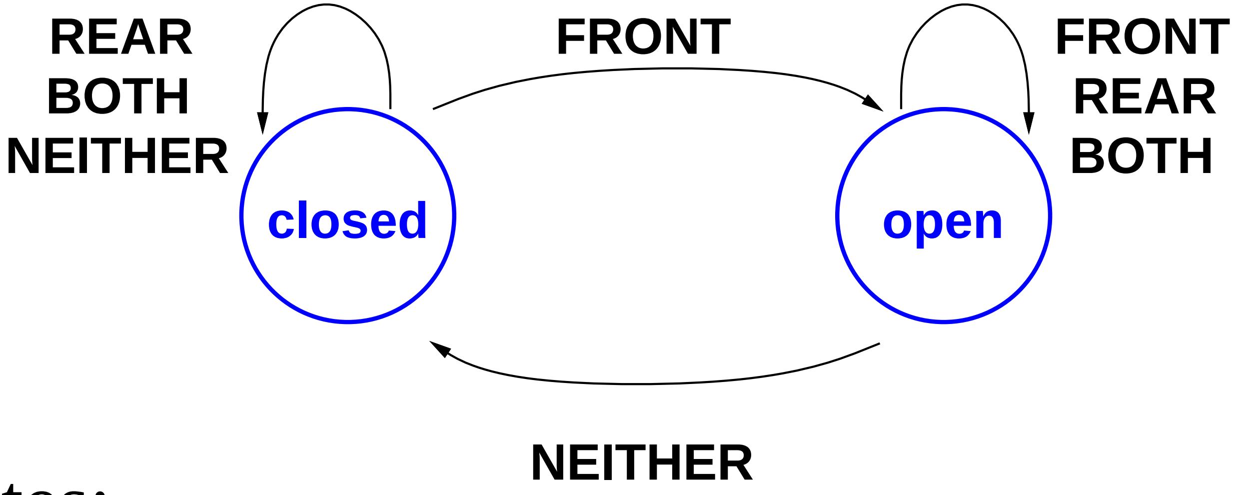
- States:
  - OPEN
  - CLOSED

The Automatic Door as DFA



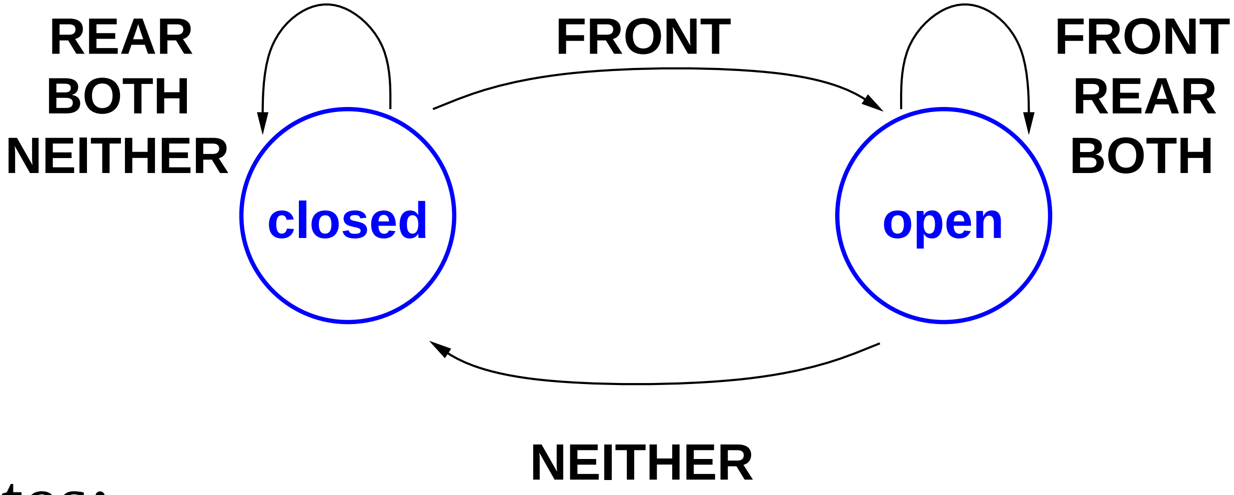
- States:
  - OPEN
  - CLOSED
- Sensor:

