Catalog

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%

Theory of Computation

- The ***Theory of Computation*** is the branch of computer science that deals with how efficiently problems can be solved on a model of computation, using an algorithm.
- The field is divided into three major branches:
- Automata theory and language
- Computability theory
- Complexity theory

Complexity theory

- The main question asked in this area is “What makes some problems computationally *hard* and other problems *easy*?”
- A problem is called “easy”, if it is efficiently solvable.

Examples of “easy” problems are (i) sorting a sequence of, say, 1,000,000 numbers, (ii) searching for a name in a telephone directory.

- A problem is called “hard”, if it cannot be solved efficiently, or if we don’t know whether it can be solved efficiently.

Examples of “hard” problems are (i) factoring a 300-digit integer into its prime factors.

Central Question in *Complexity Theory*: Classify problems according to their degree of “difficulty”. Give a proof that problems that seem to be “hard” are really “hard”.

Computability Theory

- Computability theory In the 1930's, G¨odel, Turing, and Church discovered that some of the fundamental mathematical problems cannot be solved by a "computer".
- To attack such a problem, we need formal definitions of the notions of *computer*, *algorithm*, and *computation*.
- The *theoretical model*s that were proposed in order to understand *solvable* and *unsolvable* problems led to the development of real computers.

Central Question in *Computability Theory*: Classify problems as being solvable or unsolvable.

Automata theory

- Automata Theory deals with definitions and properties of different types of "*computation models*". Examples of such models :
- *Finite Automata* :These are used in text processing, compilers, and hardware design.
- *Context-Free Grammars:* These are used to define programming languages and in Artificial Intelligence.
- *Turing Machines:* These form a simple abstract model of a "real" computer, such as your PC at home.

Central Question in *Automata Theory*: Do these models have the same power, or can one model solve more problems than the other?.

Theory of Computation

Purpose and motivation :

- What are the mathematical properties of computer hardware and software ?
- What is a *computation* and what is an *algorithm*? Can we give mathematical definitions of these notions?
- What are the *limitations* of computers? Can "everything" be computed?

Purpose of the TOC: Develop formal mathematical models of computation that reflect real-world computers.

Theory of Computation

Purpose and motivation :

- What are the mathematical properties of computer hardware and software ?
- What is a *computation* and what is an *algorithm*? Can we give mathematical definitions of these notions?
- What are the *limitations* of computers? Can "everything" be computed?

Purpose of the TOC: Develop formal mathematical models of computation that reflect real-world computers.

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. Σ={a,b}

Σ={0,1}

Σ={0,1,….9}

Σ={a,b,c}

This all are finite set

String

String: Is a finite sequence of symbol.

ɯ(String)

e.g. ɯ={0,1}

ɯ=0110

ɯ=1010

Length of String

Length of String: |ɯ|

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

ɯ = ababba = 6

| ɯ | = 6 (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. (Ø/{} = 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,…..} **Infinite set**

Power Of Σ

Σ={a,b}

Σ1 = Set of all strings over this Σ of length 1

={a,b}

Σ2 = Set of all strings over this Σ of length 2

={aa,ab,ba,bb}

Σ3 = Set of all strings over this Σ of length 3

={aaa,aab,aba,abb,baa,bab,bba,bbb}

Σ° = Set of all strings over this Σ of length 0

={ε}

Σ*

Σ* = **Set of all possible strings**.

$$\Sigma^* = \Sigma^\circ \cup \Sigma 1 \cup \Sigma 2 \cup \Sigma 3$$

$$=\{\varepsilon\} \cup \{a,b\} \cup \{aa,ab,ba,bb\} \ldots\ldots$$

Previous,

$$L1 \subseteq \Sigma^*$$

$$L2 \subseteq \Sigma^*$$

$$L3 \subseteq \Sigma^*$$

So, all language is subset of **Σ***

e.g. ⊆ = Subset

Set

Set: **is a collection of objects.**

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

S={1,2,5,6}

Set

1. Empty set S = ∅/{}
2. Not Empty Set S ≠ ∅

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,Σ,δ,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.

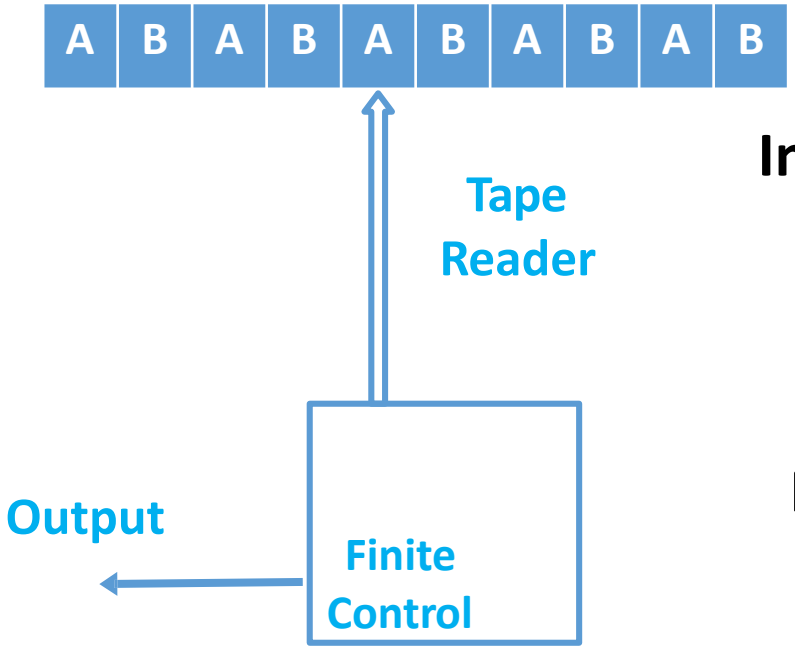
q_0 is an initial state and is in Q

F is set of final states.

