FINITE AUTOMATA

FINITE AUTOMATA

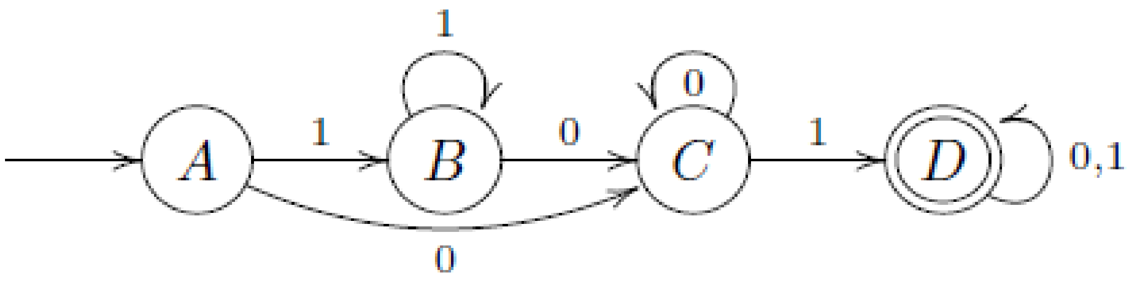
- Are finite collections of states with transition rules that takes you from one state to another.

REPRESENTING FA

- Nodes – States
- Arcs – Indicate state transitions, labels on arc tells what causes the transition
- State Machine/State Diagram – Contains finite number of states, transition, and ends with accept states. It describes the behaviour of the system.
- Transition Table – Shows the movement of a state machine based on the given input

EXAMPLE

- Finite Automaton



Transition Table

	0	1
$\rightarrow$A	C	B
B	C	B
C	C	D
*D	D	D

PARTS OF FINITE AUTOMATON

- A finite set of states
- Rules for going from one state to another depending upon the input symbol
- A finite input alphabet that indicates the allowed symbols
- A start state
- A finite set of accept states

FORMAL DEFINITION

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

1. Q is a finite set called the set of states
2. Σ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
4. $q_0 \in Q$ is the start (or initial) state
5. $F \subseteq Q$ is the set of accept (or final) states

FORMAL AUTOMATON EXAMPLE

$M_1 = (Q, \Sigma, \delta, q_1, F)$ where

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ is described by the table:

δ	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. $F = \{q_2\}$.

STATE MACHINE DIAGRAM

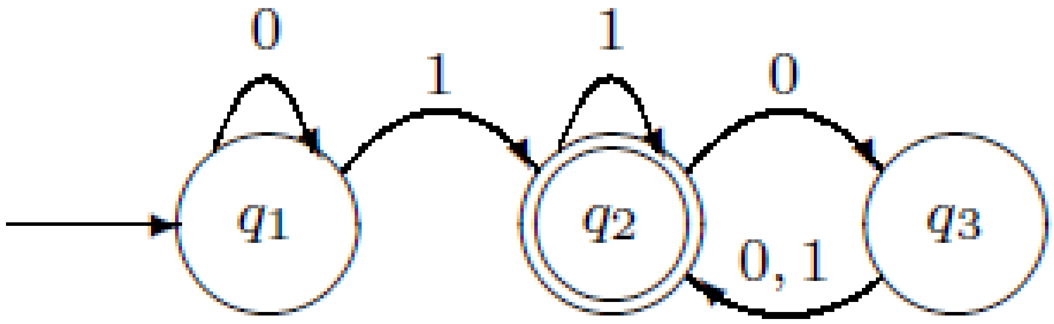


Figure 1: The finite automaton M_1

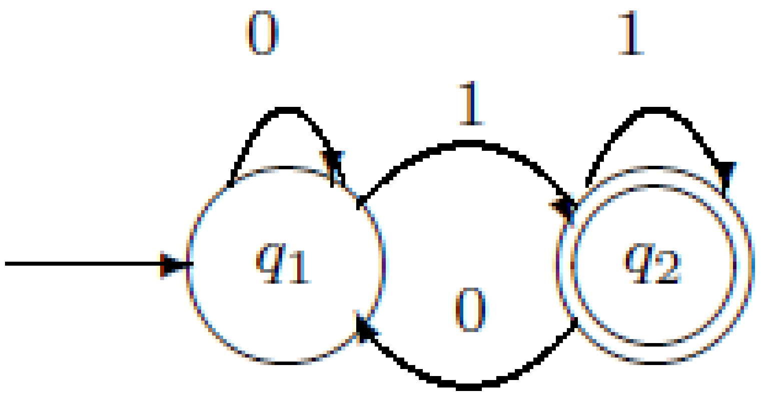
ANOTHER EXAMPLE

- Create a state diagram for the given machine:

$$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\}) \text{ where}$$

$$\delta(q_1, 0) = q_1,\ \delta(q_1, 1) = q_2, \quad \delta(q_2, 0) = q_1,\ \delta(q_2, 1) = q_2$$

STATE DIAGRAM FOR M2



LANGUAGE OF A MACHINE

- Since a finite automaton is used here as the model of a computer we also refer to a finite automaton as a “machine"
- If A is the set of all strings that a machine M accepts, we say that A is the language of the machine M and write $L(M) = A$.

TERMINOLOGY

- The term accept has a different meaning when we refer to machines accepting strings and machines accepting languages. In order to avoid confusion:
- Use accept when we refer to strings
- Use recognize when we refer to languages

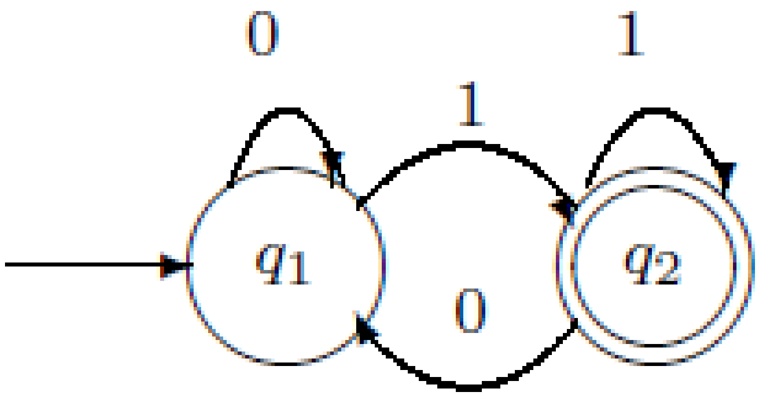
CONSEQUENCES

- A machine may accept several strings, but it always recognizes only one language
- If a machine accepts no strings, it still recognizes one language, namely the empty language $\emptyset$
- Language recognized by machine M_1 is:

  $A = \{w \mid w$ contains at least one 1 and an even number of 0s follow the last 1$\}$
- Conclusion: $L(M_1) = A$, or equivalently, M_1 recognizes A

- A good way of understanding any machine is to try it on some sample input string
- Example: Discover the language of M_2,