The Automatic Door as DFA



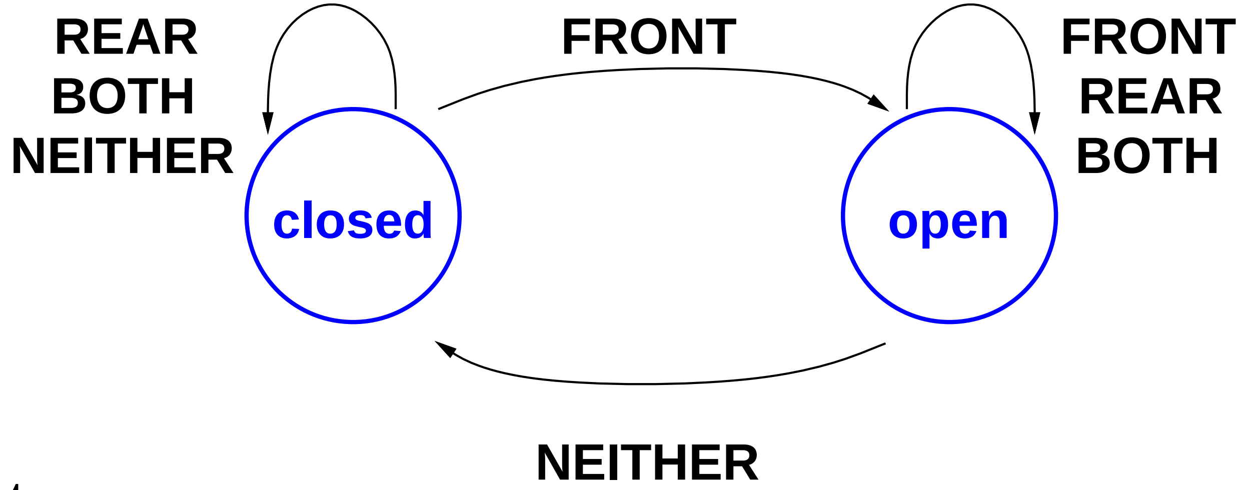
- States:
  - OPEN
  - CLOSED
- Sensor:
  - FRONT: someone on rear pad

The Automatic Door as DFA



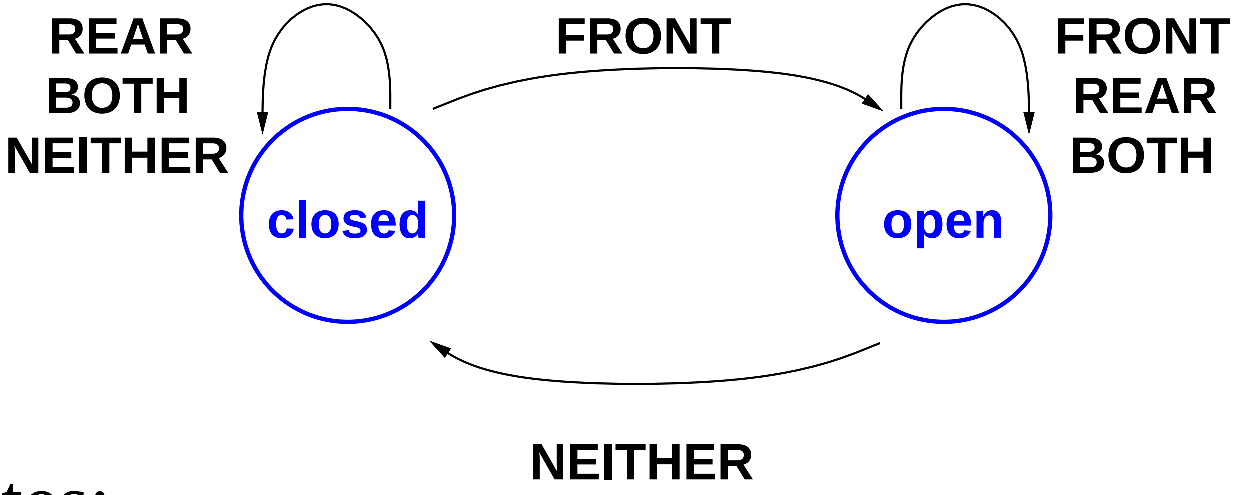
- States:
  - OPEN
  - CLOSED
- Sensor:
  - FRONT: someone on rear pad
  - REAR: someone on rear pad

The Automatic Door as DFA



- States:
  - OPEN
  - CLOSED
- Sensor:
  - FRONT: someone on rear pad
  - REAR: someone on rear pad
  - BOTH: someone on both pads

