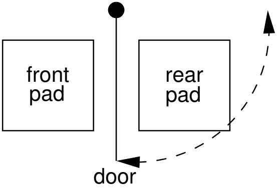
Finite Automata 01

Thumrongsak Kosiyatrakul
tkosiyat@cs.pitt.edu

Finite Automata

- Suppose you are asked to write a software to control an automatic door as shown below:



- Assume we have the following methods:
  - `getFrontPad()`: returns true if there is a person standing on the front pad. Otherwise, it returns false.
  - `getRearPad()`: returns true if there is a person standing on the rear pad. Otherwise, it returns false.
  - `openDoor()`: when called it will open the door.
  - `closeDoor()`: when called it will close the door.
- How the write the program in Java?

Finite Automata

- Program to control the automatic door:

```
public class DoorController {
    public static void main(String[] args) {
        boolean isDoorOpen = false;

        while(true) {
            if(getFrontPad() && !getRearPad() && !isDoorOpen) {
                openDoor();
                isDoorOpen = true;
            }
            if(!getFrontPad() && !getRearPad() && isDoorOpen) {
                closeDoor();
                isDoorOpen = false;
            }
        }
    }
}
```

- The variable `isDoorOpen` of type boolean is used to record the status of the door (1 bit of memory is required).

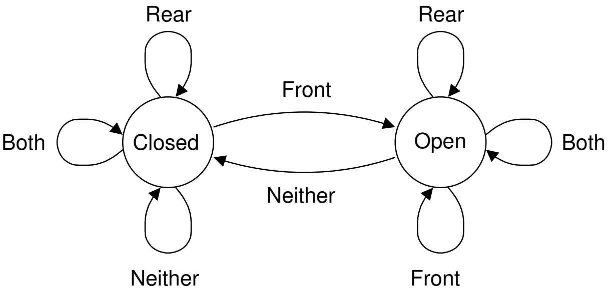
Finite Automata

- `getFrontPad()` and `getRearPad()` together acts as external input to the program:

`getFrontPad()`	`getRearPad()`	Input
`true`	`true`	Both
`true`	`false`	Front
`false`	`true`	Rear
`false`	`false`	Neither

- We can define the behavior of our program based on its input as well as the status of the door whether it is current open or close

- The program can be represented in two standard ways
  - State Diagram:



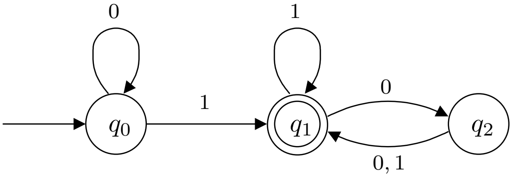
  - State Transition Table:

	Neither	Front	Rear	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open

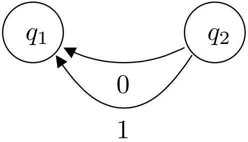
- But how to represent these in a mathematical way?

Finite State Machine

- Consider the following finite state machine M_1:

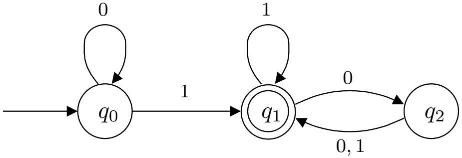


- Machine M_1 consists of:
  - Three **states**: q_0, q_1, and q_2
  - The **start state** q_0 (arrow pointing to it from nowhere)
  - An **accept state** q_1 (double circle)
    - All single circle states are called non-accept state
  - Arrows represent **transition functions**
  - The label $0, 1$ represents two transitions



Finite State Machine

- Consider the following finite state machine M_1:



- When an input string is given to this machine, it returns either **accept** or **reject**.
  - 1101: accept

$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \quad \text{(an accept state)}$$

  - 0010: reject

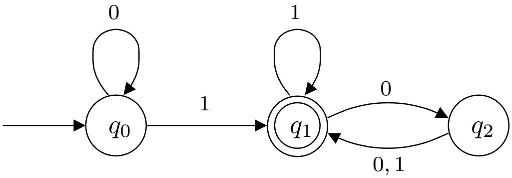
$$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \quad \text{(a non-accept state)}$$

  - 0100: accept

$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_1 \quad \text{(an accept state)}$$

Finite State Machine

- Consider the following finite state machine M_1:



- Can we define the set of inputs that is accepted by the above machine?
  - M_1 accepts any strings that end with a 1
  - M_1 also accepts a string that ends with a 0 but it needs to have even number of 0s after the last 1
- The set of all strings accepted by this machine is

