Lecture 3

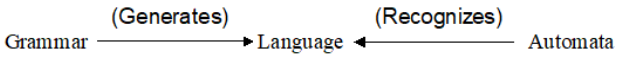
Introduction to Finite Automata

Lecture-3	**PO 1 & PSO 1**

Introduction

An automaton is an algorithm or program that automatically recognizes if a particular string belongs to the language or not, by checking the grammar of the string. In other words, an automaton is an abstract computing device (or machine). Idealized computers are called computational model because real computers used to find the theory to be complex. There are different varieties of such abstract machines (also called models of computation) which can be defined mathematically as: Finite Automaton (FA), Push-Down Automaton (PDA), Linear Bounded Automaton (LBA), Turing Machine (TM) and Post Machine (PM).

Every Automaton fulfills the following basic requirements.



- ☞ Every automaton consists of some essential features as in real computers. It has a mechanism for reading input. The input is assumed to be a sequence of symbols over a given alphabet and is placed on an input tape (or written on an input file). The simpler automata can only read the input one symbol at a time from left to right but not change. Powerful versions can both read (from left to right or right to left) and change the input.
- ☞ The automaton can produce output of some form. If the output in response to an input string is binary (say, accept or reject), then it is called an accepter. If it produces an out- put sequence in response to an input sequence, then it is called a transducer (or automaton with output).
- ☞ The automaton may have a temporary storage, consisting of an unlimited number of cells, each capable of holding a symbol from an alphabet (which may be different from the input alphabet). The automaton can both read and change the contents of

the storage cells in the temporary storage. The accusing capability of this storage varies depending on the type of the storage.

☞ The most important feature of the automaton is its control unit, which can be in any one of a finite number of interval states at any point. It can change state in some de- fined manner determined by a transition function.

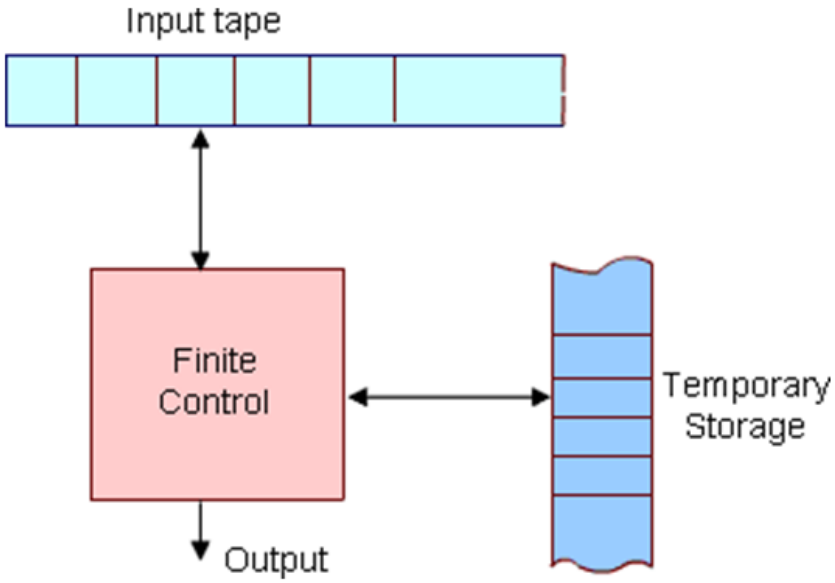


Figure 3.1: Diagrammatic representation of a generic automaton.

At any point of time the automaton is in some integral state and is reading a particular symbol from the input tape by using the mechanism for reading input. In the next time step the automaton then moves to some other integral (or remain in the same state) as defined by the transition function. The transition function is based on the current state, input symbol read, and the content of the temporary storage. At the same time the content of the storage may be changed and the input read may be modified. The automaton may also produce some output during this transition. The internal state, input and the content of storage at any point defines the configuration of the automaton at that point. The transition from one configuration to the next (as defined by the transition function) is called a move. Finite state machine or Finite Automaton is the simplest type of abstract machine we consider. Any system that is at any point of time in one of a finite number of interval state and moves among these states in a defined manner in response to some input, can be modeled by a finite automaton. It does not have any temporary storage and hence a restricted model of computation.

