LECTURE 01

Introduction to Theory of Computation

Course Code : CSE 3103
Course Title : Theory of Computation

Md. Sozib Hossain
Lecturer, CSE, RUET
sozib.hossain@cse.ruet.ac.bd

Md. Mazharul Islam
Lecturer, CSE, RUET
mazharul.islam@cse.ruet.ac.bd

INTRODUCTION

<u>THEORY OF COMPUTATION</u>

The theory of computation is the branch that deals with **what problems can be solved** on **a model of computation**, using an algorithm, **how efficiently they can be solved** or to what degree.

3 CENTRAL AREA.

1. AUTOMATA THEORY
2. COMPUTABILITY THEORY
3. COMPLEXITY THEORY

They are linked by the question:

- What are the fundamental **capabilities and limitations** of computers?

AUTOMATA THEORY

AUTOMATA THEORY

Dealing with developing formal **mathematical models of computation** that reflect real-world **computers**. Other words, representing a physical machine on paper.

COMPUTER/AUTOMATON

Computing Device or Machine. For example: Laptop, Desktop, Calculator etc.

MATHEMATICAL MODELS OF COMPUTATION

- Finite Automata
  - Deterministic Finite Automata (DFA)
  - Non-deterministic Finite Automata (NFA)
- Push Down Automata
- Context free grammar
- Turing Machine

COMPUTABILITY THEORY

COMPUTABILITY THEORY

The classification of problems is by those that are solvable and those that are not.

Problems

Computable by Designed Automata	Not Computable by Designed Automata

Certain basic problems cannot be solved by computers. One example of this phenomenon is the problem of determining **whether a mathematical statement is true or false**. No computer algorithm can perform this task.

COMPLEXITY THEORY

COMPLEXITY THEORY

The objective is to classify problems as easy ones and hard ones.

Computable Problems

Computationally Easy problem	Computationally Hard problem

Easy Problem: Sort a list of numbers in ascending order.

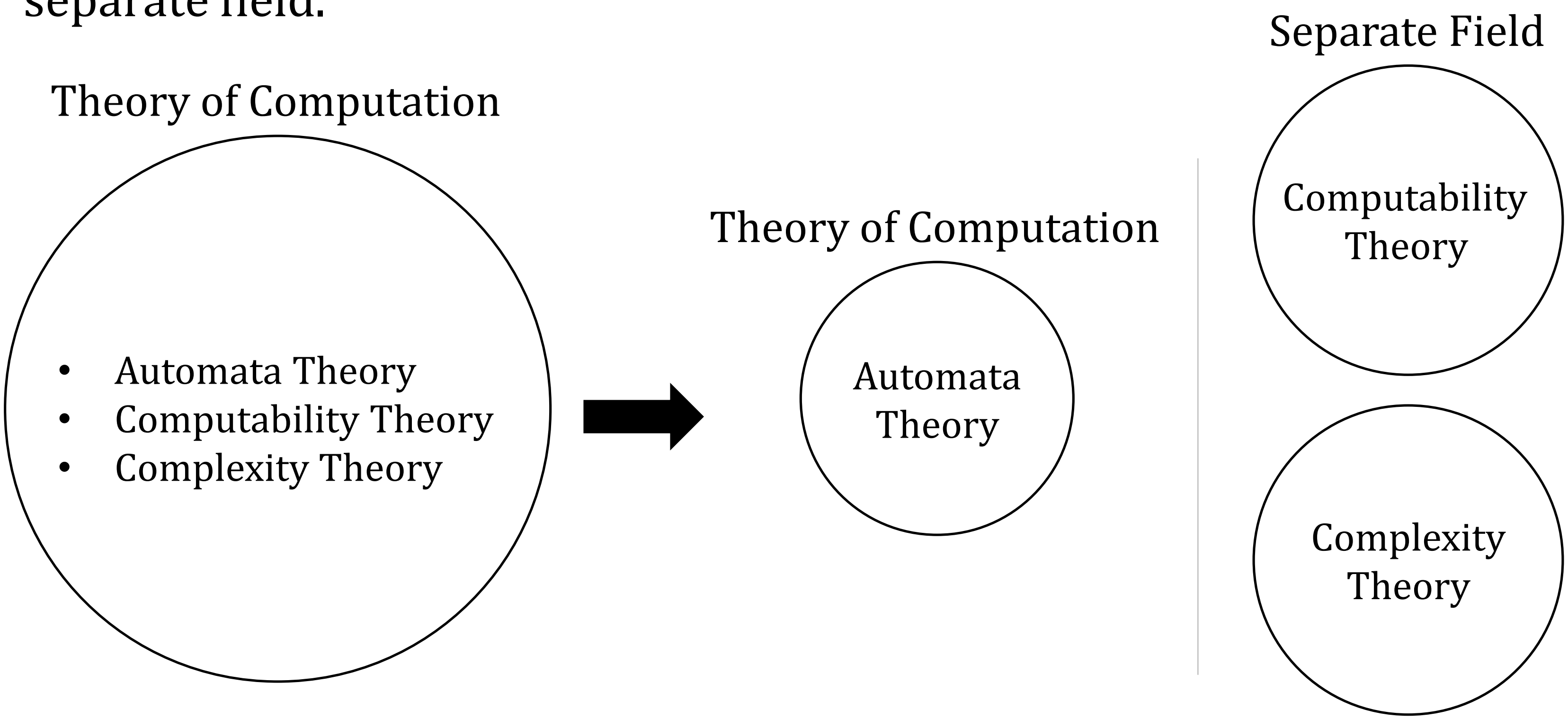
Hard Problem: Schedule of classes for the entire university to satisfy some reasonable constraints.

