

### Lecture three

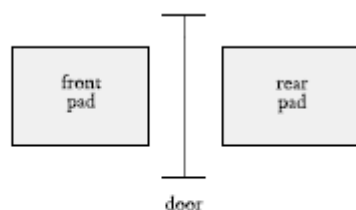
#### **Topics that must be covered in this lecture:**

- **Introduction to Finite Automata**
  - **Formal definition of Finite state automata.**
  - **Designing (drawing) FA.**
  - **Language accepted by FA.**
- 

#### **Introduction to Finite Automata:**

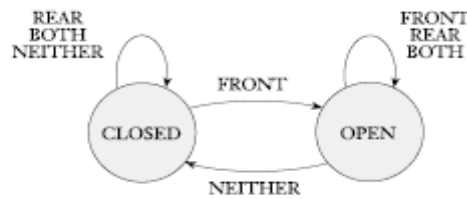
Finite automata are good models for computers with an extremely limited amount of memory. What can a computer do with such a small memory? Many useful things! In fact, we interact with such computers all the time, as they lie at the heart of various electromechanical devices.

The controller for an automatic door is one example of such a device. Often found at supermarket entrances and exits, automatic doors swing open when the controller senses that a person is approaching. An automatic door has a pad in front to detect the presence of a person about to walk through the doorway. Another pad is located to the rear of the doorway so that the controller can hold the door open long enough for the person to pass all the way through and also so that the door does not strike someone standing behind it as it opens. This configuration is shown in the following figure.



**FIGURE**  
Top view of an automatic door

The controller is in either of two states: “OPEN” or “CLOSED,” representing the corresponding condition of the door. As shown in the following figures, there are four possible input conditions: “FRONT” (meaning that a person is standing on the pad in front of the doorway), “REAR” (meaning that a person is standing on the pad to the rear of the doorway), “BOTH” (meaning that people are standing on both pads), and “NEITHER” (meaning that no one is standing on either pad).



**FIGURE**  
 State diagram for an automatic door controller

		input signal			
		NEITHER	FRONT	REAR	BOTH
state	CLOSED	CLOSED	OPEN	CLOSED	CLOSED
	OPEN	CLOSED	OPEN	OPEN	OPEN

**FIGURE**  
 State transition table for an automatic door controller

The controller moves from state to state, depending on the input it receives.

When in the CLOSED state and receiving input NEITHER or REAR, it remains in the CLOSED state. In addition, if the input BOTH is received, it stays CLOSED because opening the door risks knocking someone over on the rear pad. But if the input FRONT arrives, it moves to the OPEN state. In the OPEN state, if input FRONT, REAR, or BOTH is received, it remains in OPEN. If input NEITHER arrives, it returns to CLOSED.

For example, a controller might start in state CLOSED and receive the series of input signals FRONT, REAR, NEITHER, FRONT, BOTH, NEITHER, REAR, and NEITHER. It then would go through the series of states CLOSED (starting), OPEN, OPEN, CLOSED, OPEN, OPEN, CLOSED, CLOSED, and CLOSED.

Thinking of an automatic door controller as a finite automaton is useful because that suggests standard ways of representation as in Figures 2 and 3.

This controller is a computer that has just a single bit of memory, capable of recording which of the two states the controller is in.

Finite automata and their probabilistic counterpart Markov chains are useful tools when we are attempting to recognize patterns in data. These devices are used in speech processing and in optical character recognition. Markov chains have even been used to model and predict price changes in financial markets.

**FORMAL DEFINITION OF A FINITE AUTOMATON (FA):**

FA is a device consisting of a tape and a control circuit in figure 4 which satisfy the following conditions:

1. The tape starts from left end and extends to the right without an end.
2. The tape is dividing into squares in each a symbol.
3. The tape has a read only head.
4. The head moves to the right one square every time it reads a symbol. It never moves to the left. When it sees no symbol, it stops and the automata terminate its operation.
5. There is a control determines the state of the automaton and also controls the movement of the head.

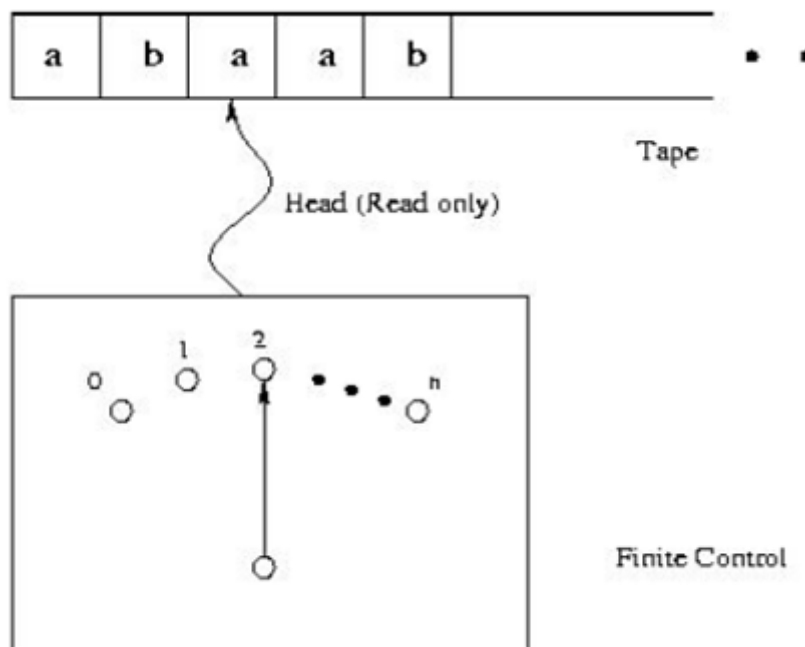

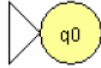




Figure 4: finite automata

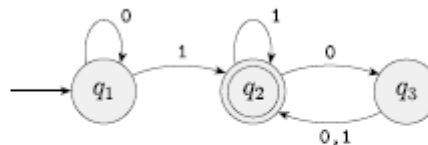
A finite automaton is defined formally by a 5-tuple  $(Q, \Sigma, \delta, q_0, F)$ , where:

1.  $Q$  is a finite set called the states,
2.  $\Sigma$  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 state, and
5.  $F \subseteq Q$  is the set of accept states (final state).