δ = 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

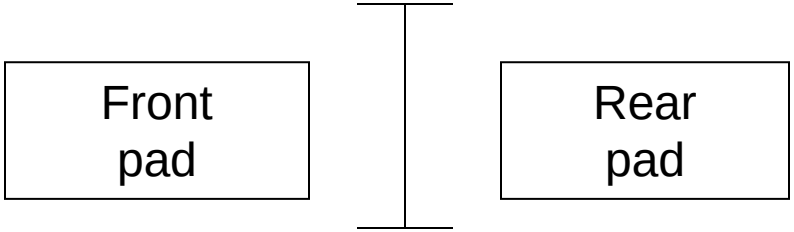


Figure: Top view of an automatic door

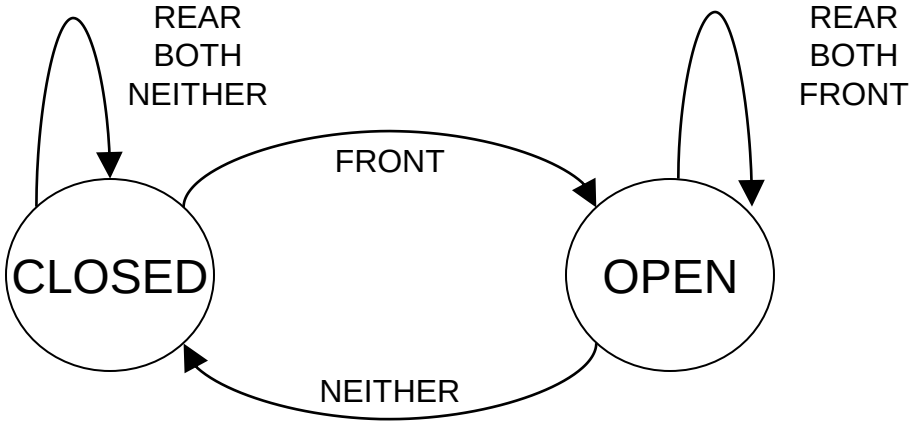


Figure: State diagram for Automatic door controller

		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

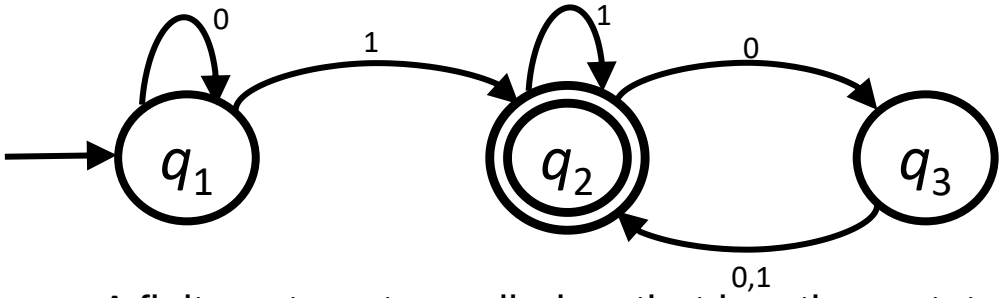


Figure: A finite automaton called M_1 that has three states.

- 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

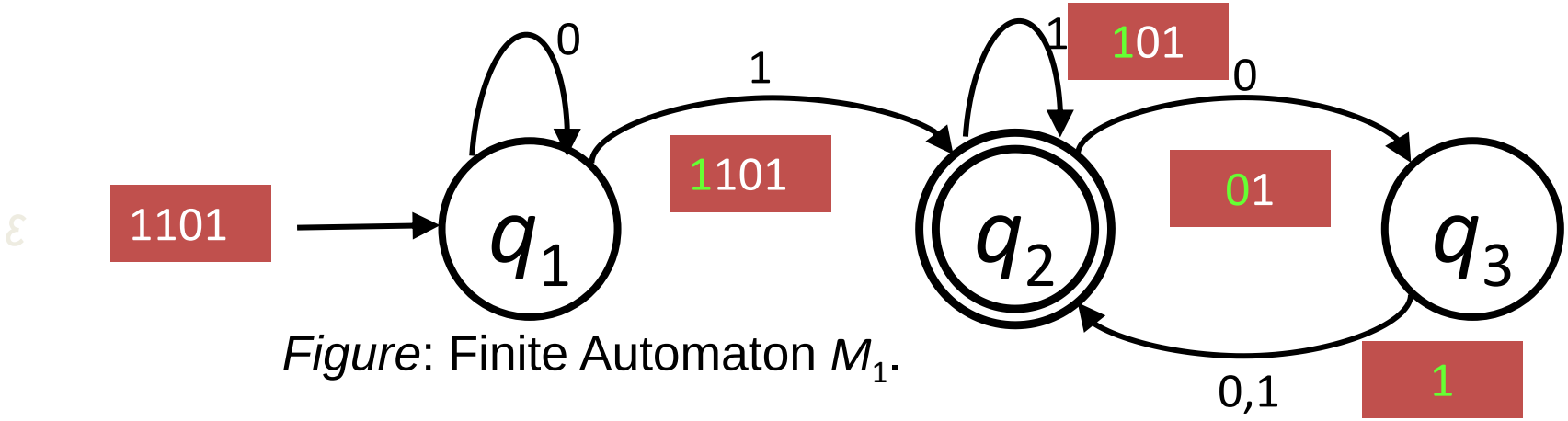


Figure: Finite Automaton M_1.