- **What makes some problems computationally hard and others easy?**

This is the central question of complexity theory.

THEORY OF COMPUTATION

In Theory of Computation, we will mostly discuss about Automata Theory. Computability and Complexity theory are discussed in the separate field.



Hence, Theory of computation is also called Automata Theory.

IMPORTANCE OF THIS COURSE

There are several reasons why the study **Theory of Computation**.

- Finite Automata are a useful model for many important kinds of **hardware and software**.
- Students are expected to learn about software for designing and checking the behavior of **digital circuits**.
- They can explain:
    - **lexical analyzer,**
    - **syntax analyzer and**
    - **semantic analyzer**

  of a typical compiler.

Modeling a Computer

Computer/Automaton

Computing Device or Machine. For example: Laptop, Desktop, Calculator etc.

Since, we can say typical calculator is a computer.

So,

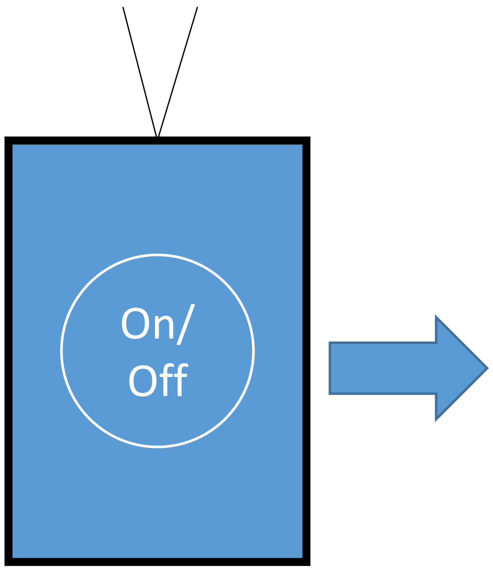
- **Is calculator that perform only addition a computer?**

  -Yes.

- **Is calculator that has only power button a computer?**

  -Yes.

MODELING A COMPUTER (CONT…)



States = {On, Off}

Input = {Press, Not Press}

Transition Function =

	Input	
States	Press	Not Press
On	Off	On
Off	On	Off

Initial State = {Off}

Fig 01: Simple computer with one button.

Physical Computer

Mathematical Model of Computer

BASIC TERMINOLOGY

SET

A set is a group of objects represented as a unit.

