Part One:
Automata and Languages

Regular Languages

Chapter 7
Chapter 5
Chapter 4
Chapter 3
Chapter 2
Chapter 1

Recall

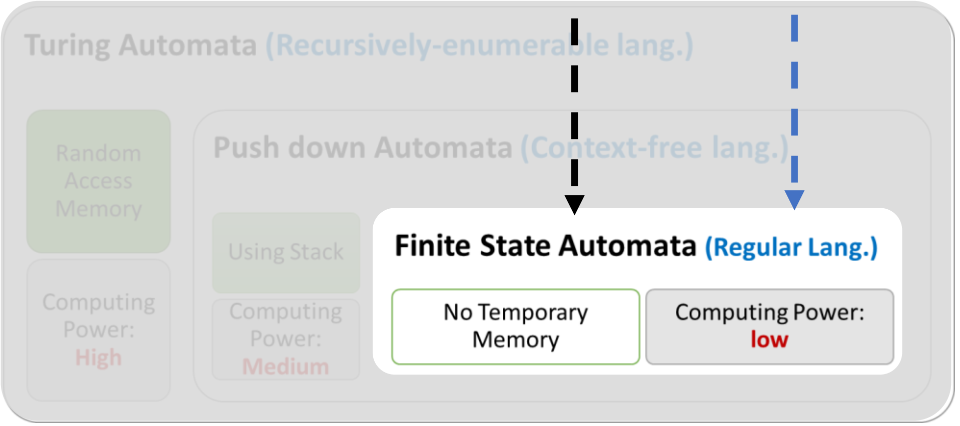
- An alphabet Σ is a finite, non-empty set of abstract symbols.
  - **Example**: $\Sigma = \{0,1\}$
- A string over an alphabet is a finite sequence of symbols from that alphabet
  - **Example**: If $\Sigma = \{0,1\} \rightarrow 01001$ is a string over Σ.
- Let Σ be an alphabet. A language over Σ is a subset, **L**, of Σ^*.
  - **Example**: $\mathrm{L} = \{\mathrm{a, bb, aba}\}$ is a language over $\{\mathrm{a, b}\}^*$.

Introduction

- The goal of the computation theory is to determine the power and limits of computation.

- It is necessary to define precisely
  - what constitutes a model of computation
  - as well as what constitutes a computational problem.

- This is the purpose of automata theory.

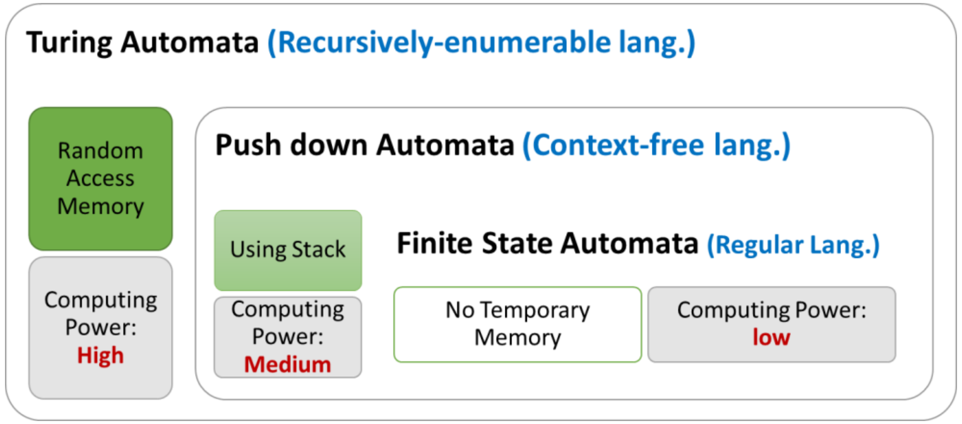


Introduction (cont.)

The Church-Turing Thesis (an open conjecture)

- conjectures that no model of computation that is physically realizable is more powerful than the Turing Machine.

- In other words, the Church-Turing thesis conjectures that **any problem that can be solved via computational means**, **can be solved by a Turing Machine**.



Chapter 1 - Outline

1.1Finite Automata
- Formal definition of a finite automaton
- Examples of finite automata
- Formal definition of computation
- Designing finite automata
- The regular operations

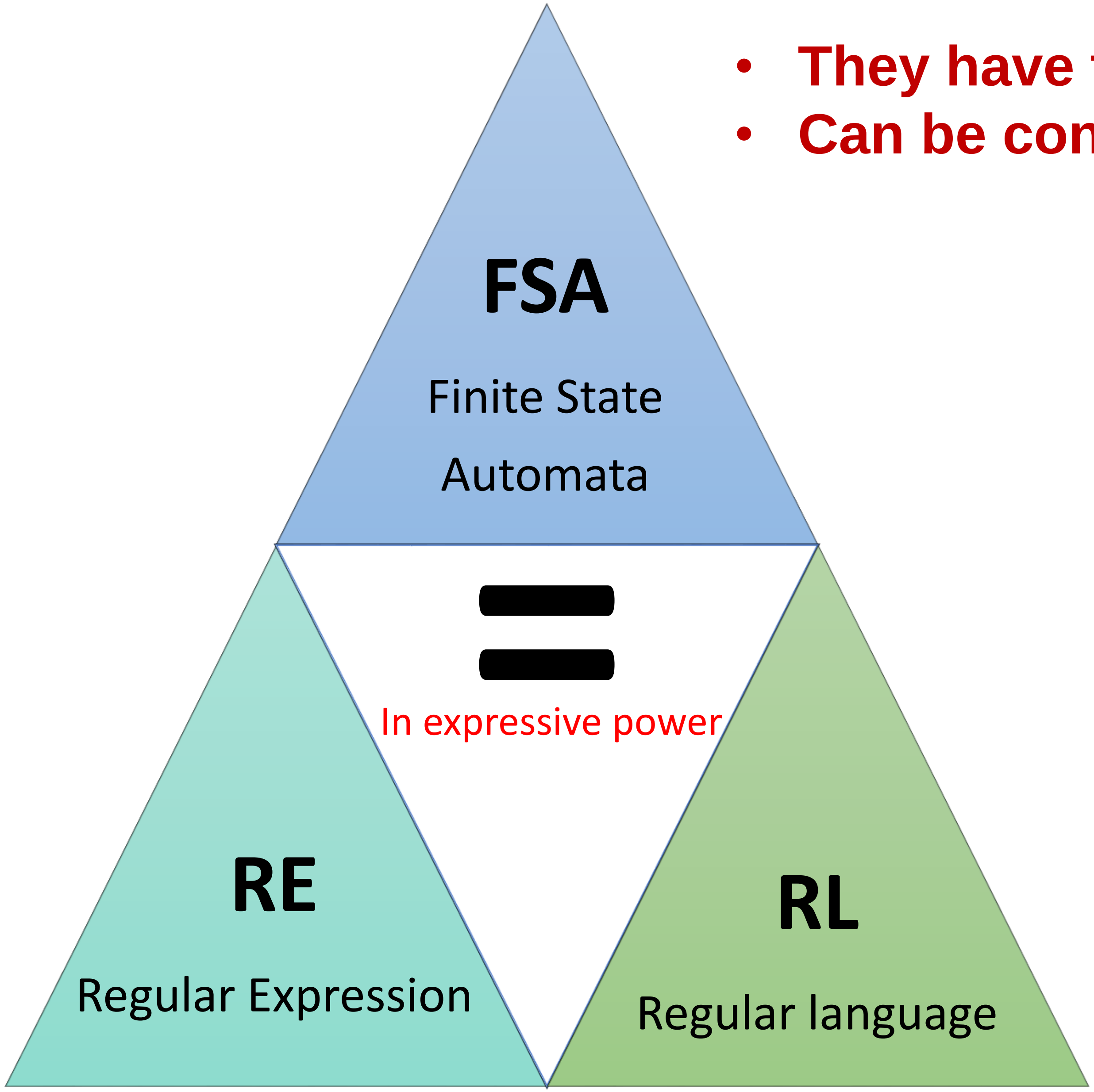
1.2 Nondeterminism
- Formal definition of a nondeterministic finite automaton
- Equivalence of NFAs and DFAs
- Closure under the regular operations