  $\{x \mid x$ ends with a 1 and x is a string

  that ends with an even number of 0s following the last 1$\}$
- The above set is called the **language** of the machine M_1

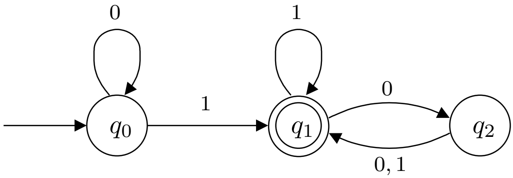
Finite-State Automaton

- A finite state machine M can be defined as five tuple

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

  - Q is a non-empty finite set of states
    - M must have at least one state
  - Σ is an alphabet (a finite set of symbols)
  - $\delta : Q \times \Sigma \to Q$ is the transition functions
    - We generally use a table to represent δ
  - $q_0 \in Q$ is the starting state
    - A finite automata can only have **exactly one** start state
  - $F \subseteq Q$ is the set of accept states
    - F can be $\emptyset \rightsquigarrow M$ can have no accept state (rejects all strings)
    - $|F|$ can be more than 1 $\rightsquigarrow M$ has more than one accept states

Example: Machine M_1



- $M_1 = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, q_2\}$
  - $\Sigma = \{0, 1\}$
  - δ can be defined using the table below:

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_1

  - q_0 is the start state
  - $F = \{q_1\}$
- The state diagram and its formal definition are equivalent

Formal Definition of Machine M_1

- $M_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ where δ is as follows:

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_1

- The above formal definition allows use to precisely answer questions about M_1:
  - Is 0101 is a valid input for this machine?
    - Yes. $0 \in \{0, 1\}$ and $1 \in \{0, 1\}$.
  - Is 01a0 is a valid input for this machine?
    - No. $a \notin \{0, 1\}$
  - Is input 010 accepted by this machine?
    - No. $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$ and $q_2 \notin \{q_1\}$.
  - Is input 101 accepted by this machine?
    - Yes. $q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1$ and $q_1 \in \{q_1\}$.

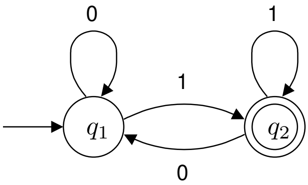
Language Recognized

- A string $w = w_1 w_2 \ldots w_n$ is **accepted** by M if and only if after processing each symbol w_i of w, where $1 \leq i \leq n$, M finds itself in an accept state (a state belonging to F). Otherwise, we say w is rejected by M.
- If A is the set of **all strings** accepted by M, we say A is the **language of finite-state machine** M, denoted by

  $$L(M) = A$$

  We say that M **recognizes** A
- A machine may accept several strings but it always recognizes only one language.

Example



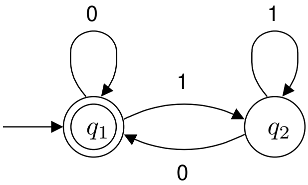
$$M_2 = (Q, \Sigma, \delta, \text{start state}, F)$$

What is the formal definition of the above machine and the language that it recognises?

- $Q = \{q_1, q_2\}$
- $\Sigma = \{0, 1\}$
-

δ	0	1
q_1	q_1	q_2
q_2	q_1	q_2

- The start state is q_1
- $F = \{q_2\}$
- $L(M_2) = \{w \mid w \text{ ends in a } 1\}$



$$M_3 = (Q, \Sigma, \delta, \text{start state}, F)$$

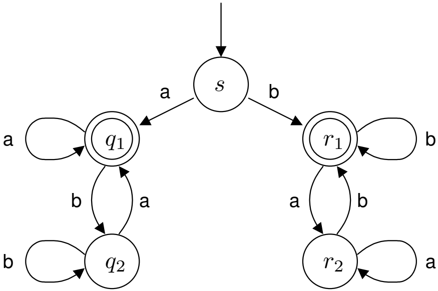
What is the formal definition of the above machine and the language that it recognises?

- $Q = \{q_1, q_2\}$
- $\Sigma = \{0, 1\}$
-

δ	0	1
q_1	q_1	q_2
q_2	q_1	q_2

- Start state is q_1
- $F = \{q_1\}$
- $L(M_2) = \{w \mid w \text{ is the empty string } \varepsilon \text{ or ends in a } 0\}$

Example



$M_4 = (Q, \Sigma, \delta, \text{start state}, F)$

What is the formal definition of the above machine and the language that it recognises?