3.1 Finite Automata (FA)

Finite Automata (singular: automaton) are the simplest computational model with limited memory. An entire classification of FA is present in the Figure 3.2 given below. FAs were initially proposed as a simple model for the behavior of neurons. But, nowadays we

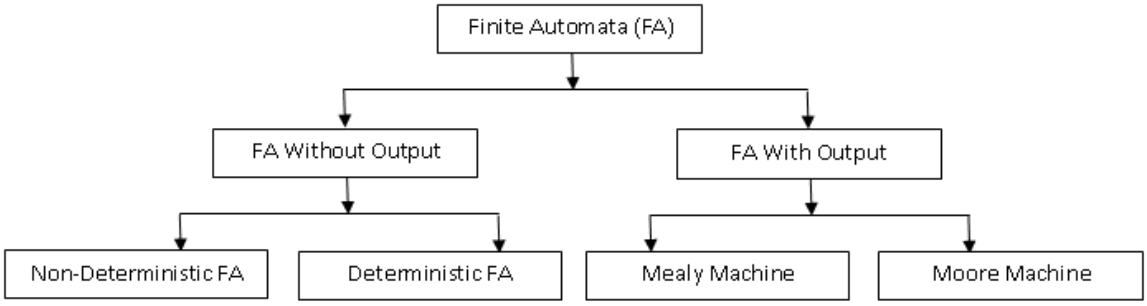


Figure 3.2: Classification of finite automaton.

interact with such computers all the time, as they lie at the heart of various electromechanical devices.

Example 3.1 Here we have presented the automaton of an on-off switch. The controller is in either of two states: "ON" or "OFF," representing the corresponding condition of the switch. Considering the input to the switch to be "PRESS", the switch changes its state each time it receives the input. A finite automaton of an on-off switch is presented in Figure 3.3 given below.

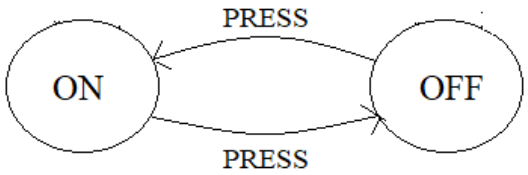


Figure 3.3: Diagram for finite automaton of on-off switch.

Table 3.1: Transition table for automaton of on-off switch.

States	PRESS
ON	OFF
OFF	ON

Another example of this type is the controller for an automatic door. The controller is in either of two states: "OPEN" or "CLOSED," representing the corresponding condition of the door. Depending upon the person(s) standing position on the pad, input conditions are "FRONT", "REAR", "BOTH", and "NEITHER". The controller moves from state to state, depending on the input it receives. Thinking of an automatic door controller as a finite automaton is useful because that suggests standard ways of representation as in Figure 3.4 given below. Hence, we can conclude that, the design of devices like controllers for various household appliances, parts of digital watches and calculators requires keeping the methodology and terminology of finite automata in mind.

We will now take a closer look at finite automata from a mathematical perspective. We will develop a precise definition of a finite automaton, terminology for describing

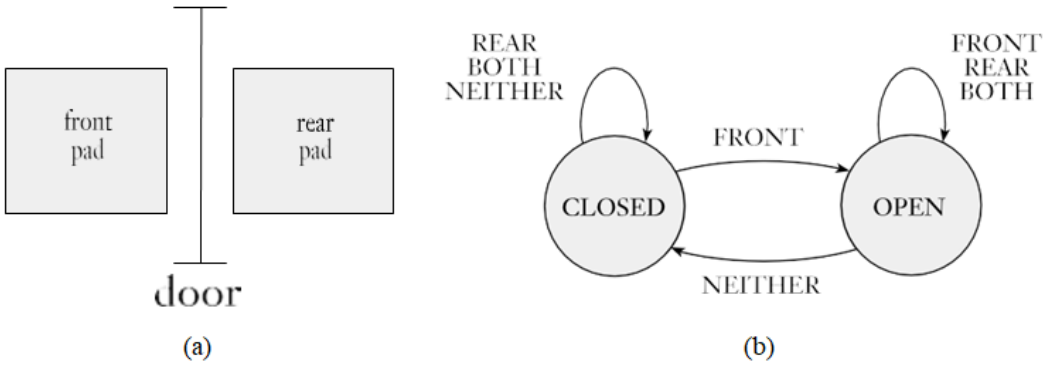


Figure 3.4: Diagram for (a) automatic door representation (b) finite automaton of automatic door.