The Automatic Door as DFA



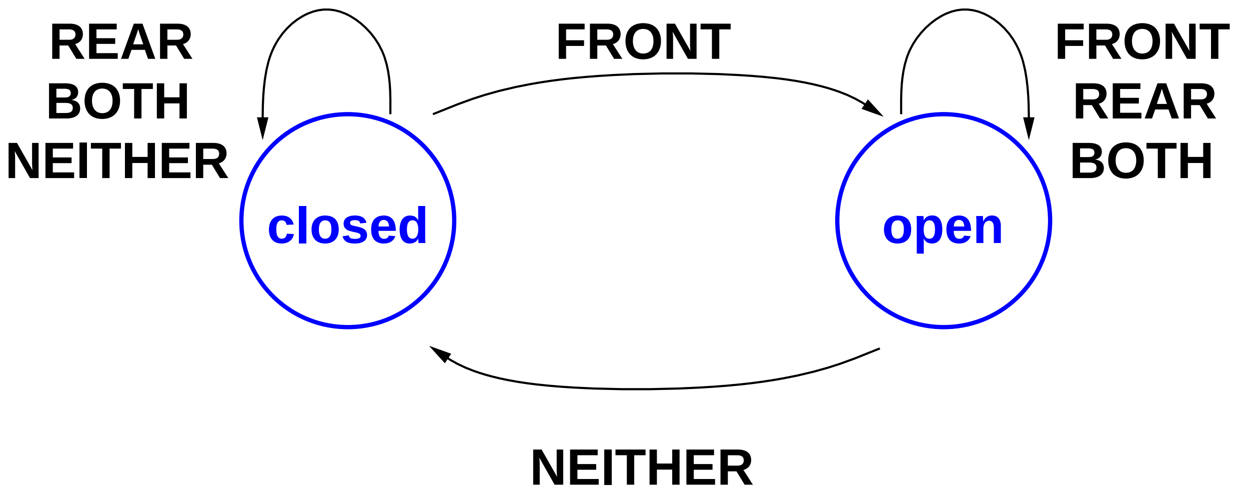
- States:
  - OPEN
  - CLOSED
- Sensor:
  - FRONT: someone on rear pad
  - REAR: someone on rear pad
  - BOTH: someone on both pads
  - NEITHER no one on either pad.

The Automatic Door as DFA

DFA is Deterministic Finite Automata

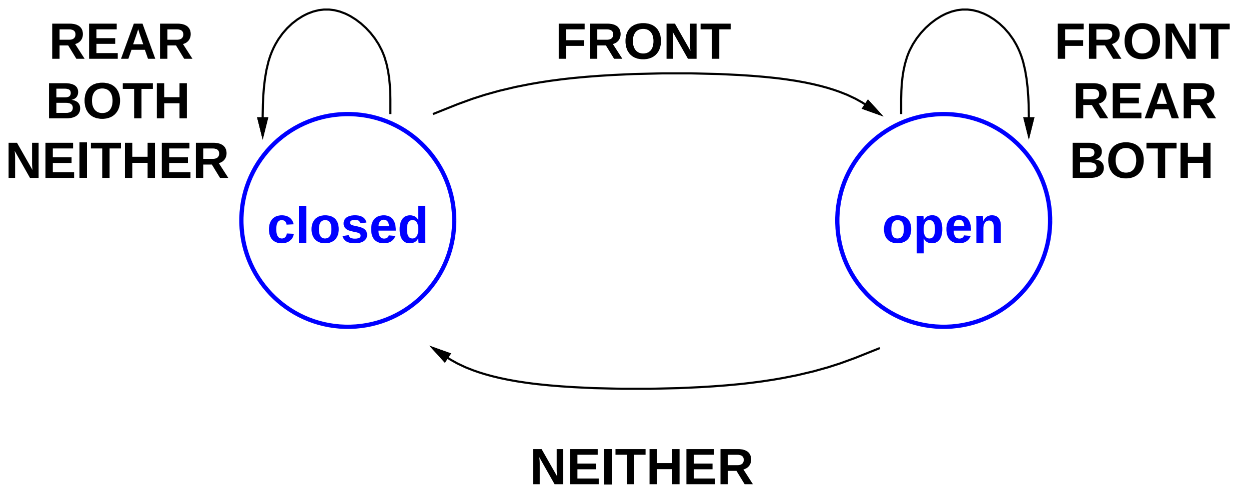
The Automatic Door as DFA

DFA is Deterministic Finite Automata



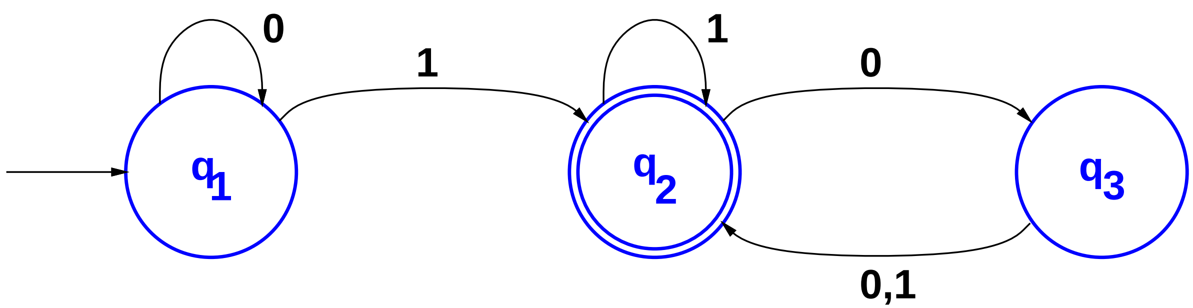
The Automatic Door as DFA

DFA is Deterministic Finite Automata



	neither	front	rear	both
closed	closed	open	closed	closed
open	closed	open	open	open

