



Bangladesh Govt. & UGC Approved

UNIVERSITY OF GLOBAL VILLAGE (UGV), BARISHAL.

THE FIRST SKILL BASED HI-TECH UNIVERSITY IN BANGLADESH

Theory of Computation

Md. Mahadi Hasan Shaon
Lecturer, UGV

Course Logistics

Marking Distribution

- Quiz 10%
- Attendance 10%
- Assignment & performance 10%
- Lab Exam 20%
- **Total 50%**

Final Grade/ Grand Total

Midterm:	20%
Final Term:	30%
Grand Total	50%

Symbol

Symbol: Is a basic building block of Theory of Computation.

e.g. a,b,...z(Latters)
 0,1,...9(Digit)

Alphabet

Alphabet: Is a finite set of symbols.

Σ (Sigma)

e.g. $\Sigma = \{a, b\}$

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

$\Sigma = \{0, 1, \dots, 9\}$

$\Sigma = \{a, b, c\}$

This all are finite set

String

String: Is a finite sequence of symbol.

$\mathbb{W}(\text{String})$

e.g. $\mathbb{W}=\{0,1\}$

$\mathbb{W}=0110$

$\mathbb{W}=1010$

Length of String

Length of String: $|W|$

$$\Sigma = \{a,b\}$$

$$W = ababba = 6$$

$$|W| = 6 \text{ (length is 6)}$$

Empty String

Empty String: ϵ (Epsilon) or λ (Lambda)

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

0 is a string over the Σ of length 1

10 is a string over the Σ of length 2

101 is a string over the Σ of length 3

e.g. ($\emptyset / \{\}$ = Empty Set)

Language

Language: Is collection of strings.

(It can be finite/ infinite)

e.g. $\Sigma = \{a, b\}$

L1= Set of all strings over Σ of length 2
= $\{aa, ab, ba, bb\}$ **Finite Set**

L2= Set of all strings over Σ of length 3
= $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ **Finite Set**

L3= Set of all strings over Σ where each string starts with 'a'
= $\{a, aa, ab, aaa, aba, aaaa, \dots\}$ **Infinite set**

Power Of Σ

$$\Sigma = \{a, b\}$$

$$\Sigma^1 = \text{Set of all strings over this } \Sigma \text{ of length 1} \\ = \{a, b\}$$

$$\Sigma^2 = \text{Set of all strings over this } \Sigma \text{ of length 2} \\ = \{aa, ab, ba, bb\}$$

$$\Sigma^3 = \text{Set of all strings over this } \Sigma \text{ of length 3} \\ = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

$$\Sigma^0 = \text{Set of all strings over this } \Sigma \text{ of length 0} \\ = \{\epsilon\}$$

$$\underline{\Sigma^*}$$

Σ^* = Set of all possible strings.

$$\begin{aligned}\Sigma^* &= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \\ &= \{\epsilon\} \cup \{a,b\} \cup \{aa,ab,ba,bb\} \dots\end{aligned}$$

Previous,

$$L1 \subseteq \Sigma^*$$

$$L2 \subseteq \Sigma^*$$

$$L3 \subseteq \Sigma^*$$

So, all language is subset of Σ^*

e.g. \subseteq = Subset

Set

Set: is a collection of objects.

$$S = \{a, b, c, h, d\}$$

$$S = \{1, 2, 5, 6\}$$

Set

1. Empty set $S = \emptyset / \{\}$
2. Not Empty Set $S \neq \emptyset$

Not Empty Set

1. Finite Set
2. Infinite Set

Finite Automata

- We will use several different models, depending on the features we want to focus on. Begin with the simplest model, called the finite automaton.
- Good models for computer with an extremely limited amount of memory. For example, various household appliances such as dishwashers and electronic thermostats, as well as parts of digital watches and calculators.
- The design of such devices requires keeping the methodology and terminology of finite automata in mind.
- Next we will analyze an example to get an idea.

Finite State Machine

- The finite state machine represents a mathematical model of a system with certain input.
- The model finally gives certain output.
- The input is processed by various states, these states are called as intermediate states.
- The finite state system is very good design tool for the programs such as `TEXT EDITORS` and `LEXICAL ANALYZERS`.