1.3 Regular Expressions
- Formal definition of a regular expression
- Equivalence with finite automata

1.4 Nonregular Languages
- The pumping lemma for regular languages

Automata Theory

- **They have the same power**
- **Can be converted to each other**



Finite Automata (Finite State Machin – F.S.M)

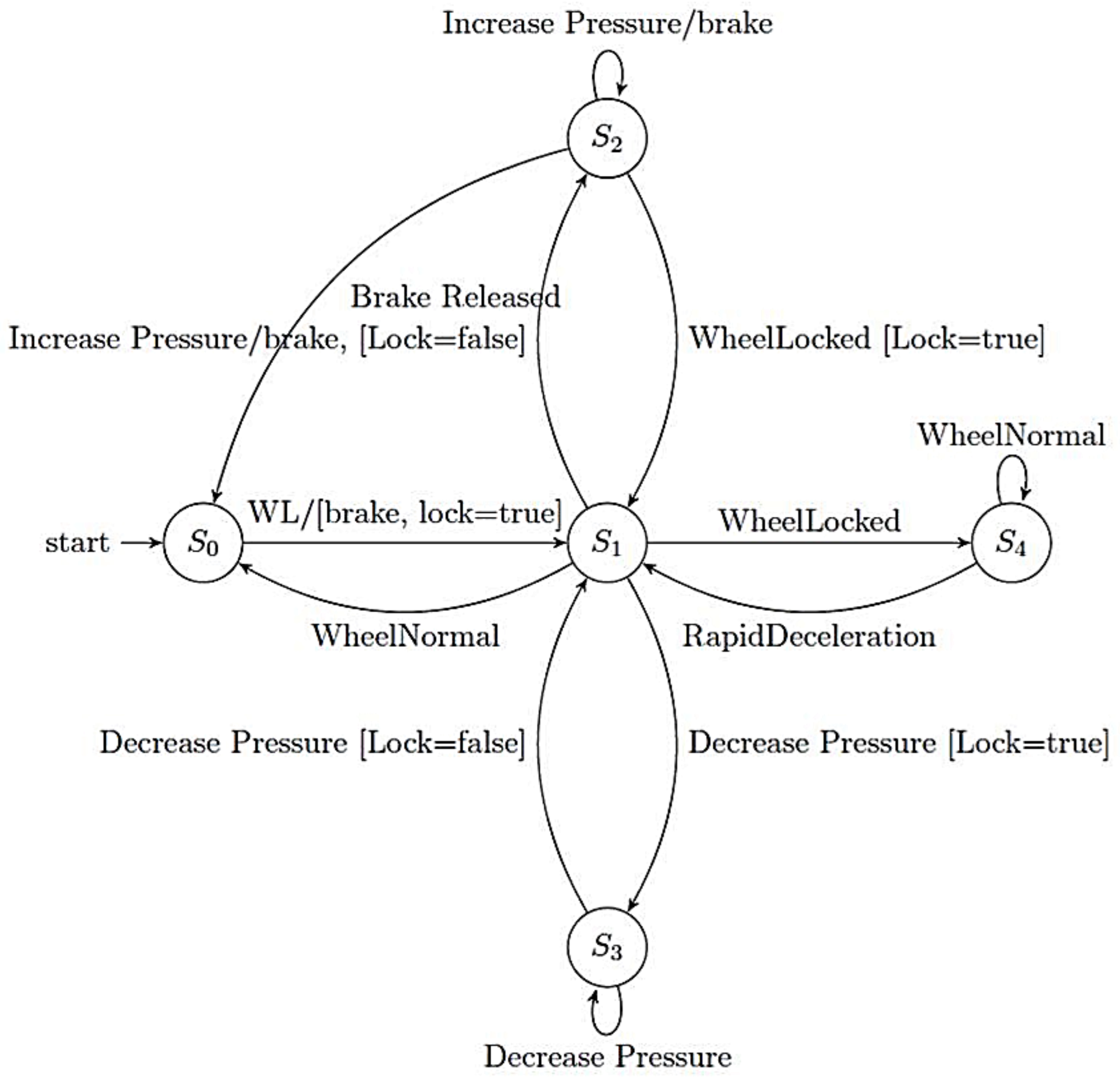
- The **simplest** model of computation.
- with an extremely **limited** amount of **memory**.
  - **Finite** and usually quite small
  - but enough for many applications.

- The word **automata** (the plural of ***automaton***) comes from the Greek word **αὐτόματα**, which means "self-acting". (Wikipedia)

- **Examples:** small computers or **controller** for
  - an automatic door
  - a vending machine
  - an elevator
  - various household appliances such as dishwashers and electronic thermostats
  - parts of digital watches and calculators
  - Various digital controllers in industrial machines

Finite Automata (Finite State Machin – F.S.M)