$$L(M_2) = \{w | \ w \text{ ends in a 1}\}$$



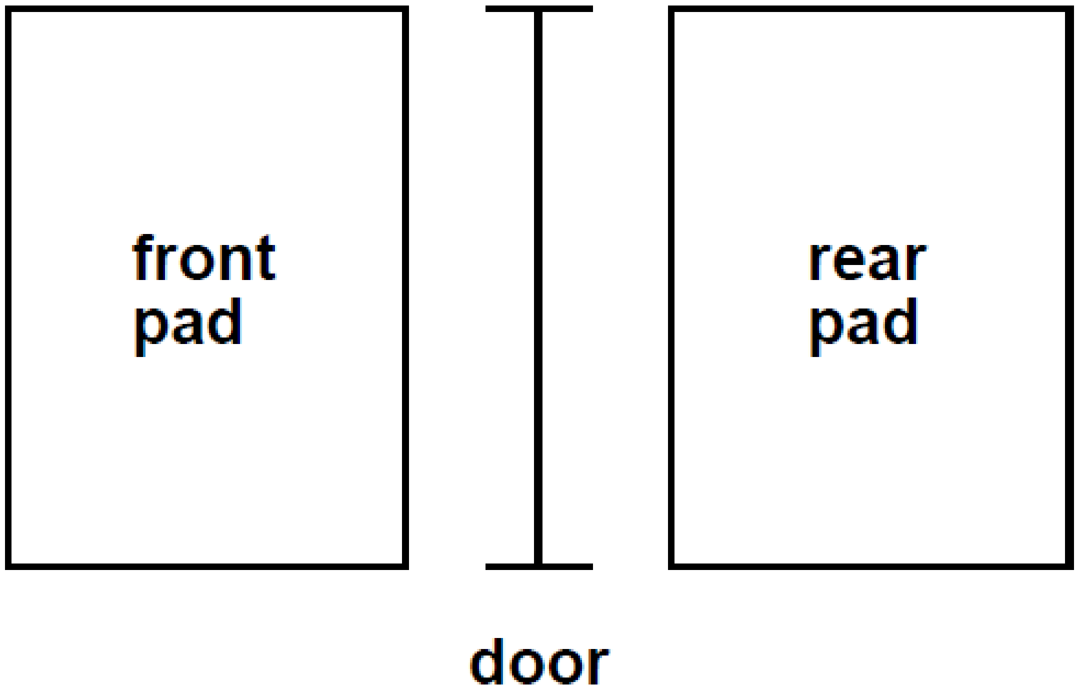
CREATE A MACHINE THAT WILL RECOGNIZE THE GIVEN INPUT:

$A = \{w|\ w$ contains at least one 1 and an even number of 0s follow the last 1$\}$

DETERMINISTIC FINITE AUTOMATA

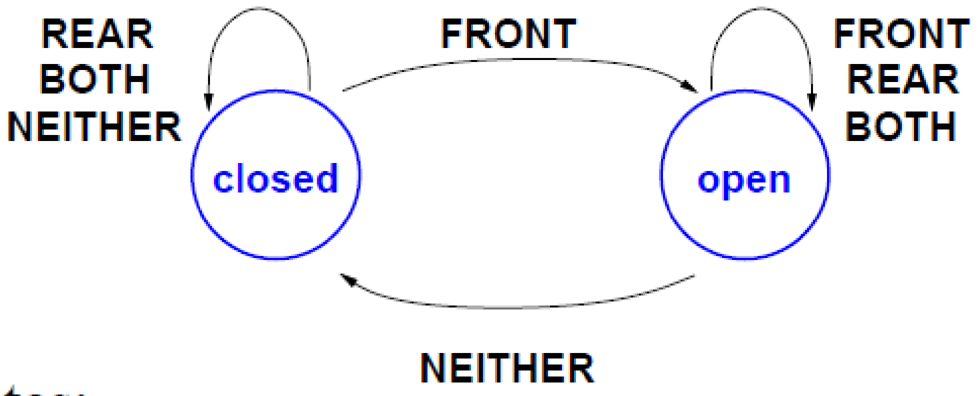
- There is a fixed number of states and we can only be in one state at a time.
- Accepts a string from start state, moves state to state and final state
- Accept or Reject input

EXAMPLE AN AUTOMATIC DOOR



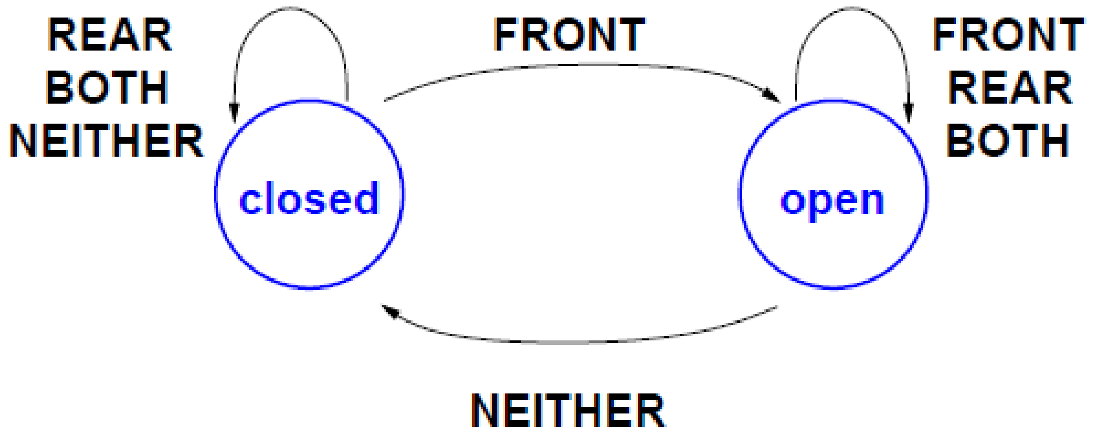
- open when person approaches
- hold open until person clears
- don’t open when someone standing behind door