Definition of Finite Automata

A finite automata is a collection of 5-tuple $(Q, \Sigma, \delta, q_0, F)$ Where,

Q is finite set of states, which is nonempty.

Σ is input alphabet, indicates input set.

δ is transition function or mapping function. We can determine the next state using this function.

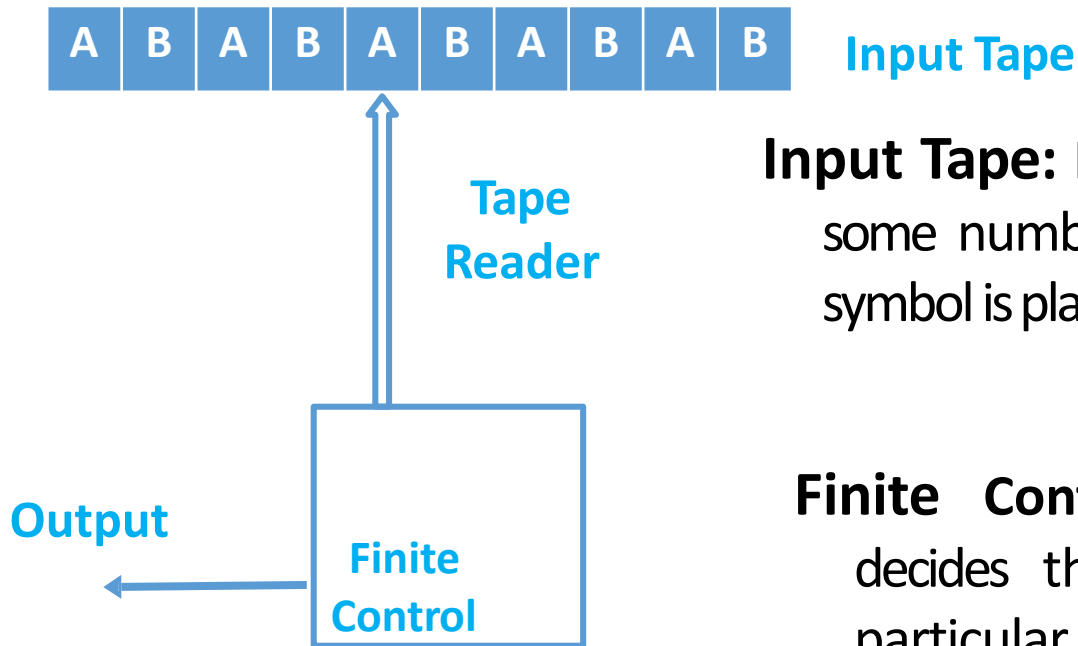
q_0 is an initial state and is in Q

F is set of final states.

$\delta = \text{Delta}$

Finite Automata Model

- The finite automata can be represented as follows:



Input Tape: It is a linear tape having some number of cells. Each input symbol is placed in each cell.

Finite Control: The finite control decides the next state on receiving particular input from input tape.