### Designing (drawing) FA:

State with numbers or any name	Start - or small arrow	Final + or double circle	Transition (only one input or symbol on the edge) a,b allowed means (a or b)
			

Example: The following figure 5 depicts a finite automaton called M1.



**FIGURE**  
 A finite automaton called  $M_1$  that has three states

Figure above is called the state diagram of M1. It has three states, labeled  $q_1$ ,  $q_2$ , and  $q_3$ . The **start state**,  $q_1$ , is indicated by the arrow pointing at it from nowhere. The **accept state**,  $q_2$ , is the one with a double circle. The arrows going from one state to another are called **transitions**.

When this automaton receives an input string such as 1101, it processes that string and produces an output. The output is either **accept or reject**. We will consider only this yes/no type of output for now to keep things simple. The processing begins in M1's start state. The automaton receives the symbols from the input string one by one from left to right. After reading each symbol, M1 moves from one state to another along the transition that has that symbol as its label. When it reads the last symbol, M1 produces its output. The output is accept if M1 is now in an accept state and reject if it is not.

For example, when we feed the input string 1101 into the machine M1 in Figure 5, the processing proceeds as follows:

1. Start in state  $q_1$ .
2. Read 1, follow transition from  $q_1$  to  $q_2$ .
3. Read 1, follow transition from  $q_2$  to  $q_2$ .
4. Read 0, follow transition from  $q_2$  to  $q_3$ .
5. Read 1, follow transition from  $q_3$  to  $q_2$ .
6. Accept because M1 is in an accept state  $q_2$  at the end of the input.

Experimenting with this machine on a variety of input strings reveals that it accepts the strings 1, 01, 11, and 0101010101. In fact, M1 accepts any string that ends with a 1, as it goes to its accept state  $q_2$  whenever it reads the symbol 1. In addition, it accepts strings 100, 0100, 110000, and 0101000000, and any string that ends with an even

number of 0s following the last 1. It rejects other strings, such as 0, 10, 101000. Can you describe the language consisting of all strings that M1 accepts? We will do so shortly.

We can describe M1 formally by writing  $M1 = (Q, \Sigma, \delta, q1, F)$ , where

1.  $Q = \{q1, q2, q3\}$ ,
2.  $\Sigma = \{0,1\}$ ,
3.  $\delta$  is described as

	0	1
$q1$	$q1$	$q2$
$q2$	$q3$	$q2$
$q3$	$q2$	$q2$

4.  $q1$  is the start state, and
5.  $F = \{q2\}$ .

If A is the set of all strings that machine M accepts, we say that A is the language of machine M and write  $L(M) = A$ . We say that M recognizes A or that M accepts A. Because the terms accept has different meanings when we refer to machines accepting strings and machines accepting languages, we prefer the term recognize for languages in order to avoid confusion.

A machine may accept several strings, but it always recognizes only one language.

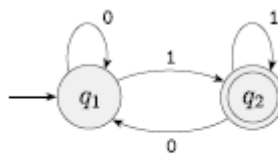
If the machine accepts no strings, it still recognizes one language namely, the empty language  $\emptyset$ .

In our example, let

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

Then  $L(M1) = A$ , or equivalently, M1 recognizes A.

Example: Here is the state diagram of finite automaton M2.



**FIGURE**  
 State diagram of the two-state finite automaton  $M2$

In the formal description, M2 is  $(\{q1, q2\}, \{0,1\}, \delta, q1, \{q2\})$ . The transition function  $\delta$  is

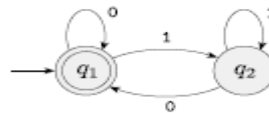
	0	1
$q1$	$q1$	$q2$
$q2$	$q1$	$q2$

On the sample string 1101, the machine M2 starts in its start state  $q1$  and proceeds first to state  $q2$  after reading the first 1, and then to states  $q2, q1,$  and  $q2$  after reading 1, 0,

and 1. The string is accepted because  $q_2$  is an accept state. But string 110 leaves  $M_2$  in state  $q_1$ , so it is rejected. After trying a few more examples, you would see that  $M_2$  accepts all strings that end in a 1.

Thus  $L(M_2) = \{w \mid w \text{ ends in a } 1\}$ .

**Example:** Consider the finite automaton  $M_3$ .



**FIGURE**  
State diagram of the two-state finite automaton  $M_3$

Machine  $M_3$  is similar to  $M_2$  except for the location of the accept state. As usual, the machine accepts all strings that leave it in an accept state when it has finished reading. Note that because the start state is also an accept state,  $M_3$  accepts the empty string  $\epsilon$ . As soon as a machine begins reading the empty string, it is at the end; so if the start state is an accept state,  $\epsilon$  is accepted. In addition to the empty string, this machine accepts any string ending with a 0.

Here,

$L(M_3) = \{w \mid w \text{ is the empty string } \epsilon \text{ or ends in a } 0\}$ .

**Language accepted by FA:**

String is **accepted by a FA** if and only if the FA starting at the initial state and ends in an accepting state after reading the string.