THE AUTOMATIC DOOR AS DFA



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

THE AUTOMATIC DOOR AS DFA



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

FORMAL DEFINITION

Definition: A deterministic finite automaton (DFA) consists of

1. a finite set of *states* (often denoted Q)
2. a finite set Σ of *symbols* (alphabet)
3. a *transition function* that takes as argument a state and a symbol and returns a state (often denoted δ)
4. a *start state* often denoted q_0
5. a set of *final* or *accepting* states (often denoted F)

We have $q_0 \in Q$ and $F \subseteq Q$

DFA

So a DFA is mathematically represented as a 5-uple

$(Q, \Sigma, \delta, q_0, F)$

The transition function δ is a function in

$Q \times \Sigma \to Q$

$Q \times \Sigma$ is the set of 2-tuples (q, a) with $q \in Q$ and $a \in \Sigma$

EXAMPLE

$Q = \{q_0, q_1, q_2\}$

start state q_0

$F = \{q_1\}$

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

δ is a *function* from $Q \times \Sigma$ to Q

$\delta : Q \times \Sigma \to Q$

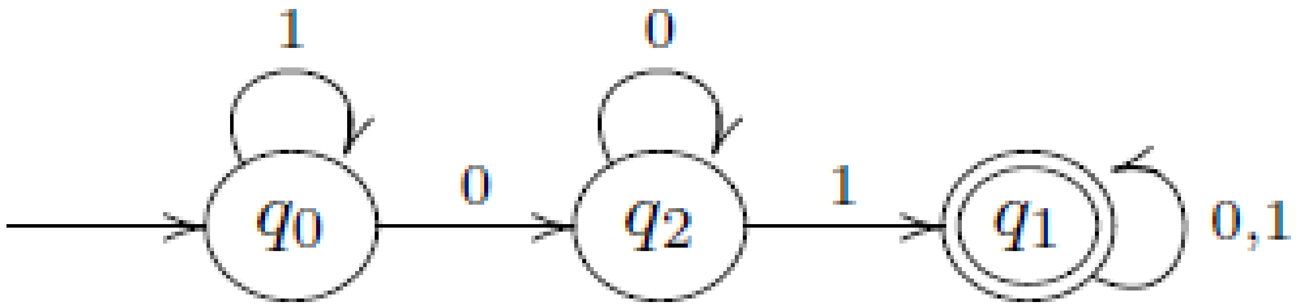
$\delta(q_0, 1) = q_0$

$\delta(q_0, 0) = q_2$

With a *transition table*

	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

STATE DIAGRAM



CREATE A DFA THAT WILL ACCEPT:

- 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

NONDETERMINISTIC FINITE AUTOMATA

- Nondeterminism gives a machine multiple options for its moves.
- In a *nondeterministic* finite automaton (NFA), for each state there can be zero, one, two, or more transitions corresponding to a particular symbol.
- If NFA gets to state with more than one possible transition corresponding to the input symbol, we say it *branches*.
- If NFA gets to a state where there is no valid transition, then that branch *dies*.

NFA ACCEPTANCE

- An NFA accepts the input string if there exists some choice of transitions that leads to ending in an accept state.
- Thus, one accepting branch is enough for the overall NFA to accept, but every branch must reject for the overall NFA to reject.
- This is a model of computation.

NONDETERMINISM AS "GUESS AND VERIFY"

- There are many ways to view nondeterminism. One way is the "*guess and verify*" idea: We assume the NFA is clairvoyant and always guesses correctly the next state to go to. However, the NFA must "check" its guesses.

