Automata Theory - Lecture 6

Deterministic and Non-Deterministic Finite Automata

Lecturer: Martha Gichuki

Lecture learning outcomes

At the end of the lecture you will be able to:

(i) Define Finite Automata.

(ii) Describe Formal Systems.

(iii) Differentiate Deterministic and Non-Deterministic Finite Automata

6.1 Finite Automata

A finite-state machine (finite automaton) is a simple, limited model of computation.

It has a number of states and transitions.

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

An automaton is a mathematical model for a Finite State Machine (FSM) that, given an **input of symbols**, "**jumps**" or transitions through a series of states according to a transition function (which can be expressed as a table).

This **transition function** tells the automaton **which state to go to next** given a **current state** and a **current symbol**.

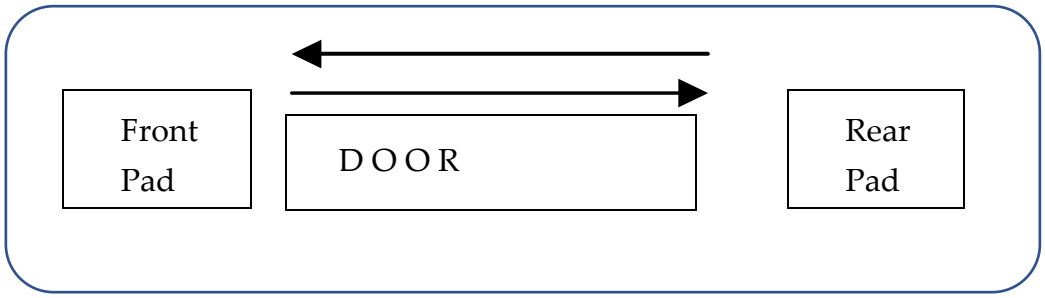
The input is read **symbol by symbol**, until it is consumed completely (think of it as a tape with a word written on it, that is read by a reading head of the automaton; the head moves forward over the tape, reading one symbol at a time). Once the input is depleted, the automaton is said to have stopped. Depending on the **state in which the automaton stops**, it's said that the automaton either **accepts or rejects the input**.

The **set of all the words accepted by an automaton** is called the **language accepted by the automaton.**

Example one: Automatic Door

States: - Door is either open or closed

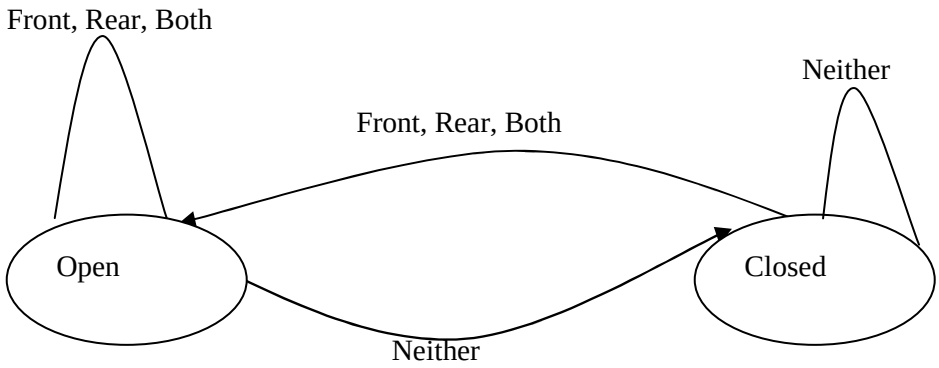
Transitions: - Changes the door from one state to another, based on certain controls like a foot pad that once stepped on, senses that someone is standing nearby. The machine can accept or reject input.



The door behaviour can be represented in a transition table as shown below: -

	Actions (Transitions)			
States	***Front***	***Rear***	***Neither***	***Both***
Closed	Open	Open	Closed	Open
Open	Open	Open	Closed	Open

The same machine can be illustrated using a state diagram as shown below: -



Example Two: Elevator

In an elevator controller a state may represent the floor the elevator is on and the inputs might be the signals received from the buttons. This computer might need several bits to keep track of this information. Controllers for various household appliances such as dish washers and electronic thermostats, as well as parts of digital watches and calculators are additional examples of computers with limited memories. The design of such devices requires keeping the methodology and terminology of finite automata in mind.

FORMAL DEFINITION OF A FINITE AUTOMATON

Other than using state diagrams, we now define Finite automata formally. Though state diagrams are easier to grasp we need the formal definition for two main reasons: -

i). A formal definition is precise – It resolves any uncertainties about what is allowed in a finite automaton.

ii). Good notation helps us think and express our thoughts clearly.

A finite automaton has several parts. It has a start state and a set of accept states, and rules for going from one state to the other, depending on the input symbol. It has an input alphabet that indicates the allowed input symbols.

The formal definition says that a finite automaton is a list of those five objects: In mathematical language a list of five elements is often called a 5-tuple. Hence, we define a finite automaton to be a 5-tuple consisting of five parts.

i). set of states,

ii). input alphabet,

iii). rules for moving,

iv). start state and

v). Accept states – sometimes called final states

We use the transition function, denoted as δ to define the rules for moving. If the finite automaton has an arrow from a state X to a state Y labeled with the input symbol 1, that means that, if the automaton is in state X when it reads a 1, it then moves to state Y. We can indicate the same thing with the transition function by saying that δ (X, 1) = Y. This notation is a kind of mathematical shorthand. Putting it all together we arrive at the formal definition of finite automata.