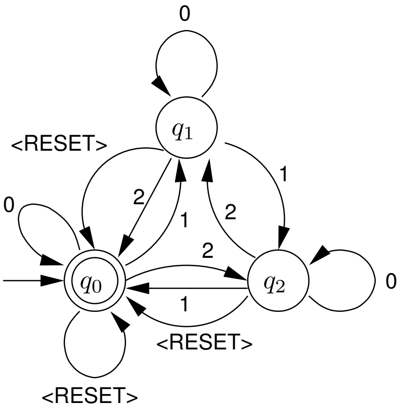
- $Q = \{s, q_1, q_2, r_1, r_2\}$
- $\Sigma = \{a, b\}$
- Transition Functions:

δ	a	b
s	q_1	r_1
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_2	r_1
r_2	r_2	r_1

- Start state is s
- $F = \{q_1, r_1\}$
- $L(M_2) =$

  $\{w \mid w$ starts and ends with the same symbol$\}$

Example



$M_5 = (Q, \Sigma, \delta, \text{start state}, F)$

What is the formal definition of the above machine and the language that it recognises?

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1, 2, \texttt{<RESET>}\}$
- Transition Functions:

δ	0	1	2	<RESET>
q_0	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_0
q_2	q_2	q_0	q_1	q_0

- Start state is q_0
- $F = \{q_0\}$
- $L(M_2) =$

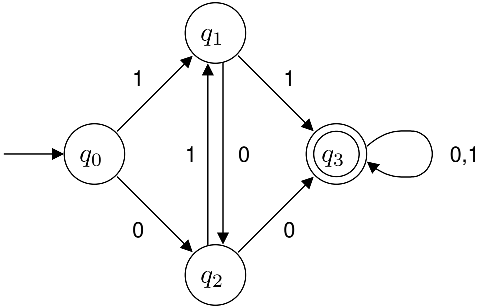
$\{w \mid w$ is the empty string ε or
ends with <RESET> or
sum of input is multiple
of 3 after the last
<RESET>$\}$

Designing a Finite-State Machine

- A computation model simulates a set of algorithms
- Designing a finite-state machine is the same as writing a program
  - Use states to capture state-of-minds
    - I just see a 1
    - I just see two consecutive 0s
    - I already saw 00 or 11
- Do not force yourself to use the least number of states
  - Nobody asks you to write a shortest possible program
  - Unless you are asked to do so

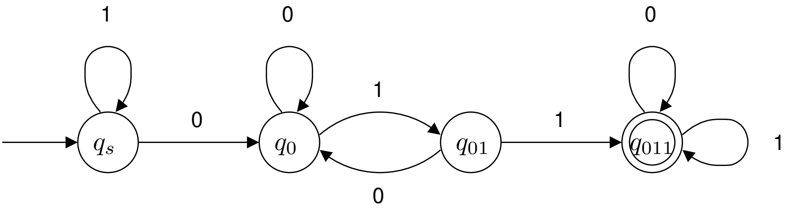
Designing Finite Automata

Suppose the alphabet Σ is $\{0, 1\}$. Create a machine such that its language is the set of all strings that contain either 11 or 00 as a substring.



- Common mistakes:
  - $\delta(q_1, 0) = q_0$
  - $\delta(q_2, 1) = q_0$

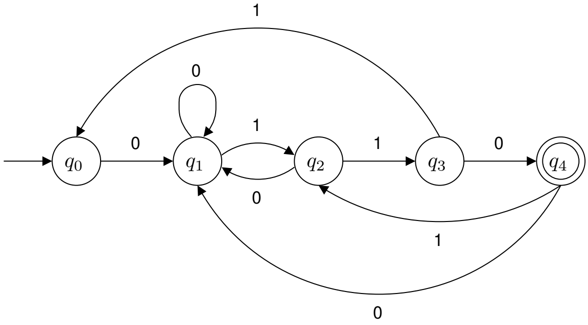
Suppose the alphabet Σ is $\{0, 1\}$. Create a machine such that its language is the set of all strings that contain 011 as a substring.



- Common mistakes:
  - $\delta(q_0, 0) = q_s$
  - $\delta(q_{01}, 0) = q_s$
- **Hint**: Name of a state can be used to indicate a state-of-mind
  - q_{01} means "I just see a 0 immediately followed by a 1"

Designing Finite Automata

Suppose the alphabet Σ is $\{0, 1\}$. Create a machine such that its language is the set of all strings that ends with 0110.



- Common mistakes:
  - $\delta(q_2, 0) = q_0$
  - $\delta(q_4, 0) = q_0$
  - $\delta(q_4, 1) = q_0$