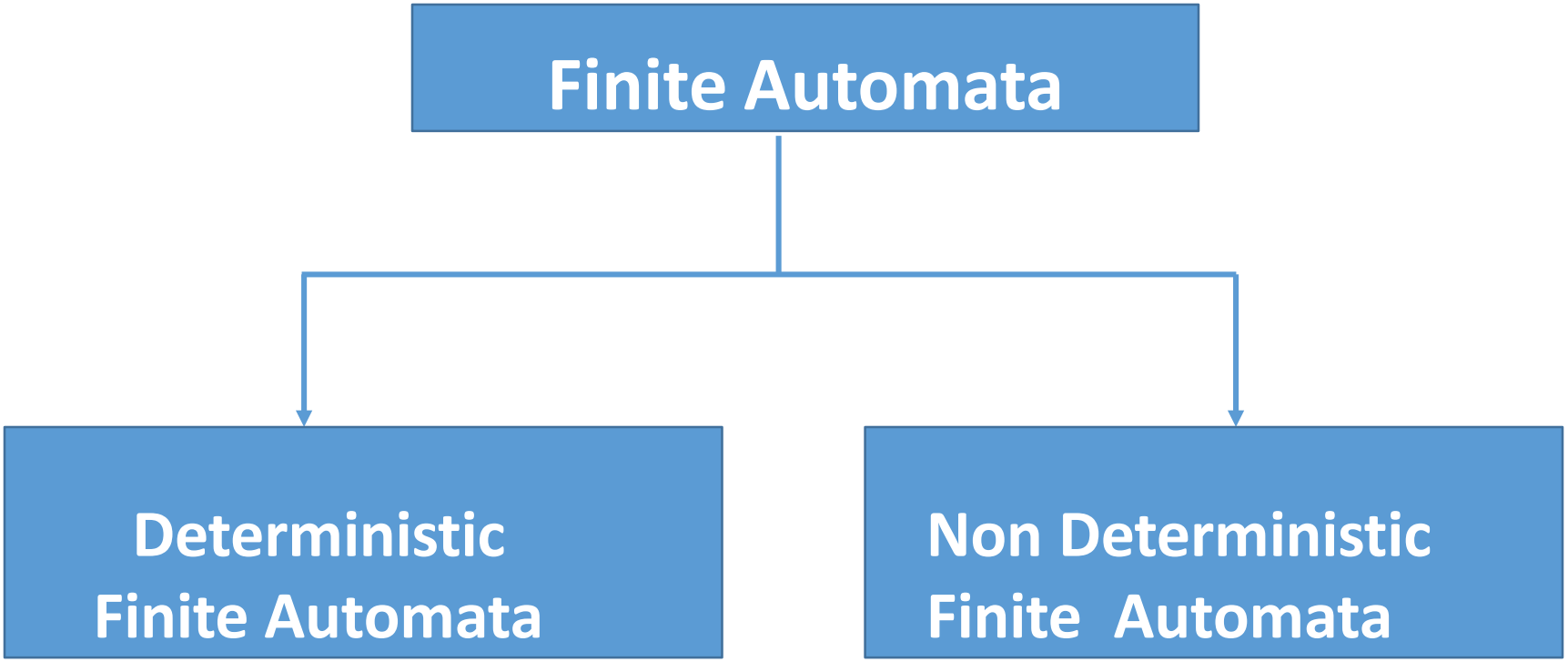
After feeding the input string 1101 to the above machine, the processing proceeds as follows –

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,Σ,δ,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 δ**:Q X** Σ **-> 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,Σ,δ,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

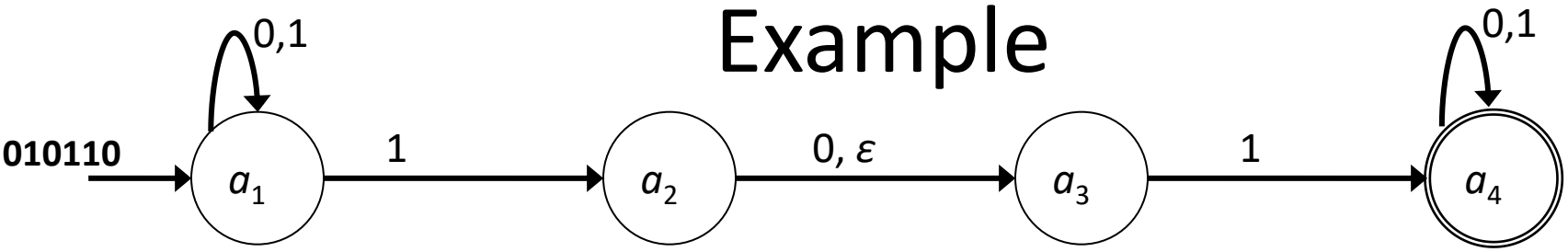
  F is set of final states.

Where δ**:Q X** Σ **->** 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 ----------