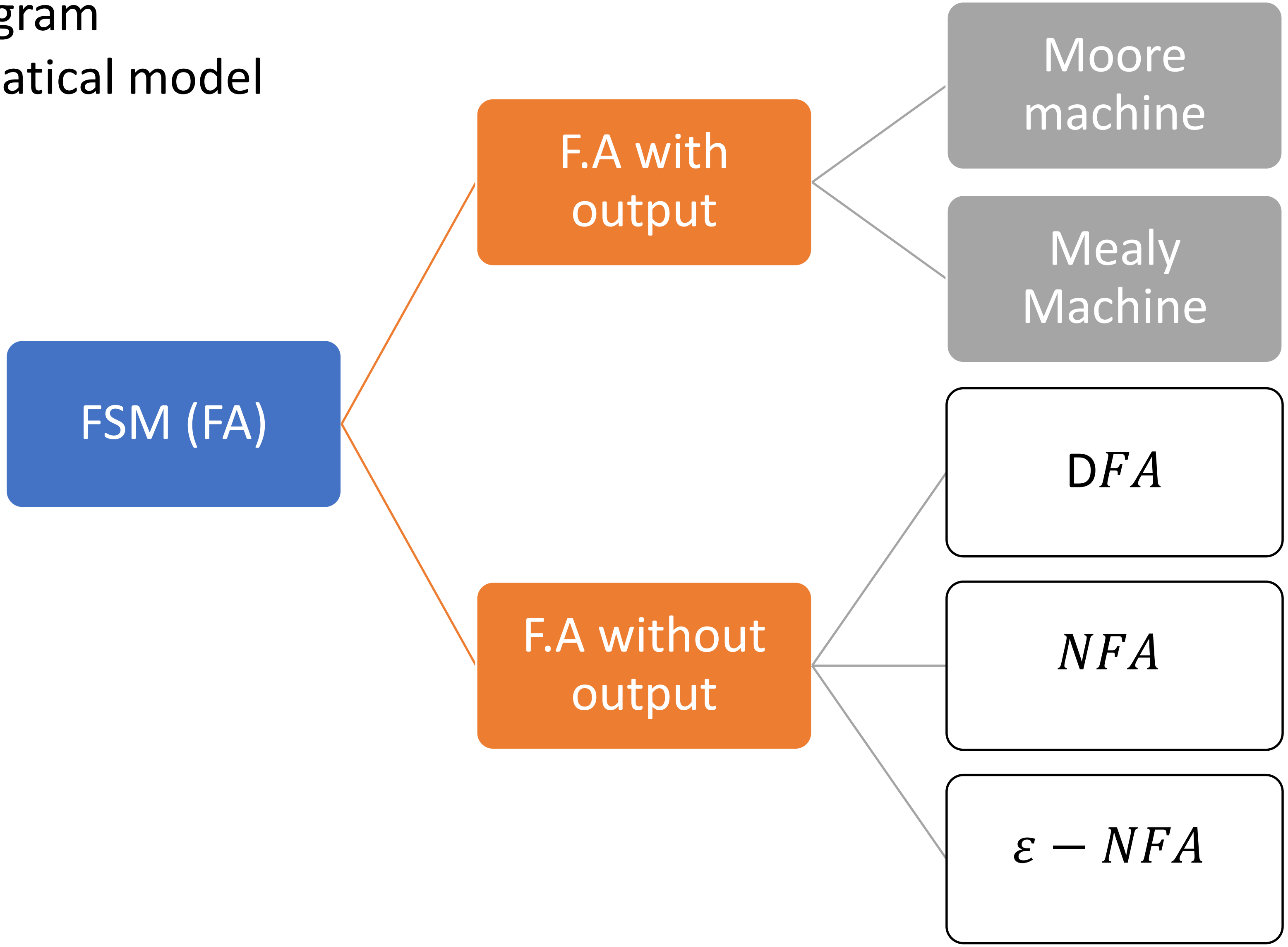
- **Example:** Finite state machine of ECU controller in the ABS system



https://www.researchgate.net/figure/Finite-state-machine-of-ECU-controller-in-the-ABS-system_fig5_265508076

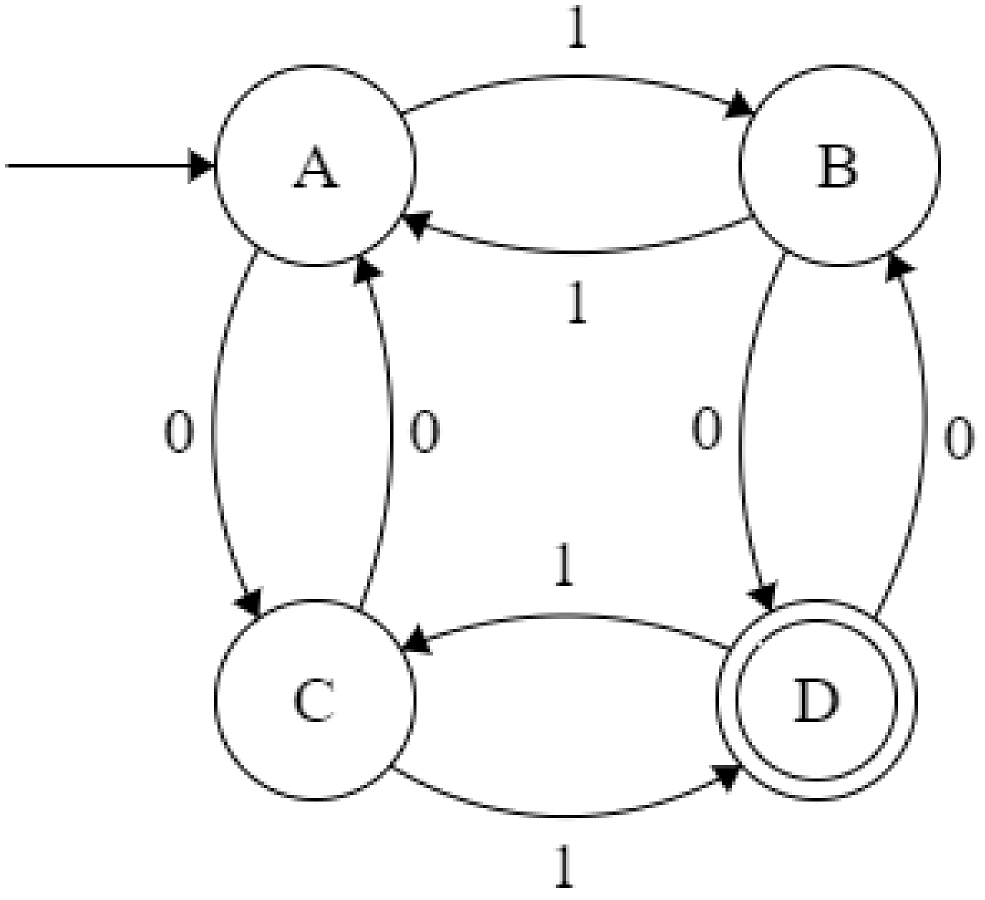
Questions

- How to describe a finite state automaton?
  - by a state diagram
  - by a mathematical model



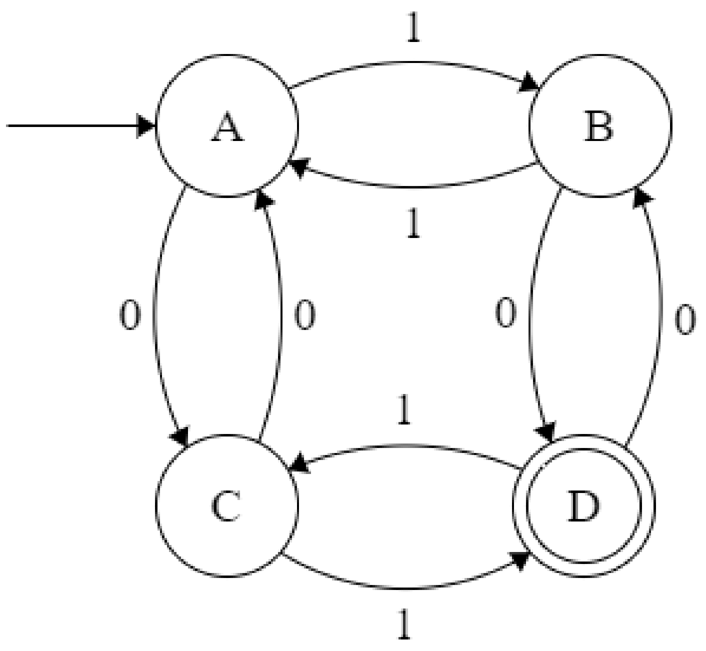
State diagram (Directed graph)

- Elements:
  - **States (nodes)**
  - **Transitions (edges)**
  - **input alphabet**