Example

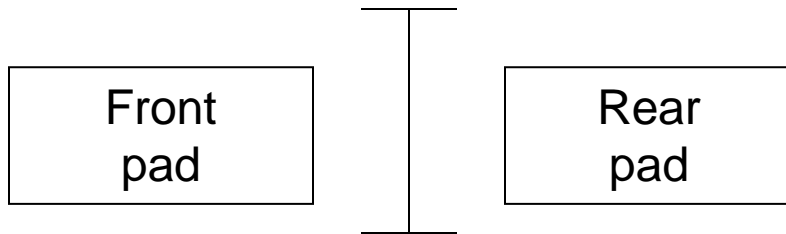


Figure: Top view of an automatic door

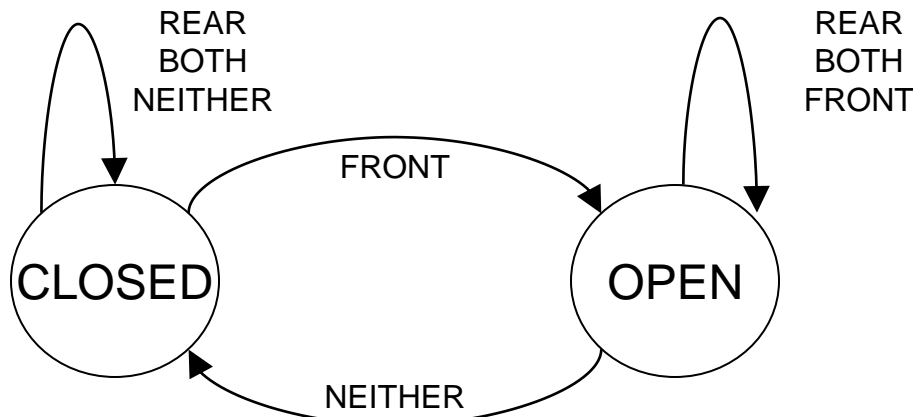


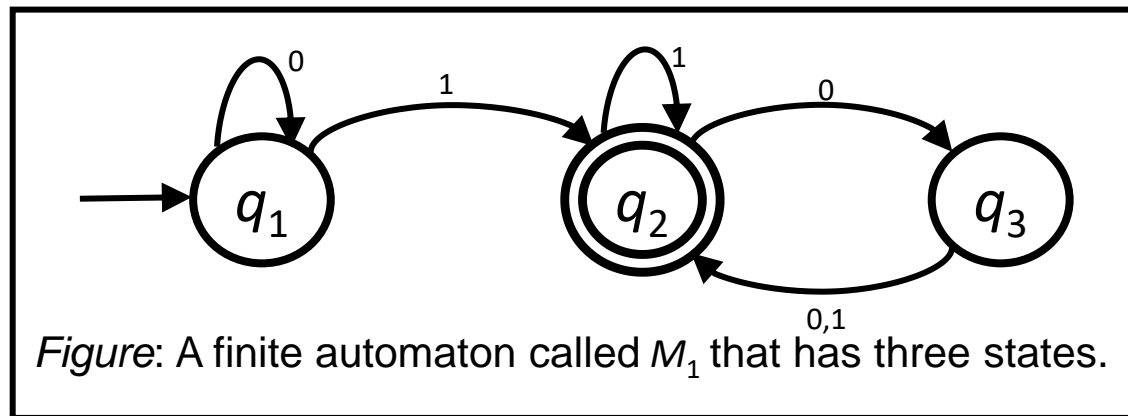
Figure: State diagram for Automatic door controller

		Input Signal			
		NEITHER	FRONT	REAR	BOTH
State	CLOSED	CLOSED	OPEN	CLOSED	CLOSED
	OPEN	CLOSED	OPEN	OPEN	OPEN

Figure: State Transition table for automatic door controller

- ❑ Automatic doors swing open when sensing 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 long enough for the person to pass all the way through.
 - ❑ The door does not strike someone standing behind it as it opens.

Terminology



- The above figure is called *state diagram* of M_1 .
- It has three *states*, labeled q_1 , q_2 , and q_3 .
- The *start state* is q_1 , indicated by the arrow pointing at it from no where.
- The *accept state*, q_2 , is the one with a double circle.
- The arrow going from one state from another are called *transitions*.
- The symbol(s) along the transition is called *label*.

M_1 works as follows –

The automaton receives the symbols from the input string one by one from left to right.

After reading each symbol, M_1 moves from one state to another along the transition that has the symbol as its label.

When it reads the last symbol, M_1 produces the output.

The output is ACCEPT if M_1 is now in an accept state and REJECT if it is not.