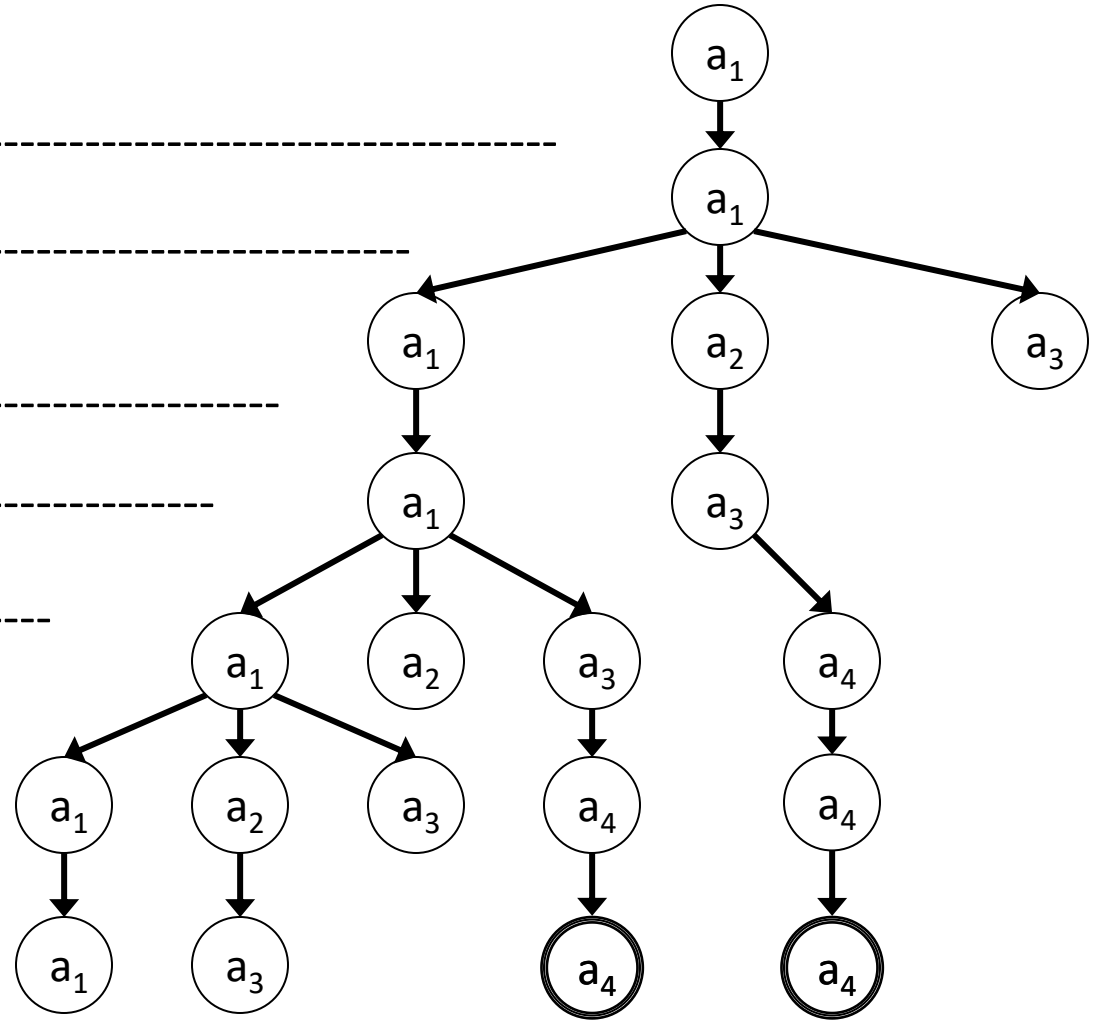
1 ----------

0 ----------

1 ----------

1 ----------

0 ----------



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

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. Σ={a,b}

Σ={0,1}

Σ={0,1,….9}

Σ={a,b,c}

This all are finite set

String

String: Is a finite sequence of symbol.

ɯ(String)

e.g. ɯ={0,1}

ɯ=0110

ɯ=1010

Length of String

Length of String: |ɯ|

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

ɯ = ababba = 6

| ɯ | = 6 (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. (Ø/{} = 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,…..} **Infinite set**

Power Of Σ

Σ={a,b}

Σ1 = Set of all strings over this Σ of length 1

={a,b}

Σ2 = Set of all strings over this Σ of length 2

={aa,ab,ba,bb}

Σ3 = Set of all strings over this Σ of length 3

={aaa,aab,aba,abb,baa,bab,bba,bbb}

Σ° = Set of all strings over this Σ of length 0

={ε}

Σ*

Σ* = **Set of all possible strings**.

$$\Sigma^* = \Sigma^\circ \cup \Sigma 1 \cup \Sigma 2 \cup \Sigma 3$$

$$=\{\varepsilon\} \cup \{a,b\} \cup \{aa,ab,ba,bb\} \ldots\ldots$$

Previous,

$$L1 \subseteq \Sigma^*$$

$$L2 \subseteq \Sigma^*$$

$$L3 \subseteq \Sigma^*$$

So, all language is subset of **Σ***

e.g. ⊆ = Subset

Set

Set: **is a collection of objects.**

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

S={1,2,5,6}

Set

1. Empty set S = ∅/{}
2. Not Empty Set S ≠ ∅

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.

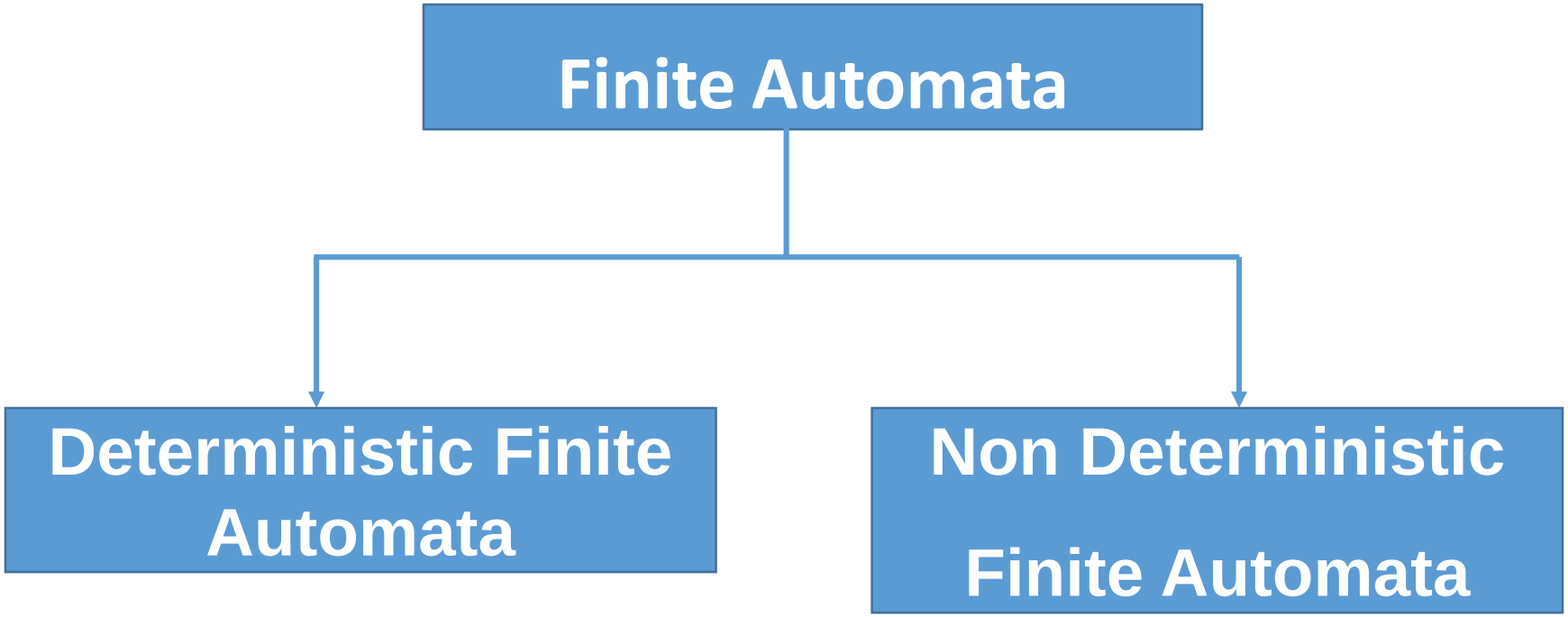
Bangladesh Govt. & UGC Approved
UNIVERSITY OF GLOBAL VILLAGE (UGV), BARISHAL.
THE FIRST SKILL BASED HI-TECH UNIVERSITY IN BANGLADESH