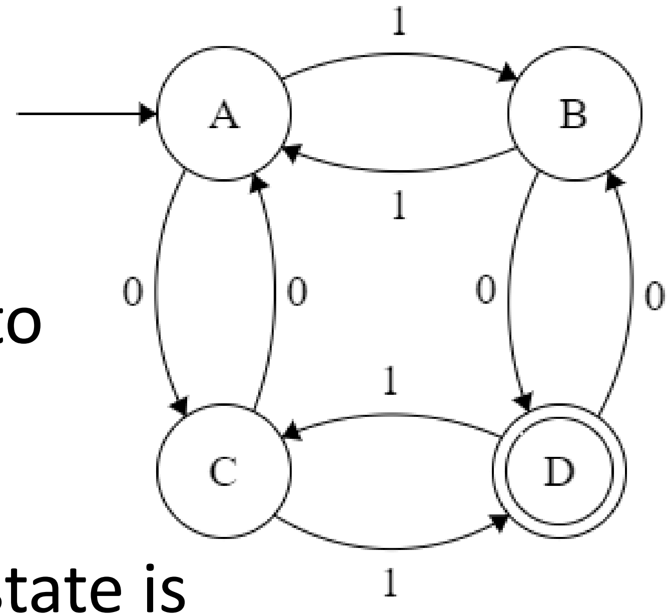
State diagram (Directed graph)

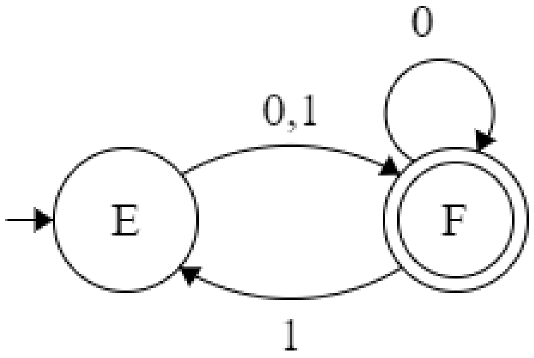
- Elements:
  - **States (nodes):**
    - set of states
    - starting state (initial state)
      - unique
      - indicated by the arrow pointing at it from nowhere
    - accepting (final) states
      - indicated by a double circle
      - may be more than one
  - **Transitions (edges)**
  - **input alphabet**



State diagram (Directed graph)

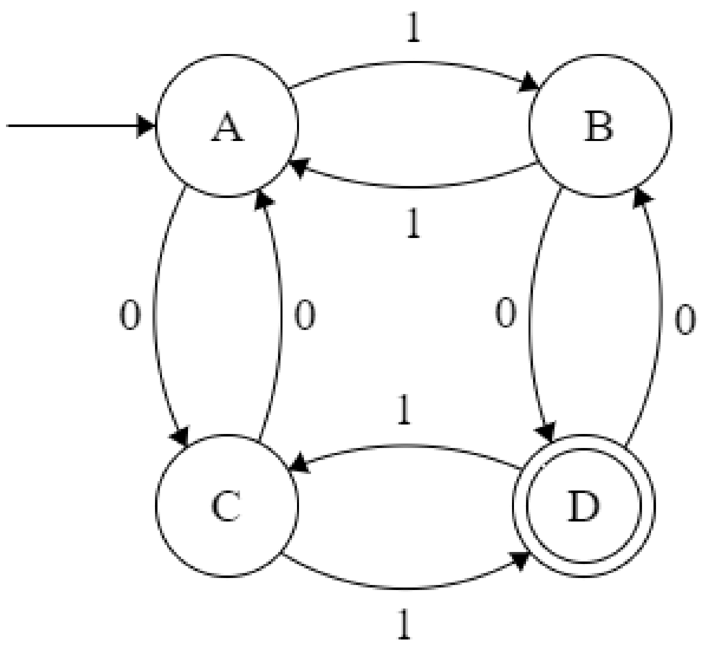
- Elements:
  - **States (nodes):**
  - **Transitions (edges):**
    - the **directed arrows** going from one state to another state
    - the rules for moving
    - Transition δ(A,1) = B means if the current state is A and input is 1, then the state will be changed to state B.
    - $\delta : (Q \times \Sigma) \rightarrow Q$
    - If δ(E,0) = F and δ(E,1) = F , then the directed arrow from E to F can labeled as "0,1".
  - **input alphabet**





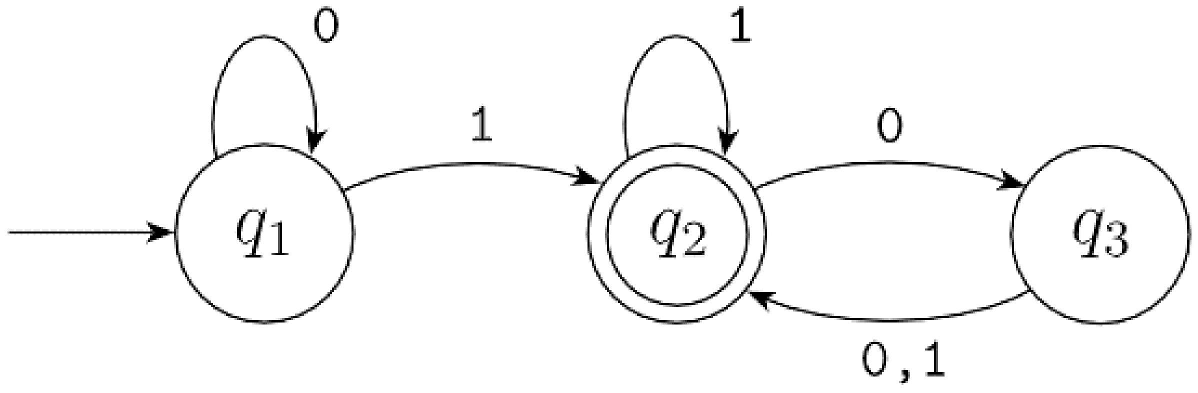
State diagram (Directed graph)

- Elements:
  - **States (nodes):**
  - **Transitions (edges):**
  - **input alphabet**
    - The symbols used for labeling the transitions
    - $\Sigma = \{0, 1\}$



State diagram (Directed graph)

- Elements:
  - States (nodes):
  - Transitions (edges):
  - input alphabet



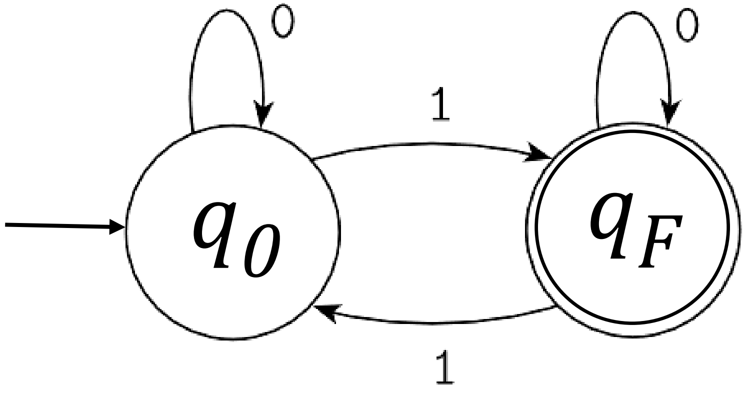
$$M = (\{q_1, q_2, q_3\}, \{0,1\}, \delta, q_1, \{q_2\})$$

language of machine

- Describing a finite automaton by **state diagram is not possible** in some cases. occur when
  - the diagram would be too big to draw or
  - the description depends on some unspecified parameter.