Simulation

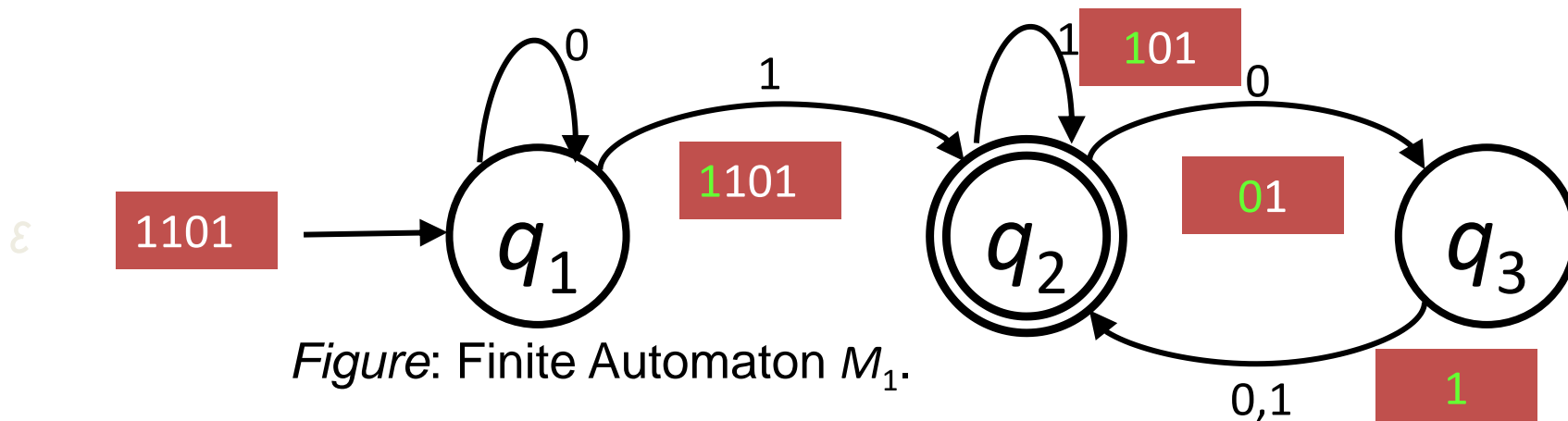


Figure: Finite Automaton M_1 .

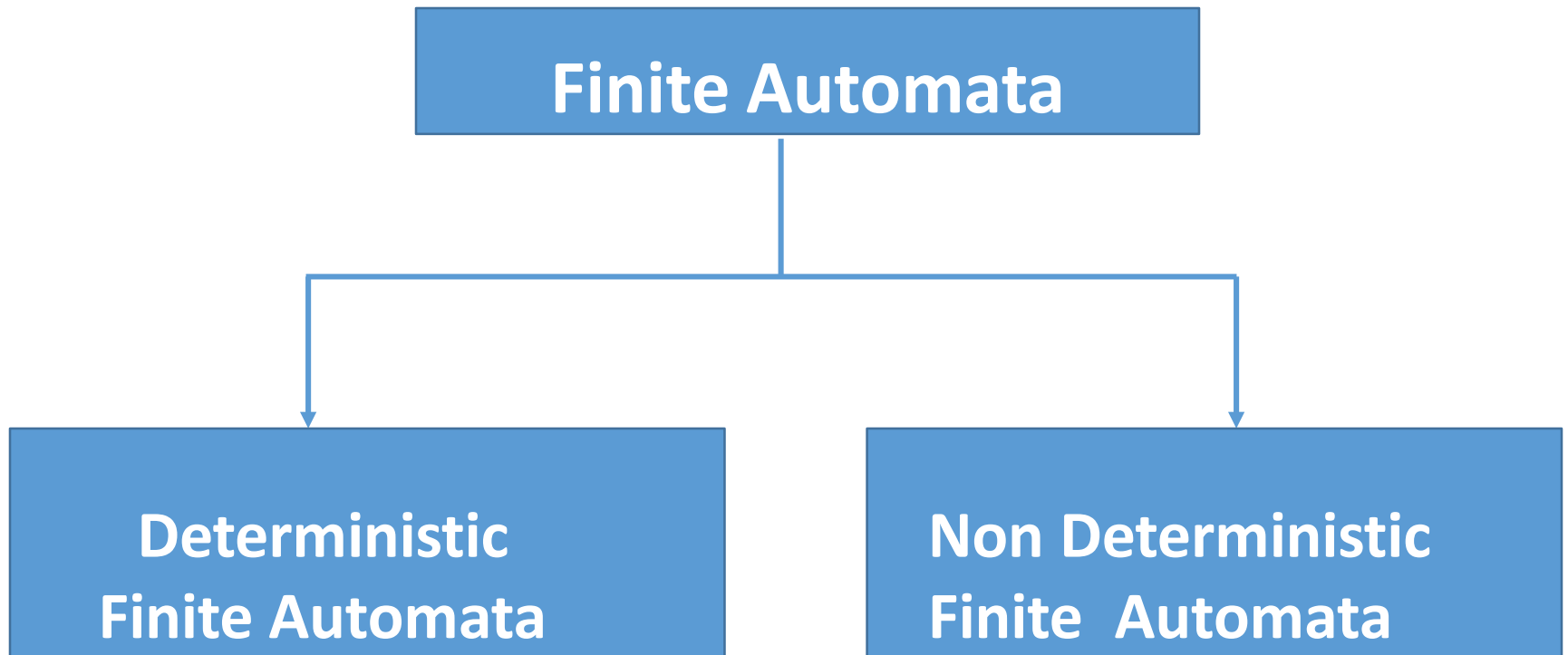
After feeding the input string 1101 to the above machine, the processing proceeds as follows –