Theory of Computation

Md. Mahadi Hasan Shaon
Lecturer, UGV

Deterministic and Nondeterministic Computation

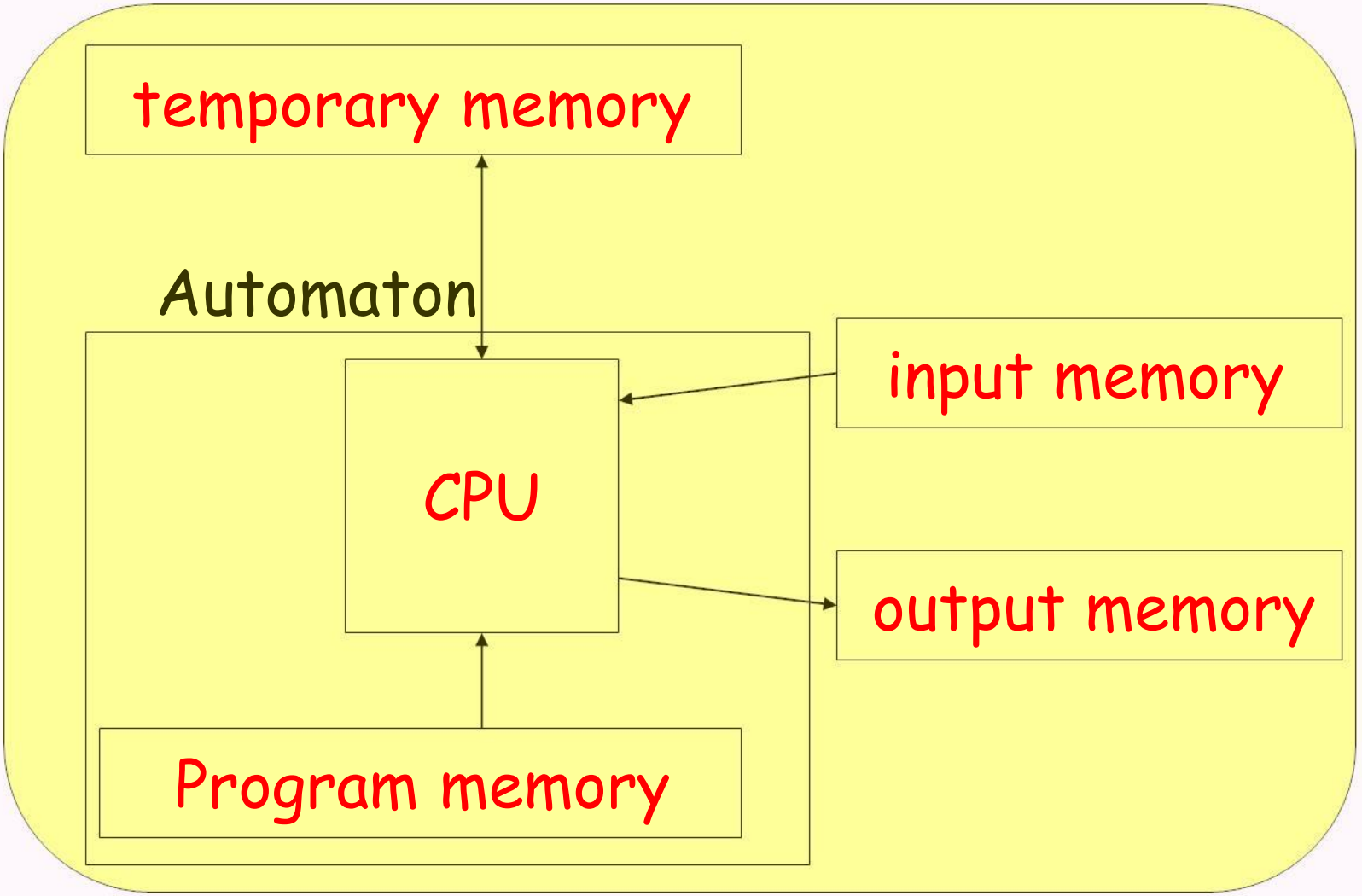
Types of Automata



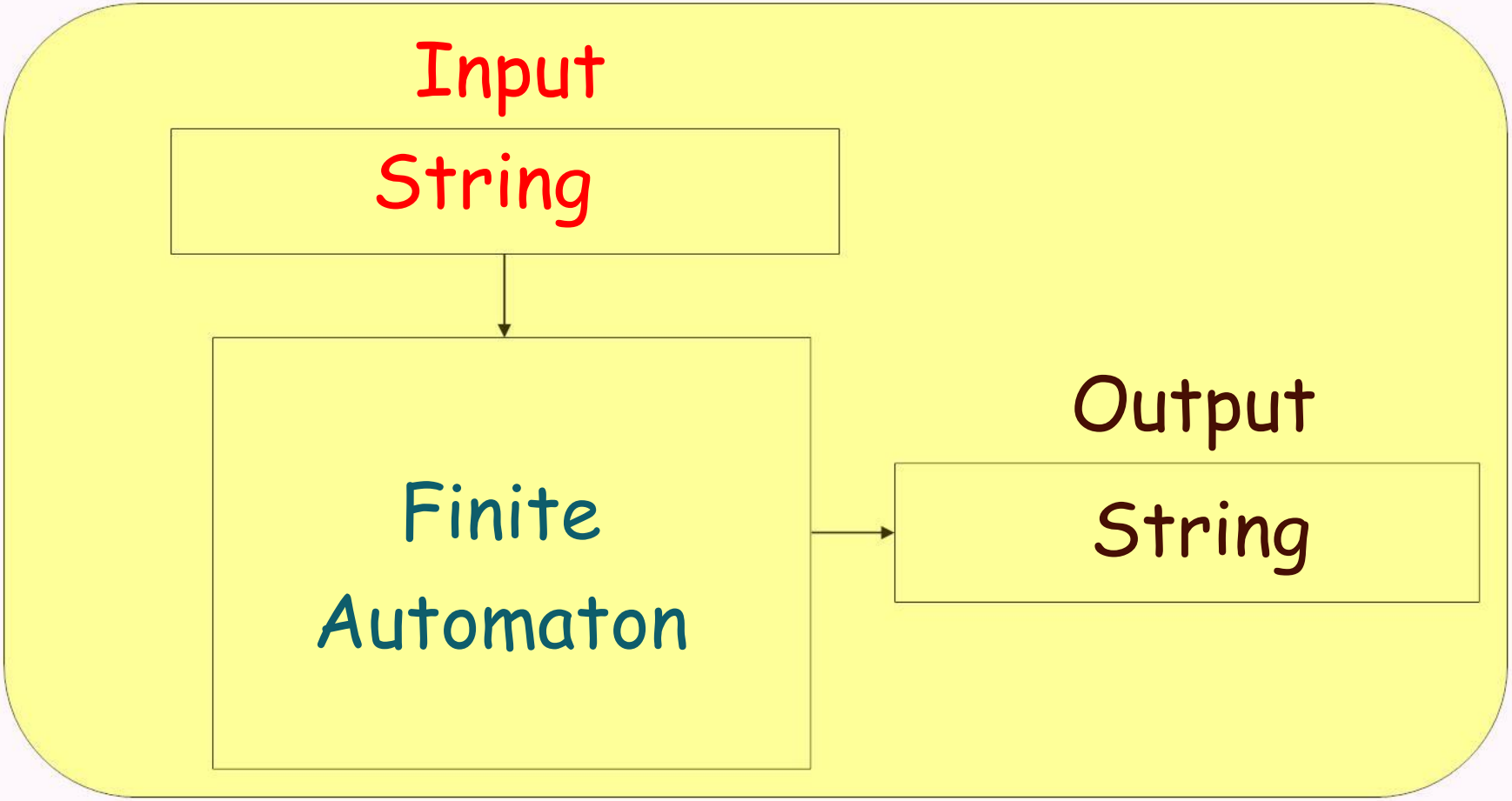
Difference between DFA & NFA

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.

Automaton


temporary memory
Automaton
CPU
input memory
output memory
Program memory

Finite Automaton



DFA

- DFA: Deterministic Finite Automaton.
- 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 the next state will be – it is determined.
- We call this deterministic computation.

Formal Definition of a DFA

- A DFA is a five-tuple:

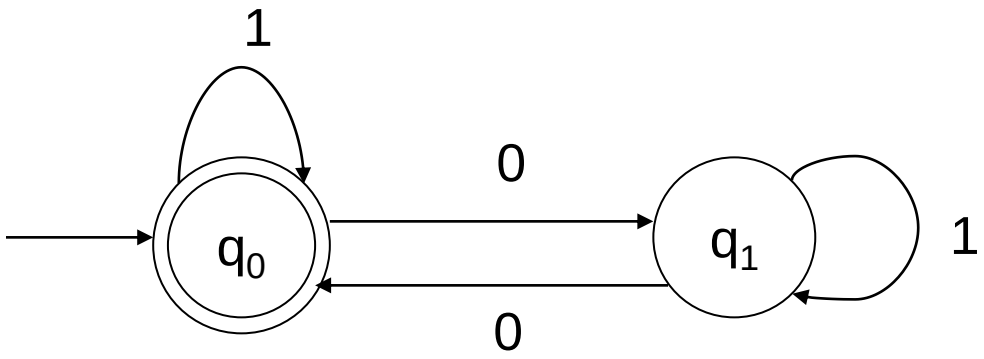
  $M = (Q, \Sigma, \delta, q_0, F)$

Q	A finite set of states
Σ	A finite input alphabet
q_0	The initial/starting state, q_0 is in Q
F	A set of final/accepting states, which is a subset of Q