Formal definition of a finite automaton

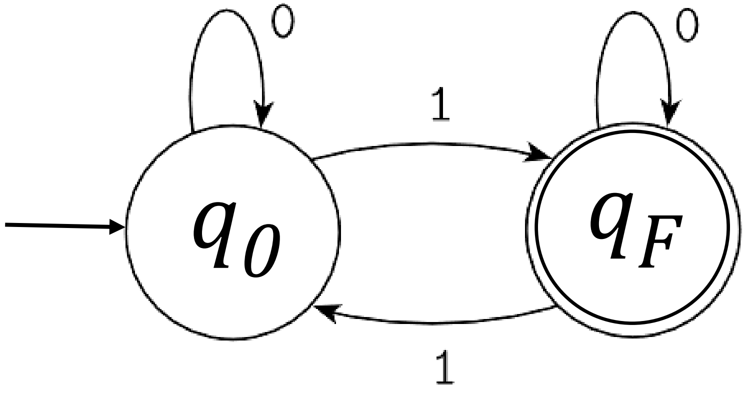
A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where



Formal definition of a finite automaton

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

1. Q is a finite set called the ***states***, → $\{q_0, q_F\}$

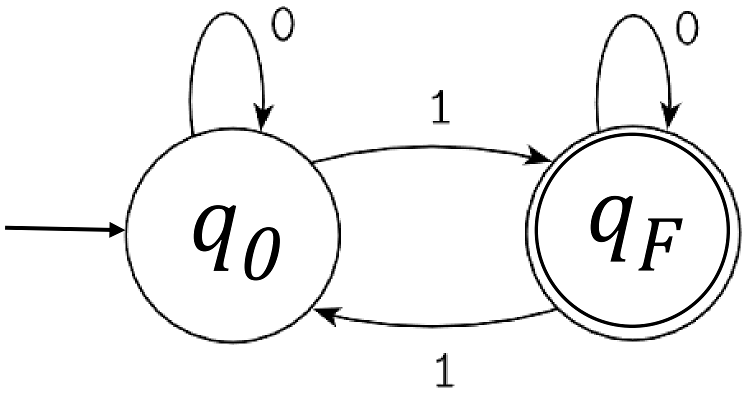


Formal definition of a finite automaton

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

1. Q is a finite set called the ***states***, → $\{q_0, q_F\}$

2. Σ is a finite set called the ***alphabet***, → $\{0, 1\}$



Chapter 1

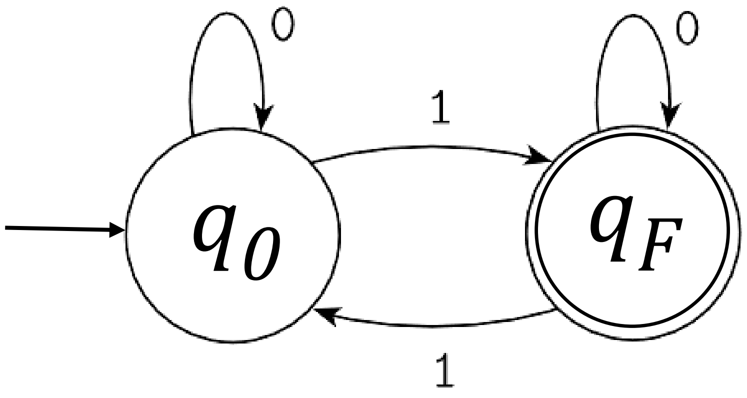
Formal definition of a finite automaton

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

1. Q is a finite set called the ***states***, → $\{q_0, q_F\}$
2. Σ is a finite set called the ***alphabet***, → $\{0, 1\}$
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,

alphabet

states	0	1
q_0	q_0	q_F
q_F	q_F	q_0



Formal definition of a finite automaton

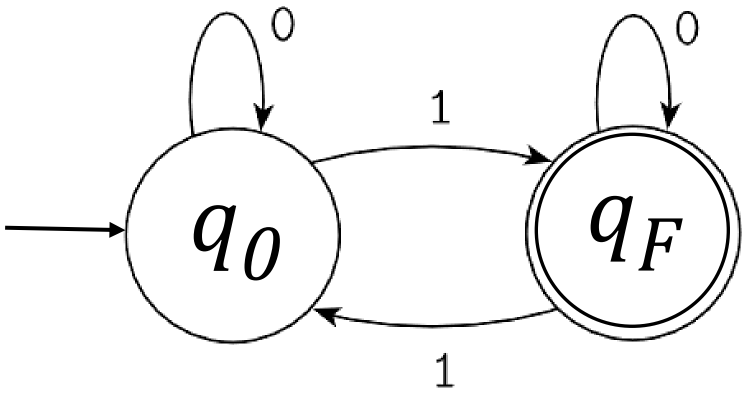
A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***, → $\{q_0, q_F\}$
2. Σ is a finite set called the ***alphabet***, → $\{0, 1\}$
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***, - - - - →
4. $q_0 \in Q$ is the ***start state***, and - - → Unique (q_0)

alphabet

states

	0	1
q_0	q_0	q_F
q_F	q_F	q_0



Formal definition of a finite automaton

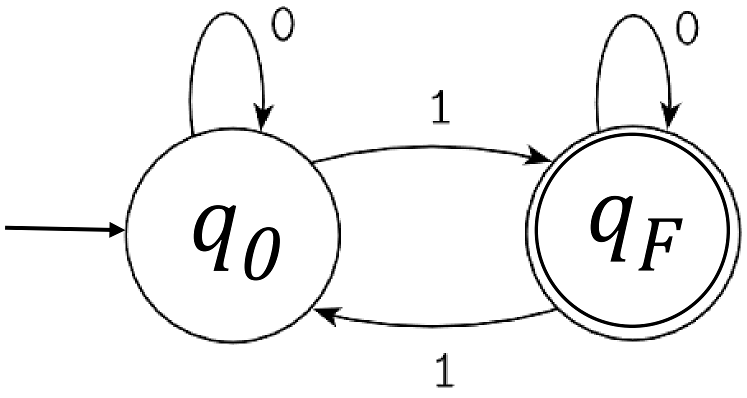
A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***, → $\{q_0, q_F\}$
2. Σ is a finite set called the ***alphabet***, → $\{0, 1\}$
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***, ⇢
4. $q_0 \in Q$ is the ***start state***, and ⇢ Unique (q_0)
5. $F \subseteq Q$ is the ***set of accept states***. ⇢ $\geq$ 0 state
   → $\{q_F\}$

alphabet

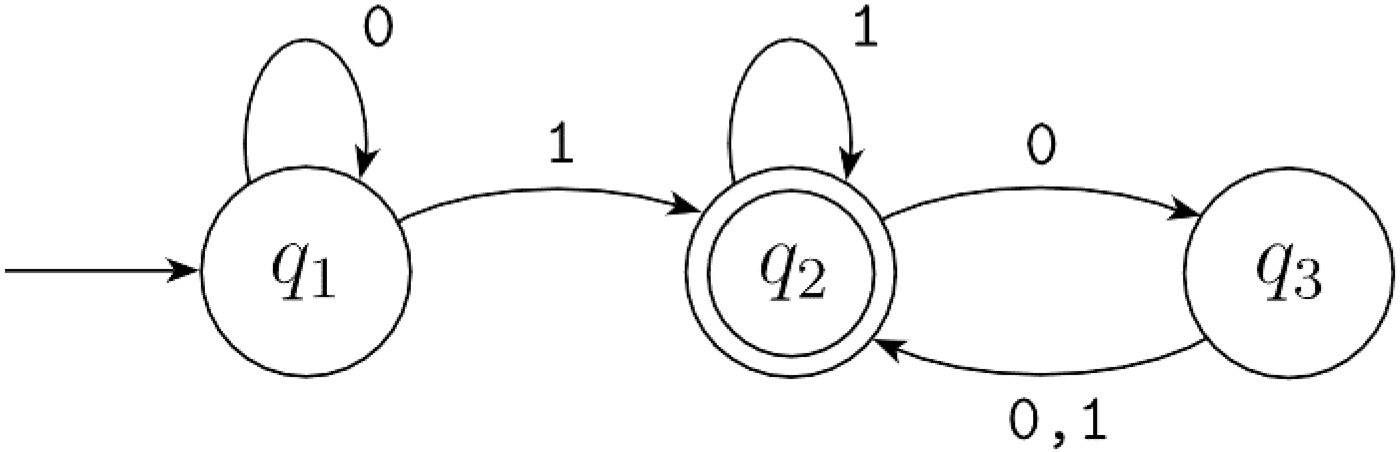
states

	0	1
q_0	q_0	q_F
q_F	q_F	q_0



Examples of finite automata

- Example 2:



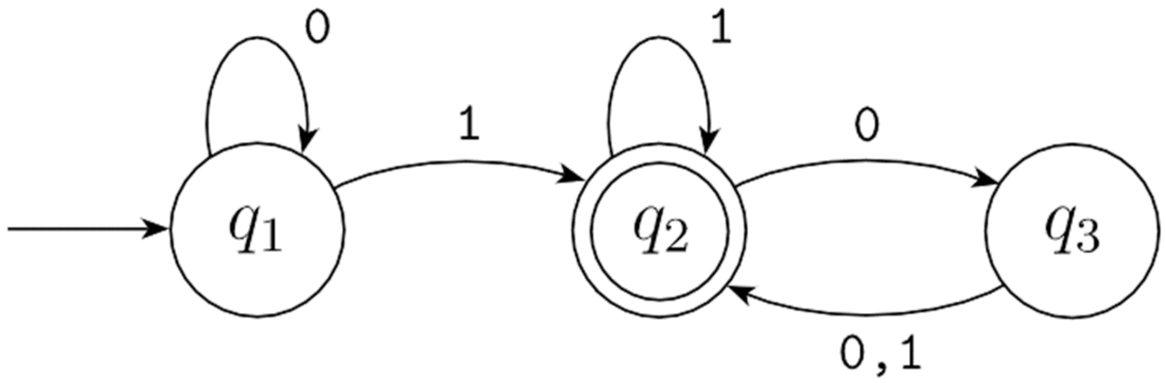
State diagram of the finite automaton M1

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.[2]

?

Examples of finite automata

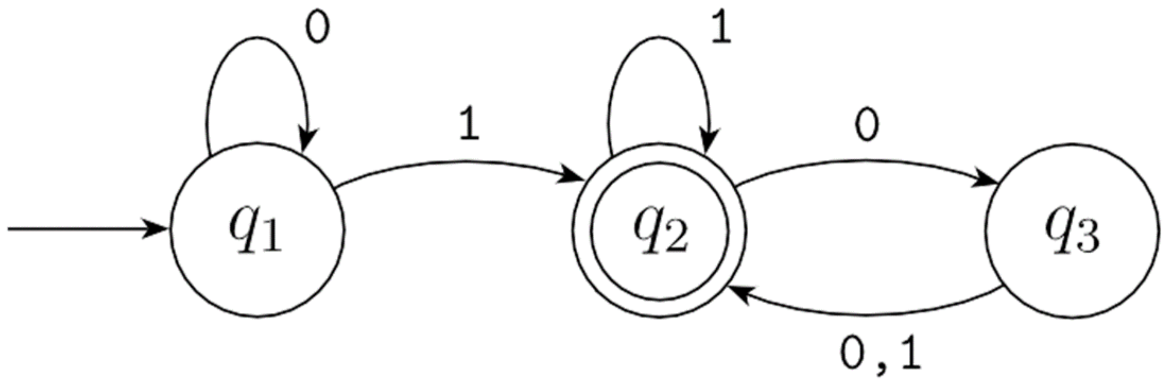
We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where