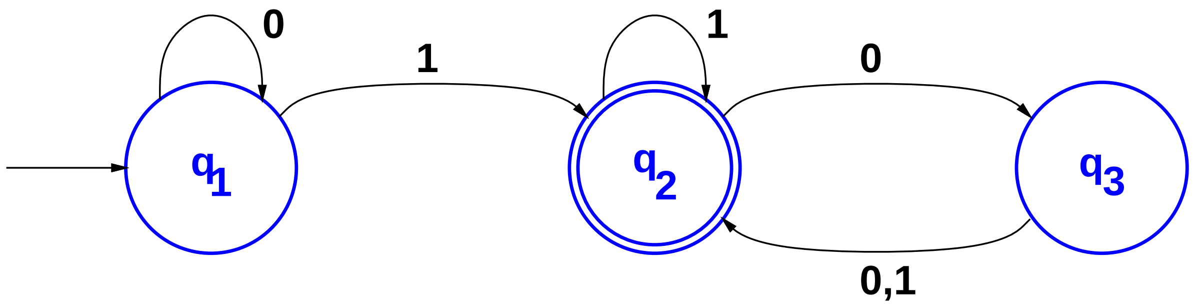
DFA: Informal Definition



The machine M_1:

- states: q_1, q_2, and q_3.

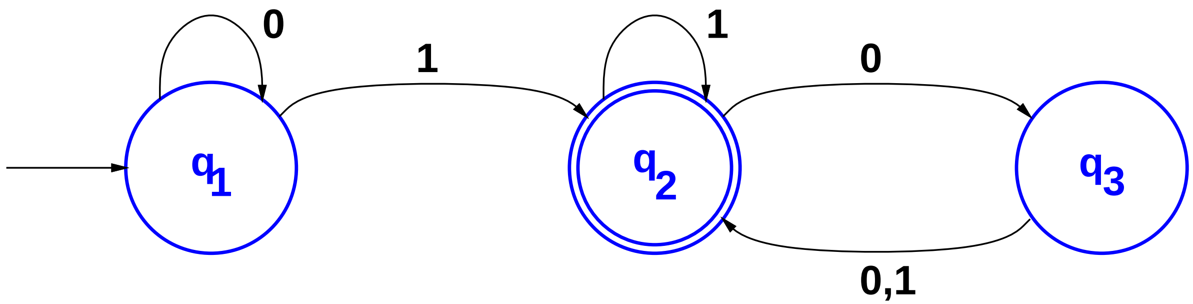
DFA: Informal Definition



The machine M_1:

- states: q_1, q_2, and q_3.
- start state: q_1 (arrow from "outside").

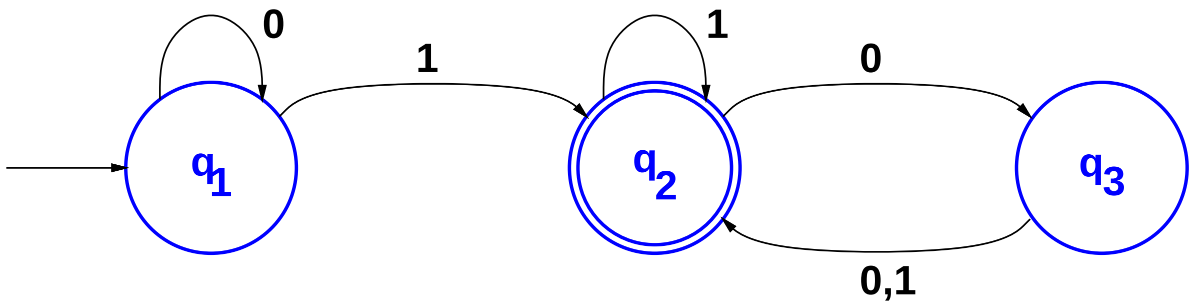
DFA: Informal Definition



The machine M_1:

- states: q_1, q_2, and q_3.
- start state: q_1 (arrow from “outside”).
- accept state: q_2 (double circle).

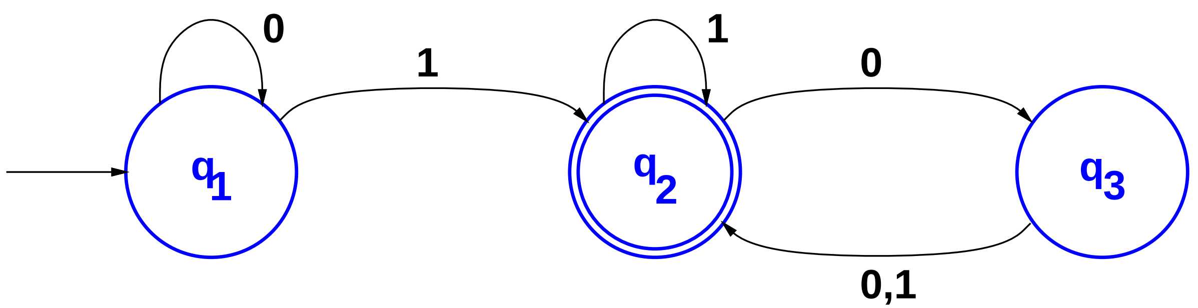
DFA: Informal Definition



The machine M_1:

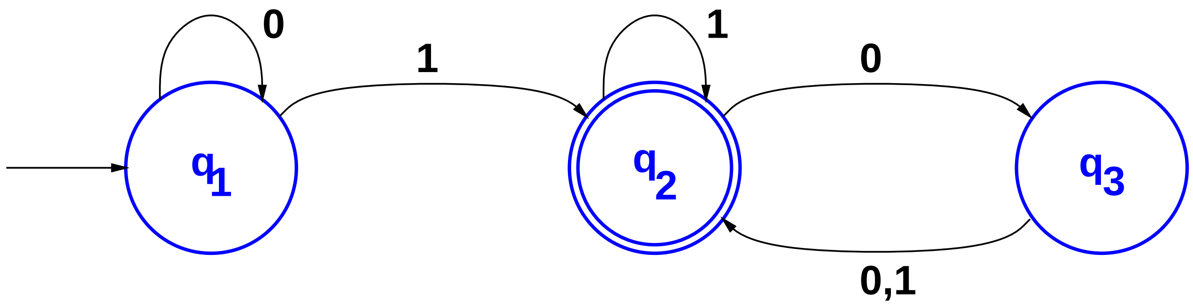
- states: q_1, q_2, and q_3.
- start state: q_1 (arrow from "outside").
- accept state: q_2 (double circle).
- state transitions: arrows.

DFA: Informal Definition (cont.)



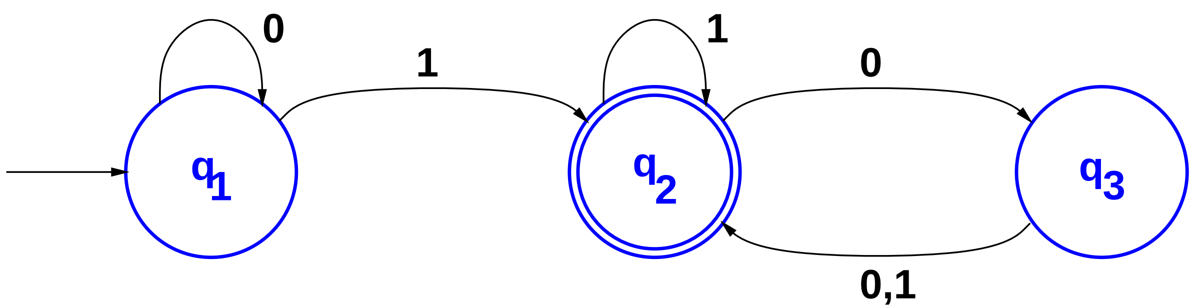
- On an input string

DFA: Informal Definition (cont.)



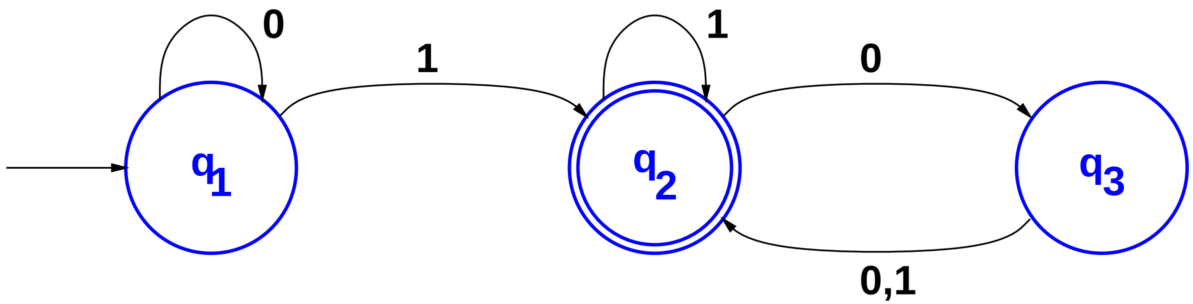
- On an input string
  - DFA begins in start state q_1

DFA: Informal Definition (cont.)



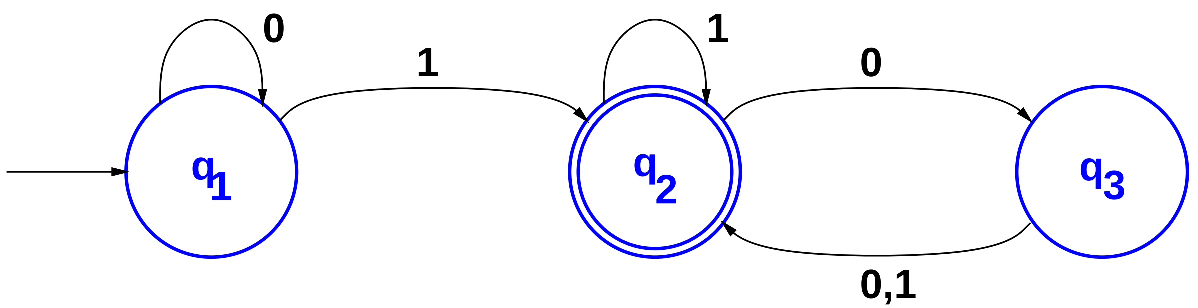
- On an input string
  - DFA begins in start state q_1
  - after reading each symbol, DFA makes state transition with matching label.