S = {5, 7, 8} Here, 5 7 and 8 are called **elements or members**.

- **A** is a **subset** of **B**, written $A \subseteq B$ and A is a **proper subset** of B, written $A \subset B$, if A is a subset of B and not equal to B.
- The **order** of a set doesn't matter, nor does **repetition** of its members. {1, 2, 3} & {2, 3, 1} - equivalent | {1, 2, 3} & {1, 2, 2, 3, 3} - equivalent.
- An infinite set use "**...**" notation to mean "continue the sequence forever".
- The set with zero members **-empty set**, written as ∅. A set with one member **-singleton set**, and a set with two members **-unordered pair**.
- Two sets **A** and **B**, the **union** written as $A \cup B$, the **intersection** written as $A \cap B$, the **complement** of A, written as $\bar{A}$.

BASIC TERMINOLOGY (CONT…)

SEQUENCES AND TUPLES

A sequence of objects is a list of these objects in some order.

$$(7, 21, 57)$$

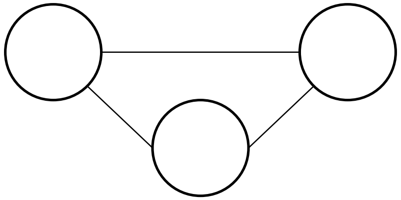
- **Order & Repetition** matters in a sequence.
- As with sets, sequences may be finite or infinite. Finite sequences often are called **tuples**. A 2-tuple is also called an **ordered pair**.
- The **power set** of A is the set of all subsets of A. If **A** is the set $\{0, 1\}$, the power set of A is the set $\{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.
- The **Cartesian product or cross product** of A and B, written $A \times B$, is the set of all ordered pairs wherein the first element is a member of A and the second element is a member of B. If $A = \{1, 2\}$ and $B = \{x, y, z\}$,

$$A \times B = \{ (1,x), (1,y), (1,z), (2,x), (2,y), (2,z) \}.$$

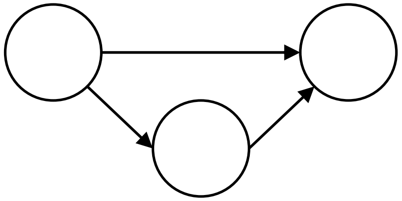
BASIC TERMINOLOGY (CONT…)

GRAPHS

An undirected graph, or simply a graph, is a set of points with lines connecting some of the points. The points are called **nodes or vertices**, and the lines are called **edges**.



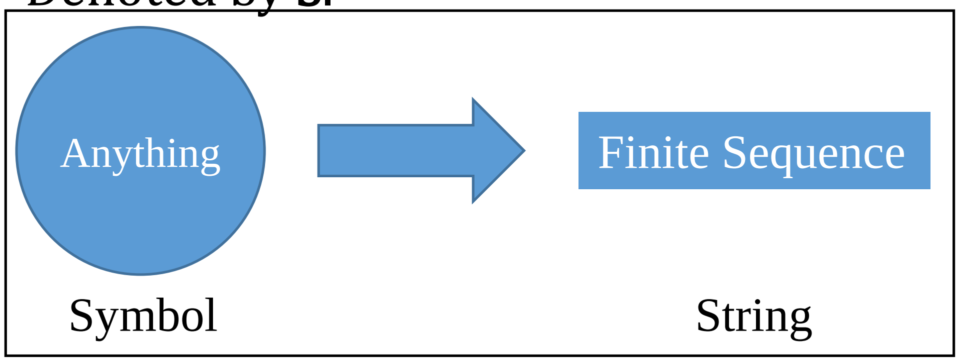
- We may allow an edge from a node to itself, called a **self-loop.**
- Label the nodes and/or edges of a graph, which then is called a **labeled graph.** Nodes are labeled by **state name.** Edges are labeled by input **symbol.**
- A **directed graph** has arrows instead of lines.