δ	A transition function, which is a total function from Q x Σ to Q

$\delta: (Q \times \Sigma) \rightarrow Q$	δ is defined for any q in Q and s in Σ, and
$\delta(q,s) = q'$	is equal to some state q' in Q, could be q'=q

  Intuitively, δ(q,s) is the state entered by M after reading symbol s while in state q.

- The finite control can be described by a transition diagram or table:



1	0	0	1	1	
q_0	q_0	q_1	q_0	q_0	q_0 accpeted

- One state is final/accepting, all others are rejecting.
- The above DFA accepts those strings that contain an even number of 0's, including the *null* string, over *Sigma* = {0,1}

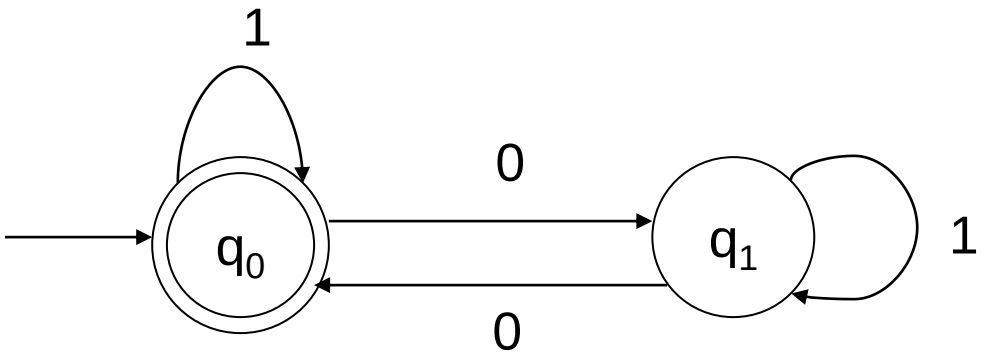
  L = {all strings with zero or more 0's}
- Note, the DFA must reject all other strings

$Q = \{q_0, q_1\}$

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

Start state is q_0

$F = \{q_0\}$



δ:

	0	1
q_0	q_1	q_0
q_1	q_0	q_1

Nondeterministic Finite State Automata (NFA)

- An NFA is a five-tuple:

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

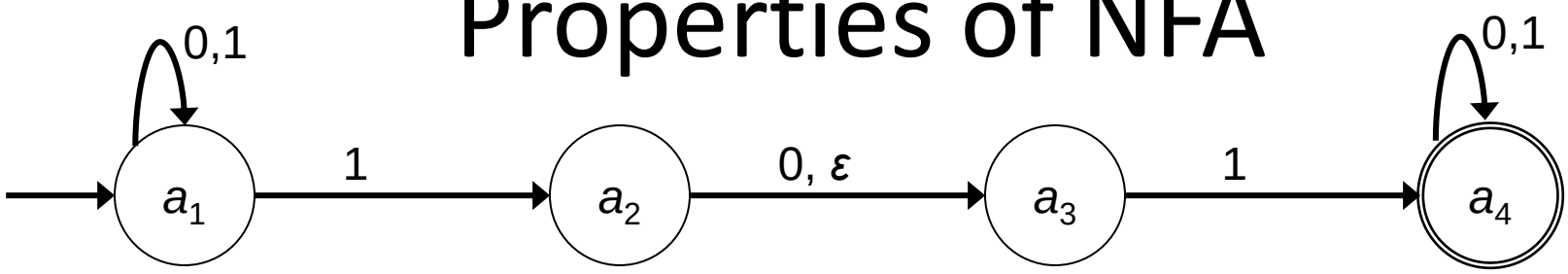
Q	A finite set of states
Σ	A finite input alphabet
q_0	The initial/starting state, q_0 is in Q
F	A set of final/accepting states, which is a subset of Q
δ	A transition function, which is a total function from Q x Σ to 2^Q

  $\delta: (Q \times \Sigma) \rightarrow \mathbf{2^Q}$: 2^Q is the power set of Q, the set of *all subsets* of Q

  $\delta(q,s)$: The set of all states p such that there is a transition labeled s from q to p

  $\delta(q,s)$ is a function from Q x S to 2^Q (but not only to Q)

Properties of NFA

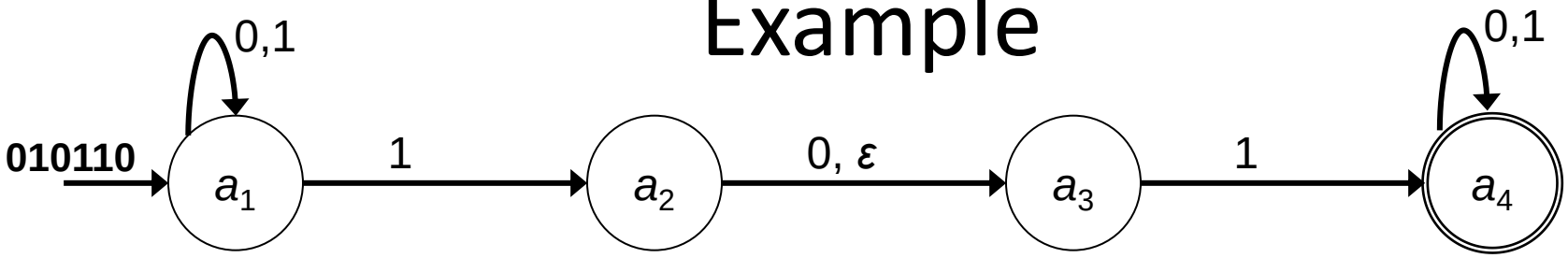


- We already know DFA, so it would be sufficient to look into the differences of properties between the two.
- In NFA a state may have –
  - Zero or more exiting arrows for each alphabet symbol.
  - Zero or more exiting arrows with the label ε.
- So we can see that, not all steps of a computation follows in a unique way from the preceding step. There can be multiple choices to move from one state to another with a symbol. That's the reason it's computation is called nondeterministic.

Running an NFA

- If we encounter a state with multiple ways to proceed –
  - The machine splits into multiple copies of itself and follows all the possibilities in parallel.
  - Each copy of the machine takes one of the possible ways to proceed and continues as before.
  - If there are subsequent choices, the machine splits again.
- If a state with an ε symbol on an exiting arrow is encountered without reading any input, the machine splits into multiple copies,
  - one following each of the exiting ε-labeled arrows and
  - one staying in the current state.
- If the next input symbol doesn't appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with it.
- If any one of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input strings.
- So, nondeterminism may be viewed as a kind of parallel computation wherein several processes can be running concurrently.
- If at least one of these processes accepts then the entire computation accepts.