DFA: Informal Definition (cont.)



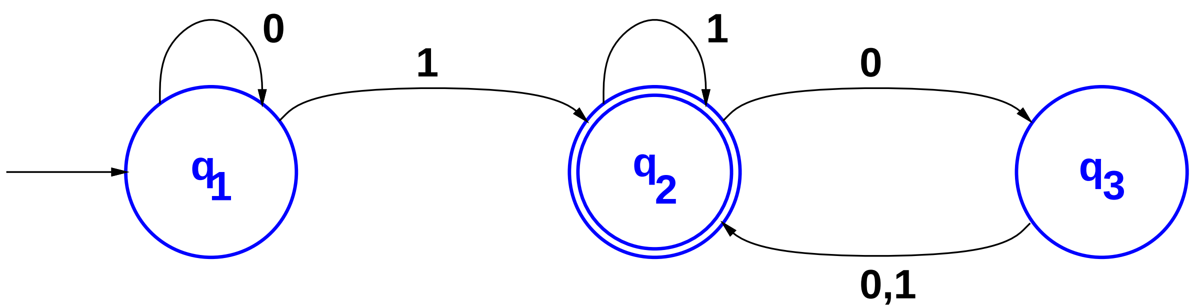
- On an input string
  - DFA begins in start state q_1
  - after reading each symbol, DFA makes state transition with matching label.
- After reading last symbol, DFA produces output:

DFA: Informal Definition (cont.)



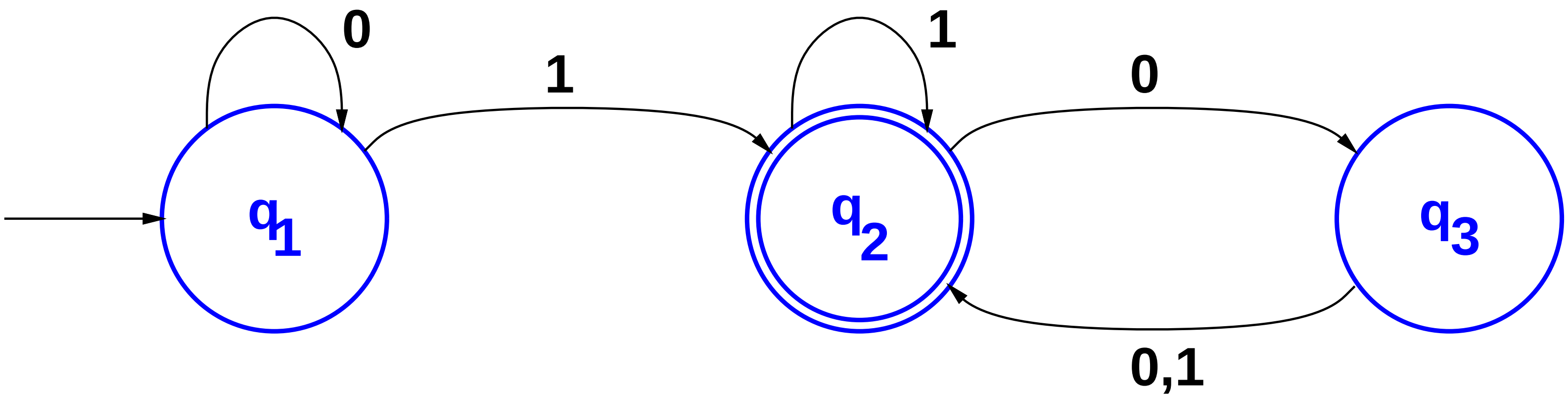
- On an input string
  - DFA begins in start state q_1
  - after reading each symbol, DFA makes state transition with matching label.
- After reading last symbol, DFA produces output:
  - accept if DFA is an accepting state.

DFA: Informal Definition (cont.)



- On an input string
  - DFA begins in start state q_1
  - after reading each symbol, DFA makes state transition with matching label.
- After reading last symbol, DFA produces output:
  - accept if DFA is an accepting state.
  - reject otherwise.