BASIC TERMINOLOGY (CONT…)

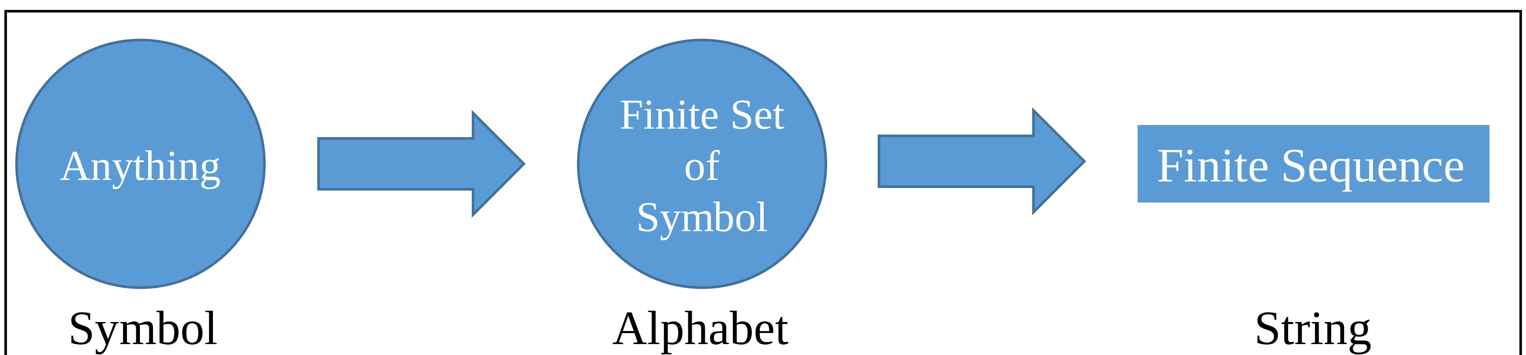
STRING

A string is a finite sequence of symbols selected from some alphabet.
Denoted by **S**.



In general

In TOC



BASIC TERMINOLOGY (CONT…)

SYMBOL & ALPHABET

Symbol is user defined entity or it is collection of letters, digits, pictures and special characters.

{a-z, A-Z, 0-9, *, #, @...}

An alphabet is a finite set of symbols denoted by Σ.