Table 3.2: Transition table for automaton of automatic door.

States	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

and manipulating finite automata, and theoretical results that describe their power and limitations.

From the previous examples that we have discussed, we got an observation that a finite automaton basically consisting of three major things: States, Input and Transitions. Informally, a state of a system is an instantaneous description of that system which gives all relevant information necessary to determine how the system can evolve from that point on. Input is something that is supplied from outside environment to the state, by getting which a state goes into the transition. Transitions are changes of states that can occur spontaneously or in response to inputs to the states. Though transitions usually take time, we assume that state transitions are instantaneous (which is an abstraction). A system containing only a finite number of states and transitions among them is called a finite-state transition system. Hence a Finite-state transition systems can be modeled abstractly by a mathematical model called finite automaton. The diagram (those we have represented in the examples discussed above) representing a finite automaton is known as state diagram. Before defining the finite automaton let us consider another example with detailed observation.

Example 3.1

The Figure 3.5 showing an automaton is called a state diagram. From the state diagram we observe that:

- ☞ 3 states represented by circles and labelled as q_1, q_2 and q_3.
- ☞ Inputs are 0 and 1.
- ☞ The start state q_1 is represented as an arrow pointing from nowhere.
- ☞ Accept state or final state is represented by a double circle and labelled as q_2.

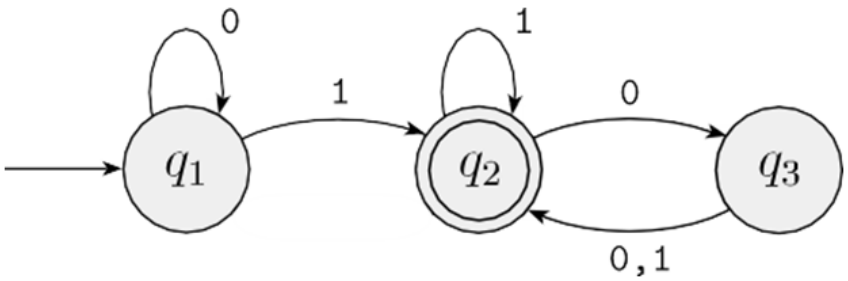


Figure 3.5: Diagram for finite automaton M_1.

☞ Arrows from one state to another are called transitions, which represents the rules to move from one state to another.

3.1.1 Deterministic Finite State Automaton (DFA)

Informally, a DFA (Deterministic Finite State Automaton) is a simple machine that reads an input string, one symbol at a time and then, after the input has been completely read, decides whether to accept or reject the input. As the symbols are read from the tape, the automaton can change its state, to reflect how it reacts to what it has seen so far. A machine for which a deterministic code can be formulated, i.e. if there is only one unique way to change the state on a particular input, then the machine is called deterministic finite automata.

Formal Definition of DFA

A Deterministic Finite Automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

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

We have to remember some points about DFA. In a DFA,

☞ The number of initial state is always 1.

☞ Zero or more number of final states is allowed.

☞ If a finite automaton contains 'n' number of states and 'm'number of input symbols, then the total number of transitions is n×m.

☞ The transition function specifies exactly one next state for each possible combination of a state and an input symbol.

We can describe the automaton in Figure 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

States	0	1
$\rightarrow q_1$	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. $F = \{q_2\}$

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. Considering the input string 1101 into the machine M_1 in Figure 1.4, the processing proceeds as follows:

$\rightarrow$ Start in state q_{13}.

$\rightarrow$ Read 1, follow transition from q_1 to q_2.

$\rightarrow$ Read 1, follow transition from q_2 to q_2.

$\rightarrow$ Read 0, follow transition from q_2 to q_3.

$\rightarrow$ Read 1, follow transition from q_3 to q_2.

$\rightarrow$ Accept because M1 is in an accept state q2 at the end of the input.