Start in state q_1 ;

- Read 1, follow transition from q_1 to q_2 ;
- Read 1, follow transition from q_2 to q_2 ;
- Read 0, follow transition from q_2 to q_3 ;
- Read 1, follow transition from q_3 to q_2 ;

Accept, as the machine M_1 is in an accept state q_2 at the end of the input string.

Types of Automata



Types of Automata

- **Deterministic Finite Automata:** The Finite Automata is called Deterministic Finite Automata if there is only one path for a specific input from current state to next state.

It can be represented as follows:

- A machine $M = (Q, \Sigma, \delta, q_0, F)$ Where ,
Q is finite set of states, which is non empty.
 Σ is input alphabet, indicates input set.
 δ is transition function or mapping function. We can determine the next state using this function.
 q_0 is an initial state and is in Q
F is set of final states.

Where $\delta: Q \times \Sigma \rightarrow Q$

Types of Automata

- **Non Deterministic Finite Automata:** The Finite Automata is called Non Deterministic Finite Automata if there are more than one path for a specific input from current state to next state.

It can be represented as follows:

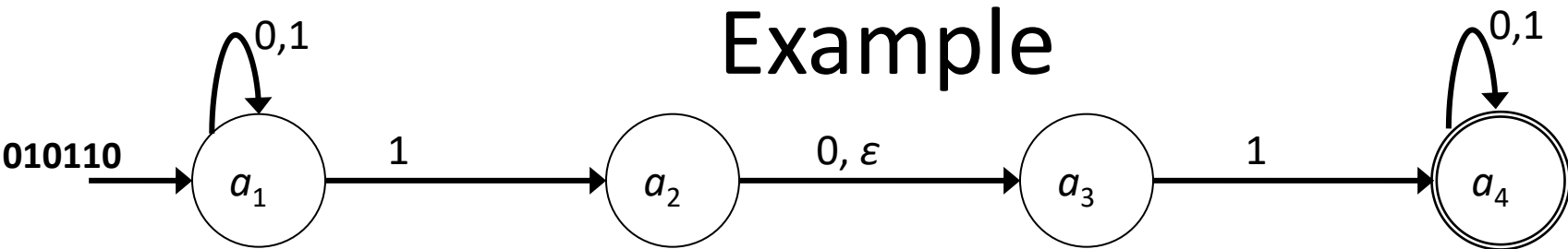
- A machine $M = (Q, \Sigma, \delta, q_0, F)$ Where ,
Q is finite set of states, which is non empty.
 Σ is input alphabet, indicates input set.
 δ is transition function or mapping function. We can determine the next state using this function.
 q_0 is an initial state and is in Q
F is set of final states.

Where $\delta: Q \times \Sigma \rightarrow 2^Q$

Difference between DFA & NFA

Deterministic Finite Automata	Non Deterministic Finite Automata
For Every symbol of the alphabet, there is only one state transition in DFA.	We do not need to specify how does the NFA react according to some symbol.
DFA cannot use Empty String transition.	NFA can use Empty String transition.
DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
DFA will reject the string if it end at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
Backtracking is allowed in DFA.	Backtracking is not always allowed in NFA.
DFA is more difficult to construct.	NFA is easier to construct.

Example



Symbol read

0 -----

1 -----

0 -----

1 -----

1 -----

0 -----