Examples of finite automata

We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

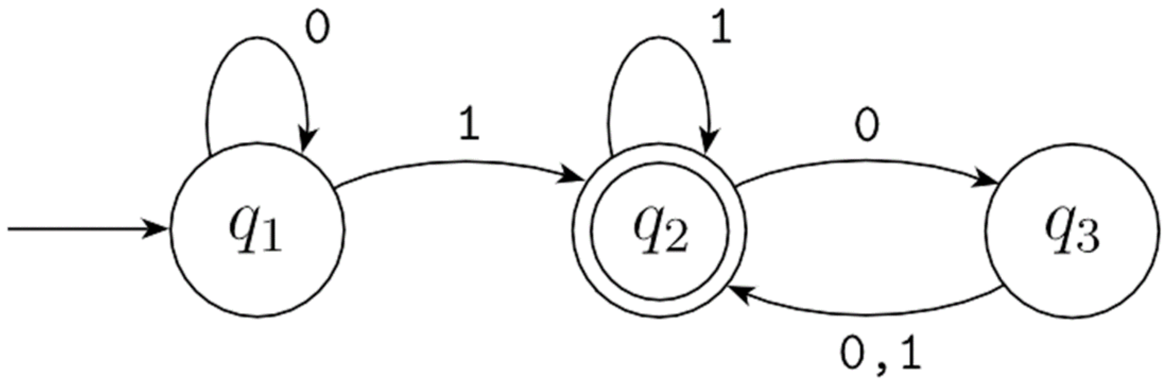
1. $Q = \{q_1, q_2, q_3\}$,



Examples of finite automata

We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,

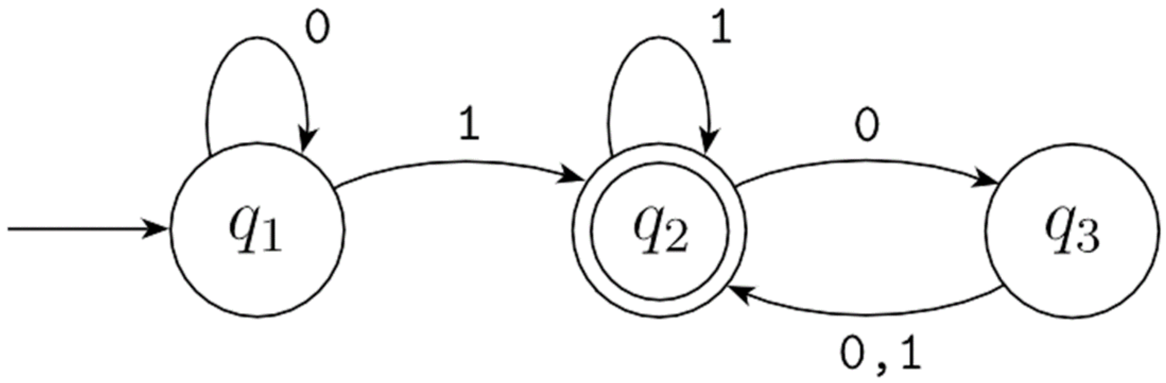


Examples of finite automata

We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,



Examples of finite automata

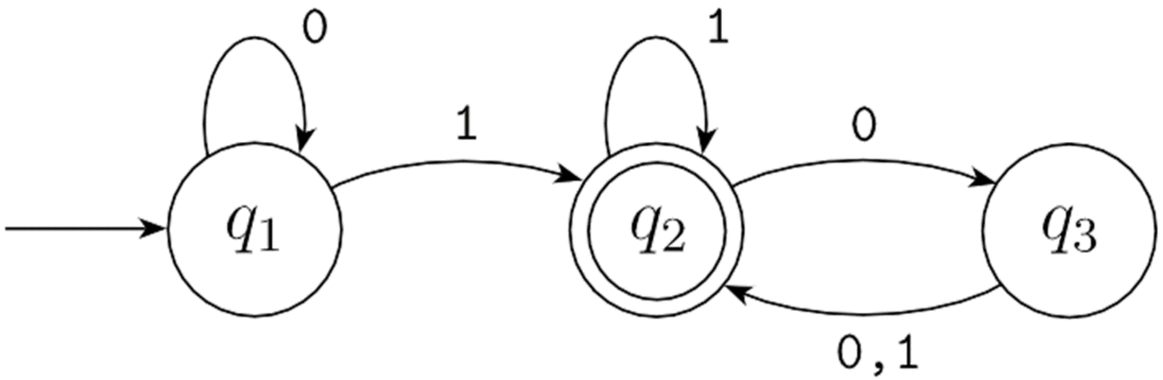
We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,

2. $\Sigma = \{0,1\}$,

3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,



4. q_1 is the start state, and

Examples of finite automata

We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,

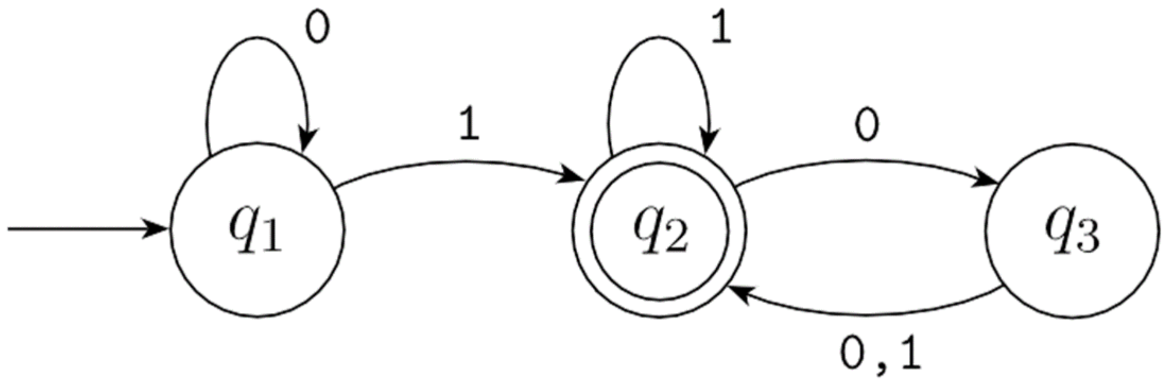
2. $\Sigma = \{0,1\}$,

3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

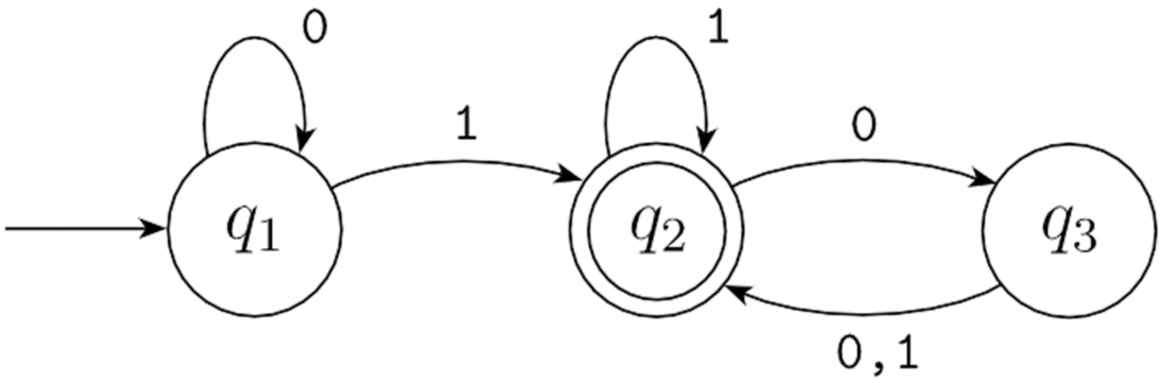
4. q_1 is the start state, and

5. $F = \{q_2\}$.



Examples of finite automata

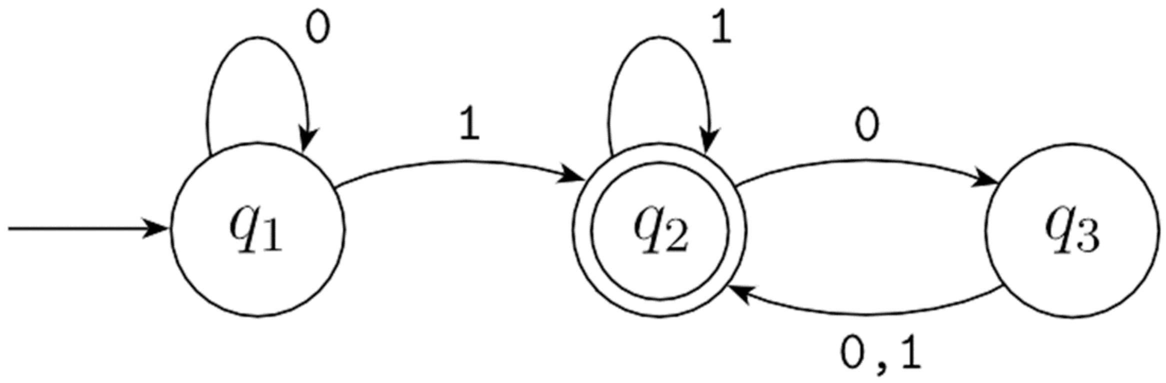
- Question: Does the following machine, M_1, accept string w = 1101 ?



Input	1	1	0	1	
State	q_1				

Examples of finite automata

- Question: Does the following machine, M_1, accept string w = 1101 ?

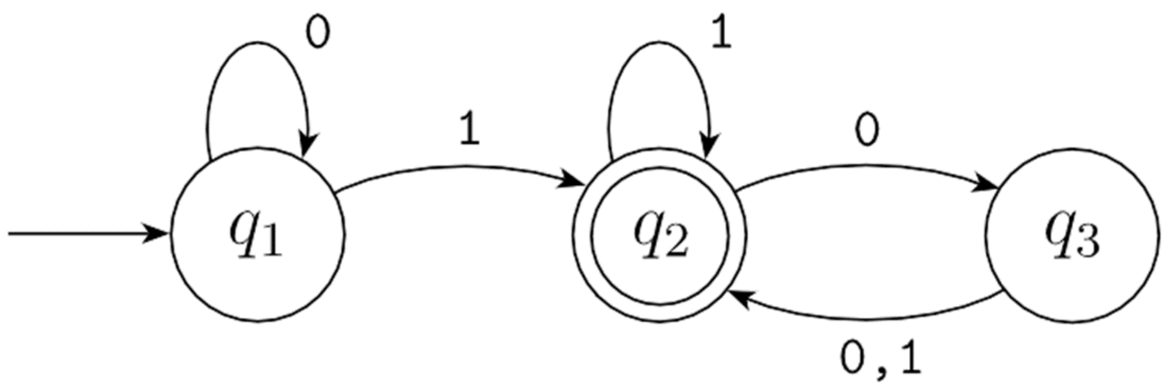


Input	1	1	0	1	
State	q_1	q_2			

Chapter 1

Examples of finite automata

- Question: Does the following machine, M_1, accept string w = 1101 ?