Example



Symbol read

0 ----------

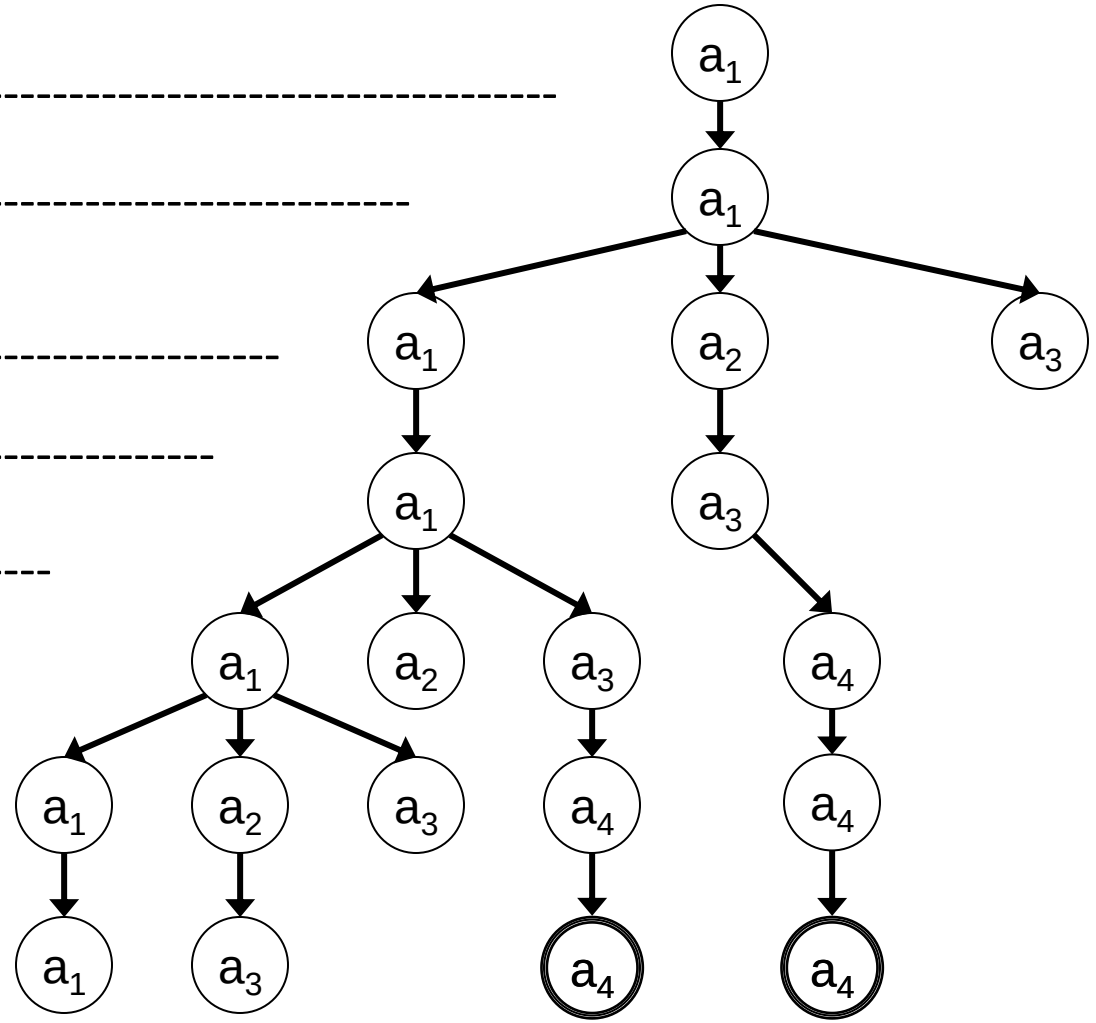
1 ----------

0 ----------

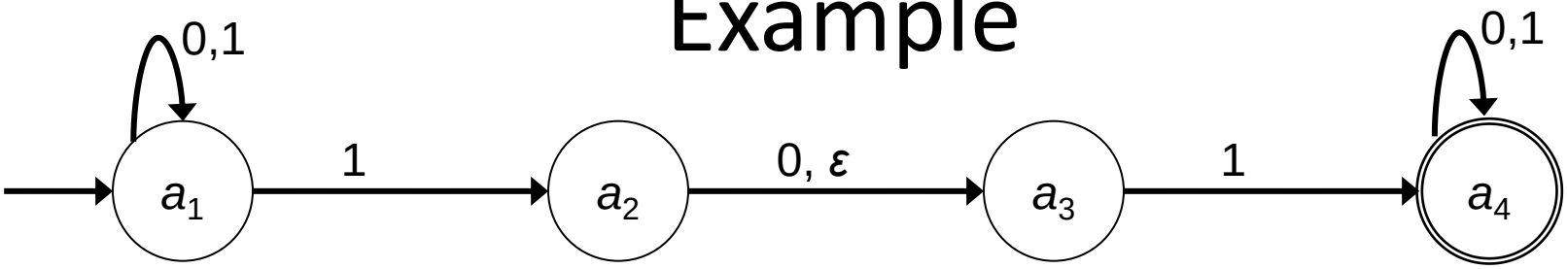
1 ----------

1 ----------

0 ----------



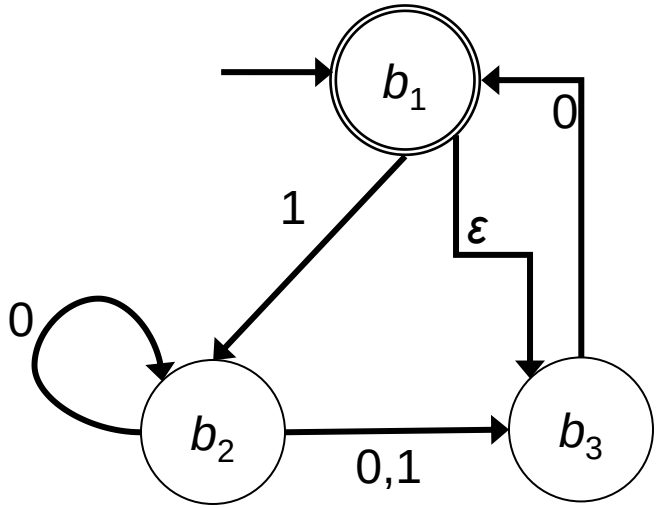
Example



- Let, the above NFA $N_1 = (Q_1, \Sigma, \delta_1, a_1, F_1)$.
  - $Q_1 = \{a_1, a_2, a_3, a_4\}$.
  - $\Sigma = \{0, 1\}$.
  - δ_1 is given as –

	0	1	ε
a_1	$\{a_1\}$	$\{a_1, a_2\}$	ϕ
a_2	$\{a_3\}$	ϕ	$\{a_3\}$
a_3	ϕ	$\{a_4\}$	ϕ
a_4	$\{a_4\}$	$\{a_4\}$	ϕ

  - a_1 is the start state.
  - $F_1 = \{a_4\}$.



Example

- Let, the above NFA $N_2 = (Q_2, \Sigma, \delta_2, b_1, F_2)$.
  - $Q_2 = \{b_1, b_2, b_3\}$.
  - $\Sigma = \{0, 1\}$.
  - δ_2 is given as –

	0	1	ε
b_1	ϕ	$\{b_2\}$	$\{b_3\}$
b_2	$\{b_2, b_3\}$	$\{b_3\}$	ϕ
b_3	$\{b_1\}$	ϕ	ϕ

  - b_1 is the start state.
  - $F_2 = \{b_1\}$.