FORMAL DEFINITION

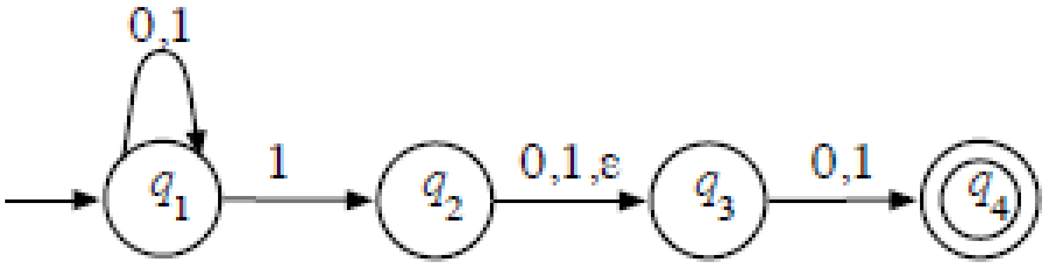
Formally, an NFA is a 5-tuple $(Q, \Sigma, q_0, T, \delta)$ where as before:

- Q is finite set of states;
- Σ is alphabet of input symbols;
- q_0 is start state;
- T is subset of Q giving the accept states; and
- δ is the transition function.

Now the transition function specifies a set of states rather than a state: it maps $Q \times \Sigma$ to $\{$ subsets of Q $\}$.

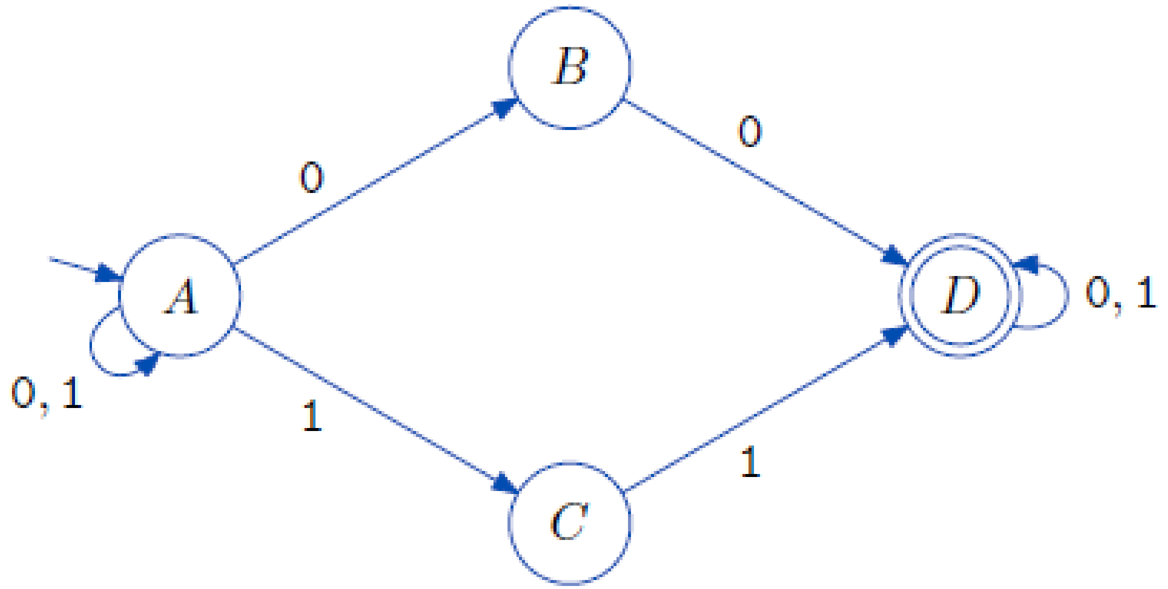
EXAMPLE

state	symbol		
	0	1	ϵ
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\{q_3\}$	$\{ q_3 \}$
q_3	$\{q_4\}$	$\{q_4\}$	$\emptyset$
q_4	$\emptyset$	$\emptyset$	$\emptyset$



ANOTHER EXAMPLE

- It accepts any binary string that contains 00 or 11 as a substring.



EXERCISE

Give an NFA for the set of all binary strings that have either the number of `0`'s odd, or the number of `1`'s not a multiple of 3, or both.

SOLUTION