For Binary number

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

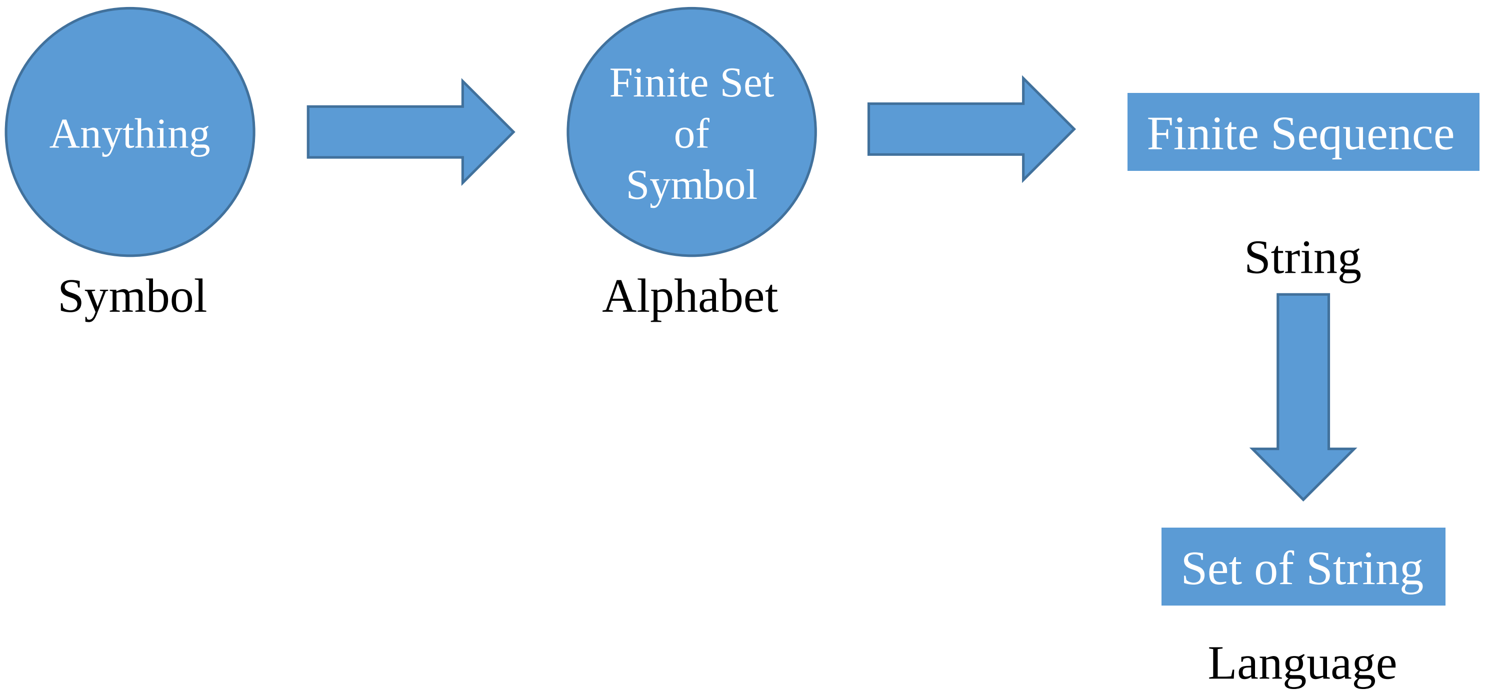
For Hex Number

$$\Sigma = \{0\text{-}9, A\text{-}F\}$$

BASIC TERMINOLOGY (CONT…)

LANGUAGE

A language is a set of string that satisfy specific criteria.



BASIC TERMINOLOGY (CONT…)

LANGUAGE

For Example, all three digits binary numbers.

Symbol: {a-z, A-Z, 0-9, *, #, @...}

Alphabets: {0, 1}

String: {0, 1, 00, 01, 10, 10, 11, 000, 001, 010, 011, . . .}

Language: {000, 001, 010, 011, 100, 101, 110, 111}

BASIC TERMINOLOGY (CONT…)

MORE LANGUAGE EXAMPLE

1. All binary even numbers.
2. All three digit binary even numbers.
3. Identifier in C programming Language.
4. All binary number starting with 101.
5. All binary number ending with 111.
6. All binary number starting with 111 and ending with 101.
7. All string that contains "aba" as sub string over the alphabet $\Sigma = \{a, b, c, d\}$.
8. All binary numbers that contains even number of 0.
9. All binary numbers that contains odd number of 1.

LENGTH OF A STRING

Length of a string represented by $|S|$.

$\Sigma = \{a,b\}$ and $|S| = 2$

$L = \{aa,ab,ba,bb\}$

$\Sigma = \{a,b\}$ and $|S| = 3$

$L = ?$

$\Sigma = \{a,b\}$ and $|S| = 0$

$L = ?$

POWER OF ALPHABET

It is expressed by Σ^k which represent the set of strings of length k each of those symbols are of Σ.

Example-

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

Then $\Sigma^0= \{\varepsilon\}$ which means a empty set.

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

$\Sigma^2 = \{00,01,10,11\}$

$\Sigma^3 = \{000,001,010,011,100,101,110,111\}$

$\Sigma^* = \Sigma+ + \varepsilon$

POWER OF ALPHABET

KLEEN CLOSURE

Strings of all possible lengths over ∑ including ε.

$$\Sigma^* = \Sigma^0 + \Sigma^1 + \Sigma^2 + \Sigma^3 + \ldots$$

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

Then $\Sigma^*= \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

POSITIVE CLOSURE

Strings of all possible lengths over ∑ excluding ε.

$\Sigma^+ = \Sigma^1 + \Sigma^2 + \Sigma^3 + \Sigma^4 + \ldots$

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

Then $\Sigma^+= \{0, 1, 00, 01, 10, 11, \ldots\}$