From the above, we can analyze that, M_1 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 any string that ends with an even number of 0's following the last 1. Hence, we can conclude that, if $A = \{w \mid w$ contains at least one 1 and an even number of 0's follow the last 1$\}$. Then $L(M_1) = A$, or equivalently, $M1$ recognizes A.

3.1.2 Examples of Deterministic Finite Automata

Example 3.2: In the formal description, M_2 is ($\{q_1, q_2\}$, {0,1}, δ, q_1, $\{q_2\}$). The transition function δ is

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

On the sample string 1101, the machine M_2 starts in its start state q_1 and proceeds first

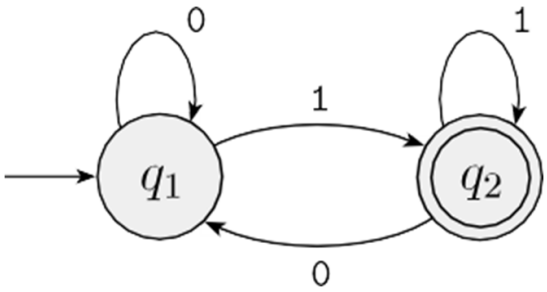


Figure 3.6: Diagram for finite automaton M_2.

to state q_2 after reading the first 1, and then to states q_2, q_1, and q_2 after reading 1, 0, and 1 respectively. Finally 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 will find that M_2 accepts all strings that end in a 1. Thus $L(M_2) = \{w \mid w \text{ ends in a } 1\}$.

Example 3.3:

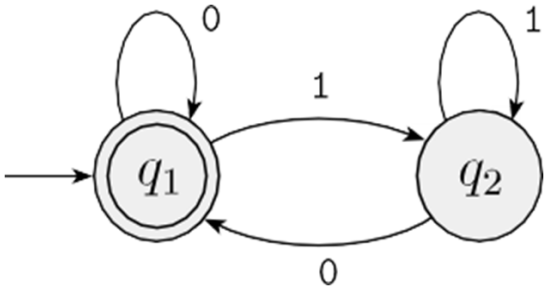


Figure 3.7: Diagram for finite automaton M_3.

Machine M_3 is similar to M_2 except for the location of the accept state. As the start state is also an accept state, M_3 accepts the empty string , and comes to an end; so if the start state is an accept state, is accepted. In addition to the empty string, this machine accepts any string ending with a 0. Hence, $M_3 = \{w \mid w \text{ is the empty string or ends in a } 0\}$.

Example 3.4:

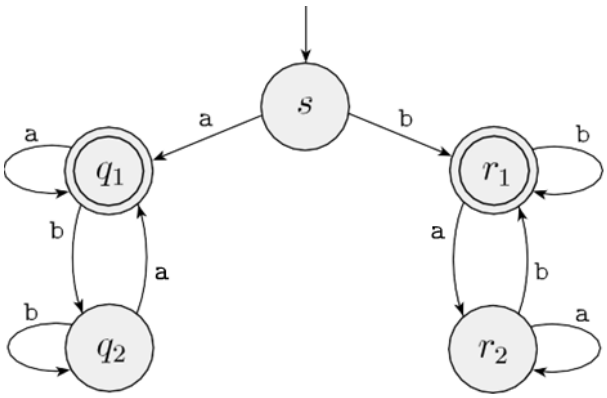


Figure 3.8: Diagram for finite automaton M_4.

In the formal description, M_4 is ({s, q_1, q_2, r_1, r_2}, {a,b}, δ, s, {q_1, r_1}). The transition function δ is

States	a	b
$\rightarrow s$	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

Machine M_4 has two accept states, q_1 and r_1, and operates over the alphabet $\Sigma = \{a, b\}$,. Some experimentation shows that it accepts strings a, b, aa, bb, and bab, but not strings ab, ba, or $bbba$. This machine begins in state s, and after it reads the first symbol in the input, never return to the start state, as it has no way to get from any other state back to s. If the first symbol in the input string is a, then it goes left and accepts when the string ends with an a. Similarly, if the first symbol is a b, the machine goes right and accepts when the string ends in b. So M_4 accepts all strings that start and end with a or that start and end with b. In other words, M_4 accepts strings that start and end with the same symbol.