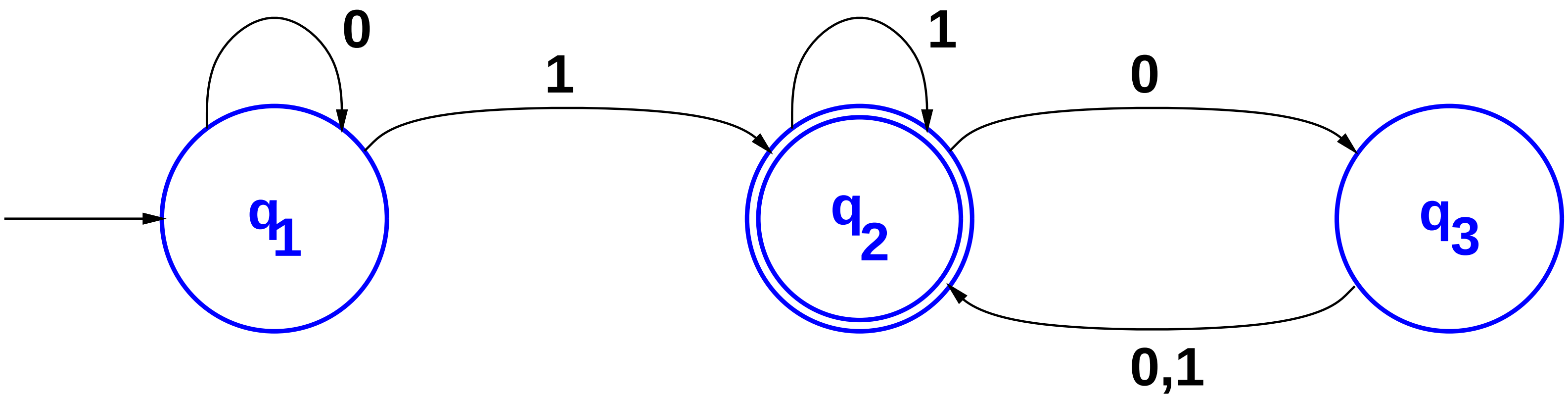
Informal Definition - Example



What happens on input strings

- 1101

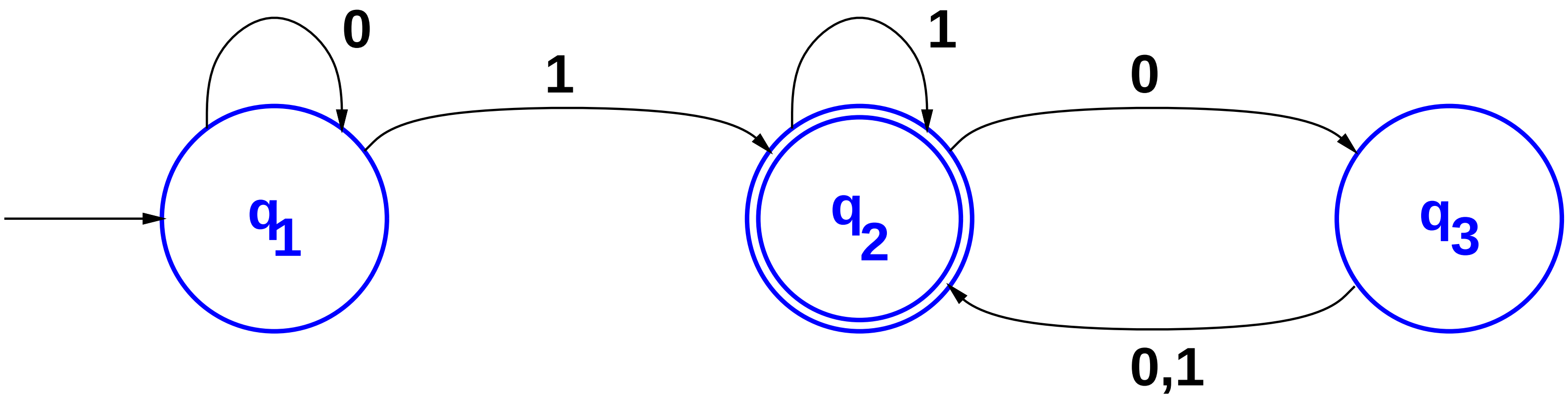
Informal Definition - Example



What happens on input strings

- 1101
- 0010

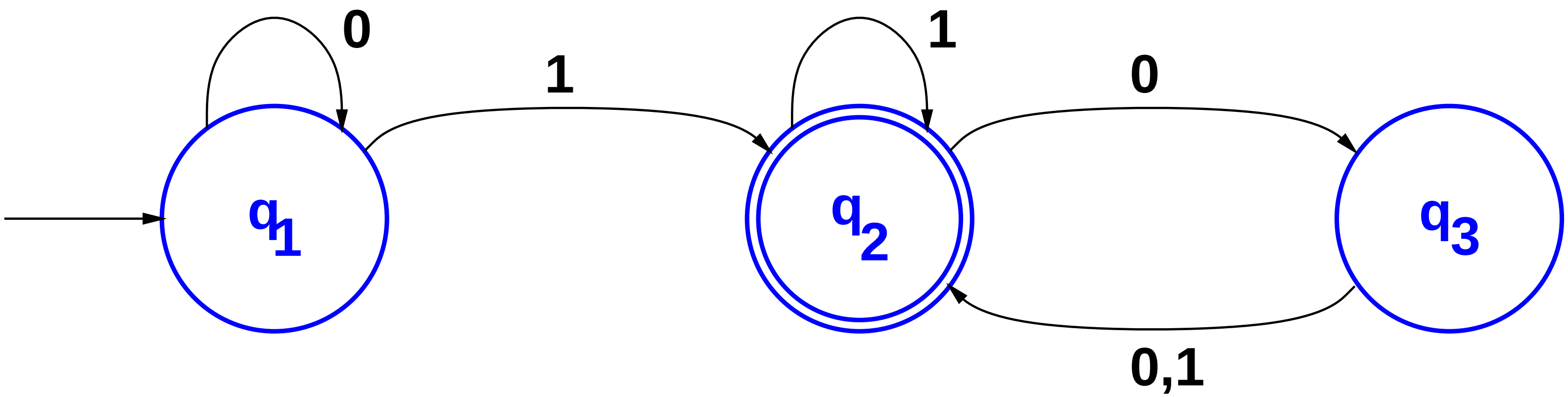
Informal Definition - Example



What happens on input strings

- 1101
- 0010
- 01100

Informal Definition



This DFA accepts

- all input strings that end with a 1
- all input strings that contain at least one 1, and end with an even number of 0's
- no other strings

Languages and Alphabets

An alphabet Σ is a finite set of letters.

- $\Sigma = \{a, b, c, \ldots, z\}$ – the English alphabet.
- $\Sigma = \{\alpha, \beta, \gamma, \ldots, \zeta\}$ – the Greek alphabet.
- $\Sigma = \{0, 1\}$ – the binary alphabet.
- $\Sigma = \{0, 1, \ldots, 9\}$ – the digital alphabet.

Languages and Alphabets

An alphabet Σ is a finite set of letters.

- $\Sigma = \{a, b, c, \ldots, z\}$ – the English alphabet.
- $\Sigma = \{\alpha, \beta, \gamma, \ldots, \zeta\}$ – the Greek alphabet.
- $\Sigma = \{0, 1\}$ – the binary alphabet.
- $\Sigma = \{0, 1, \ldots, 9\}$ – the digital alphabet.

The collection of all strings over Σ is denoted by Σ^*.

Languages and Alphabets

An alphabet Σ is a finite set of letters.

- $\Sigma = \{a, b, c, \ldots, z\}$ – the English alphabet.
- $\Sigma = \{\alpha, \beta, \gamma, \ldots, \zeta\}$ – the Greek alphabet.
- $\Sigma = \{0, 1\}$ – the binary alphabet.
- $\Sigma = \{0, 1, \ldots, 9\}$ – the digital alphabet.

The collection of all strings over Σ is denoted by Σ^*.

For the binary alphabet, ε, 1, 0, 000000000, 1111111000 are all members of Σ^*.

Languages and Examples

A language over Σ is a subset $L \subseteq \Sigma^*$. For example

- Modern English.

Languages and Examples

A language over Σ is a subset $L \subseteq \Sigma^*$. For example

- Modern English.
- Ancient Greek.

Languages and Examples

A language over Σ is a subset $L \subseteq \Sigma^*$. For example