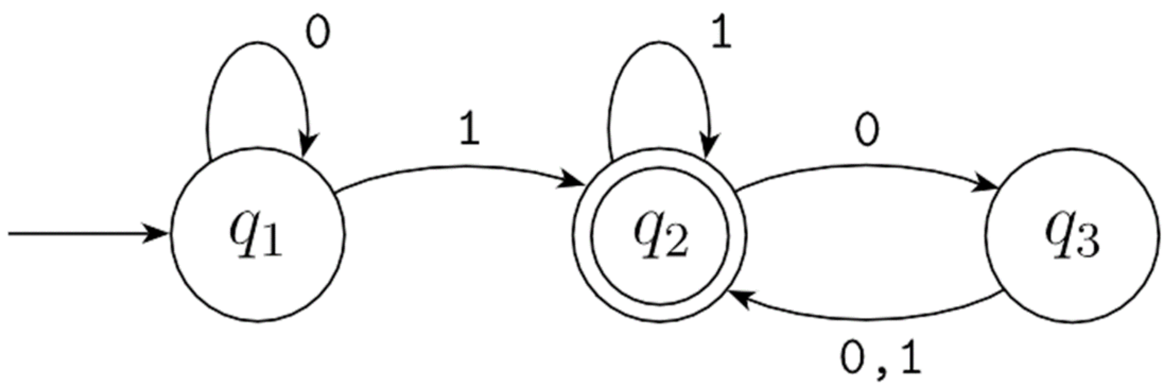


Input	1	1	0	1	
State	q_1	q_2	q_2		

Chapter 1

Examples of finite automata

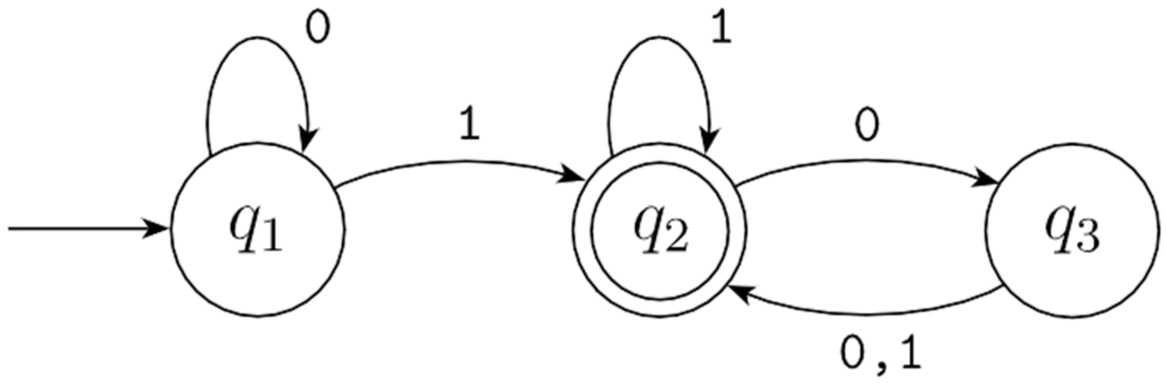
- Question: Does the following machine, M_1, accept string w = 1101 ?



Input	1	1	0	1	
State	q_1	q_2	q_2	q_3	

Examples of finite automata

- Question: Does the following machine, M_1, accept string w = 1101 ?



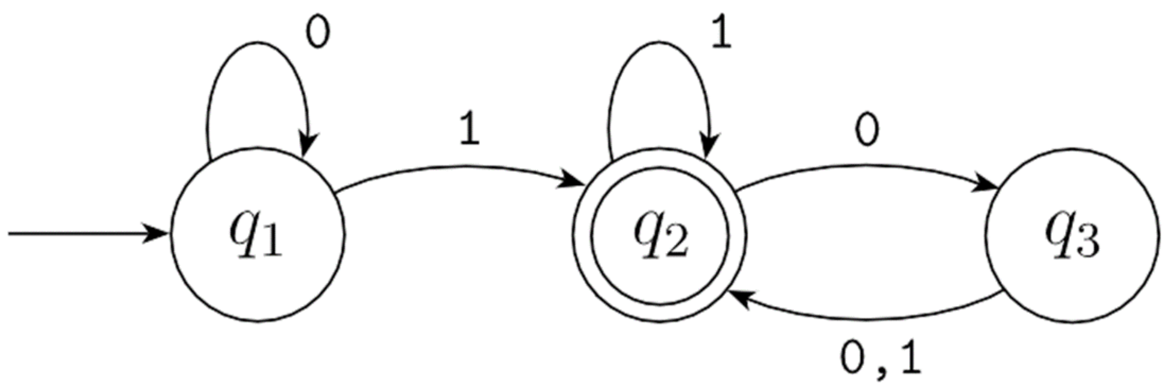
Input	1	1	0	1	
State	q_1	q_2	q_2	q_3	q_2

Accept

- M_1 Accepts w= 1101,
  - because M_1 is in an accept state, q_2, at the end of the input.

Examples of finite automata

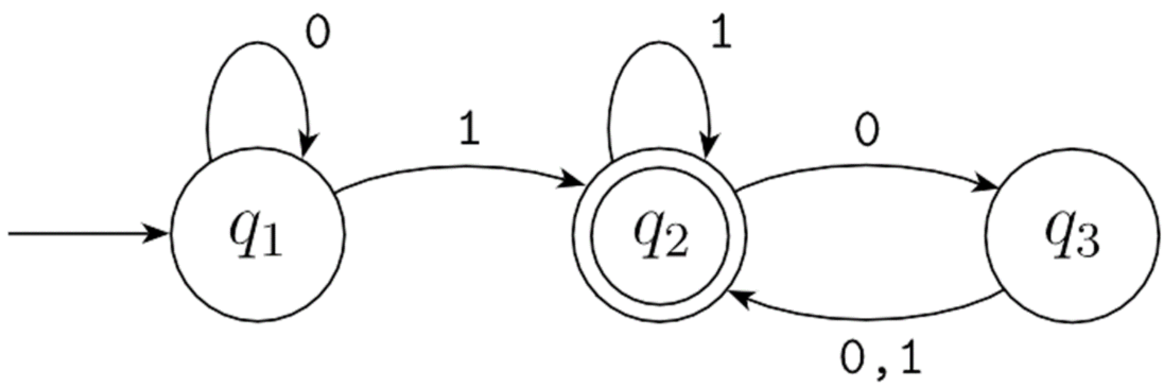
- Question: Does the following machine, M_1, accept string w = 1101 ?



- By experimenting a variety of input strings reveals that it **accepts**
  - the strings1,01,11,and0101010101. (any string that **ends with a 1**)
  - The strings100,0100,110000,and0101000000. (any string that **ends with an even number of 0s following the last 1**.
  - It **rejects** other strings, such as 0,10,101000.

Examples of finite automata

- Does the following machine, M_1, accept string w = 1101 ?



- **Conclusion 1 :** Finite state machine can **recognize (accept) strings**.
- **Conclusion 2 :** Finite state machine can **generate a strings**.
- **Question:** What is the set of the strings that a particular FSM can generate?