Types of Automatons

1. Deterministic Finite Automata (DFA)

This is an automaton in which each move (transition from one state to another) is uniquely determined by the current configuration. If the internal state, input and contents of the storage are known, it is possible to predict the future behaviour of the automaton. This is said to be Deterministic Finite Automaton (DFA), otherwise it is Non-Deterministic Finite Automaton (NFA).

Formal Definition of a Deterministic Finite Automaton (DFA)

An automaton whose output response is "Yes" or "No" is called an acceptor. The formal description of a DFA is a five-tuple M= (Q, ∑, δ, q_0, F); where: -

Q = Finite set of Internal States

∑= Finite Set of Symbols "Input Alphabet"

δ: Q X ∑→Q is the Transition function

q_0∈Q is the Initial State

F ⊆ Q is the Set of Final States

The input mechanism can move from left to right and reads exactly one symbol on each step.

The transition from one internal state to another is governed by the transition function (δ).

For example, if δ (q_0, a) = q_1, this means that if the DFA is in state "q_0" and the current input symbol is "a", then the DFA will go into state q_1.

The start state is normally represented using an Oval shape that has an arrow pointing into it from nowhere.

The Final state is represented using a double circle/oval.

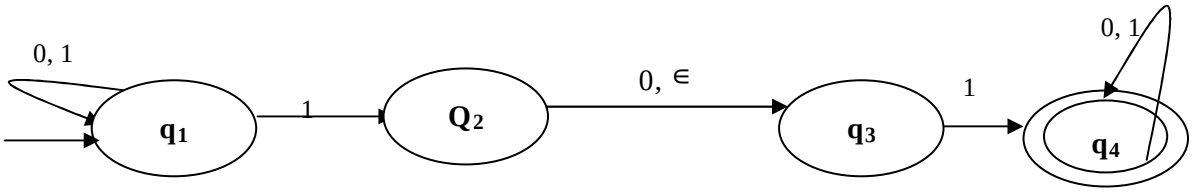
q_0= {S} F = {q1, r1}

It is important to note that for a DFA, exactly one transition arrow exits every state for each possible input alphabet.

2. Non-deterministic Finite Automata (NFA)

Non-determinism is a useful concept that has had great impact on the theory of computation. So far in our discussion, every step of a computation follows in a unique way from the preceding step. When the machine is in a given state and reads the next input symbol, we know what the next state will be – it is determined. We call this deterministic computation. In a non-deterministic machine, several choices may exist for the next state at any point and the machine therefore may have additional features.

Example: - Consider the NFA machine N1 below: -



From this state diagram we can come up with a table showing the differences between DFA and NFA as follows: -

DFA	NFA
Every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet.	A state may have zero, one or many exiting arrows for each alphabet symbol.
Labels on the transition arrows are symbols from the alphabet.	May have arrows labeled with members of the alphabet or∈. Zero, one, or many arrows may exit from each state with the label ∈.
The transition function takes a state and an input symbol to produce the next state.	The transition function takes a state and input symbol or the empty string∈, to produce the set of possible next states.

Formal Definition of a Nondeterministic Finite Automaton (NFA)

The formal definition of a nondeterministic finite automaton is similar to that of a deterministic finite automaton. Both have states, an input alphabet, a transition function, a start state and a collection of accept states. However, they differ in one essential way: in the type of transition function. For the NFA, the function produces a set of possible next states. (As mentioned earlier in the table). For any set Q we write P(Q) to be the collection of all subsets of Q. Here P(Q) is called the power set of Q. For any alphabet $\sum$ we write $\sum_{\in}$ to be $\sum \cup \{\in\}$. Now we can easily write the formal description of the type of the transition function in an NFA. It is $\delta: Q \times \sum_{\in} \rightarrow P(Q)$.

A Nondeterministic Finite Automaton is a 5-tuple M= $(Q, \sum, \delta, q_0, F)$; where: -

Q is a Finite set of Internal States

$\sum$ is a Finite Set of Symbols "Input Alphabet"

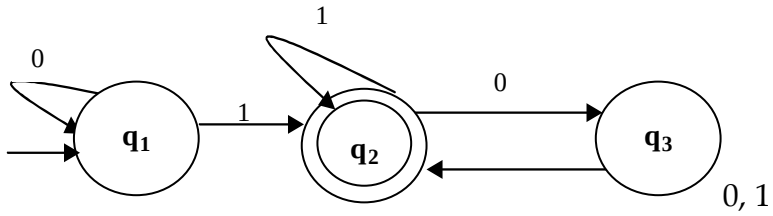
$\delta: Q \times \sum_{\in} \rightarrow P(Q)$ is the transition Symbol $q_0 \in Q$ is the initial State

$F \subseteq Q$ is the Set of Final/accept States

Recall that the formal definition precisely describes what we mean by a finite automaton.

Review Questions:

a) Using formal notation, briefly describe the operation of a deterministic finite automaton
b) Given the state diagram below for a finite automaton M_5, describe the machine formally: -



Solution:

M= (Q, ∑, δ, q_0, F), Where

Q = {q1, q2, q3}

∑ = {0,1}

δ is the transition function given by the table?

	Input Alphabet	
States	0	1
q1	q1	q2
q2	q3	q2
q3	q2	q2

q_0 = {q1}

F = {q2}

References

1 Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012).

2 Introduction to languages and the theory of computation (3rd ed.), Martin, J., New York: McGraw-Hill. ISBN-13: 978-0072322002, (2002)