- Modern English.
- Ancient Greek.
- All prime numbers, writen using digits.

Languages and Examples

A language over Σ is a subset $L \subseteq \Sigma^*$. For example

- Modern English.
- Ancient Greek.
- All prime numbers, writen using digits.
- $A = \{w | w \text{ has at most seventeen 0's}\}$

Languages and Examples

A language over Σ is a subset $L \subseteq \Sigma^*$. For example

- Modern English.
- Ancient Greek.
- All prime numbers, writen using digits.
- $A = \{w | w \text{ has at most seventeen 0's}\}$
- $B = \{0^n 1^n | n \geq 0\}$

Languages and Examples

A language over Σ is a subset $L \subseteq \Sigma^*$. For example

- Modern English.
- Ancient Greek.
- All prime numbers, writen using digits.
- $A = \{w | w \text{ has at most seventeen 0's}\}$
- $B = \{0^n 1^n | n \geq 0\}$
- $C = \{w | w \text{ has an equal number of 0's and 1's}\}$

Languages and DFA

Definition: $L(M)$, the language of a DFA M, is the set of strings L that M accepts, $L(M) = L$.

Languages and DFA

Definition: $L(M)$, the language of a DFA M, is the set of strings L that M accepts, $L(M) = L$.

Note that

- M may accept many strings, but
- M accepts only one language

Languages and DFA

Definition: $L(M)$, the language of a DFA M, is the set of strings L that M accepts, $L(M) = L$.

Note that

- M may accept many strings, but
- M accepts only one language

What language does M accept if it accepts no strings?

Languages and DFA

Definition: $L(M)$, the language of a DFA M, is the set of strings L that M accepts, $L(M) = L$.

Note that

- M may accept many strings, but
- M accepts only one language

What language does M accept if it accepts no strings?

A language is called regular if some deterministic finite automaton accepts it.

Formal Definitions

A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set called the states,
- Σ is a finite set called the alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accept states.