

# CSE-217: Theory of Computation

## Introduction

Lec Md Jakaria

Department of Computer Science and Engineering  
Military Institute of Science and Technology

October 25, 2019



# Overview



# Overview

Three traditionally central areas of the theory of computation.

- Automata
- Computability
- Complexity



# Overview

Three traditionally central areas of the theory of computation.

- Automata
- Computability
- Complexity

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



# Complexity Theory

Computer problems come in different varieties

- Easy
- Hard



# Complexity Theory

Computer problems come in different varieties

- Easy
- Hard

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



# Computability Theory

## Again Computer problems come in different varieties

- Solvable
- Unsolvable



# Computability Theory

## Again Computer problems come in different varieties

- Solvable
- Unsolvable

What makes some problems computationally solvable and others unsolvable?





## Complexity Theory vs Computability Theory

The theories of computability and complexity are closely related. In complexity theory, the objective is to classify problems as easy ones and hard ones, whereas in computability theory the classification of problems is by those that are solvable and those that are not. Computability theory introduces several of the concepts used in complexity theory.



# Automata Theory

## Automata Theory

**Automata theory deals with the definitions and properties of mathematical models of computation.**



# Automata Theory

## Automata Theory

**Automata theory deals with the definitions and properties of mathematical models of computation.**

## Example 1

**The Finite Automaton**  
used in text processing, compilers, and hardware design.



# Automata Theory

## Automata Theory

**Automata theory deals with the definitions and properties of mathematical models of computation.**

## Example 1

### **The Finite Automaton**

used in text processing, compilers, and hardware design.

## Example 2

### **The Context-Free Grammar**

used in programming languages and artificial intelligence.



# MATHEMATICAL NOTIONS AND TERMINOLOGY



## SETS

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

$$S = \{2, 13, 4, 256\}$$



## SETS

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

$$S = \{2, 13, 4, 256\}$$

- Elements or members
- Subset / Proper subset
- Multiset
- Finite / Infinite Set
- Empty/Singleton set
- Unordered Pair
- Union
- Intersection
- Complement
- Venn diagram



# SEQUENCES AND TUPLES

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

$$S = (2, 8, 512)$$





# SEQUENCES AND TUPLES

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

$$S = (2, 8, 512)$$

Finite sequences often are called  
**tuples.**

A sequence with **k** elements is a  
**k-tuple.**



# FUNCTIONS AND RELATIONS

A **function** is an object that sets up an input–output relationship.

$$f(a) = b$$

A function also is called a **mapping**.

$$f : A \rightarrow B$$



# GRAPS

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**



# STRINGS AND LANGUAGES

An **alphabet** to be any nonempty finite set. The members of the alphabet are the **symbols** of the alphabet.

$$\begin{aligned}\Sigma_1 &= \{0, 1\} \\ \Sigma_2 &= \{a, b, c, \dots, x, y, z\} \\ \Gamma &= \{0, 1, x, y, z\}\end{aligned}$$



# STRING AND LANGUAGES

A **string** over an **alphabet** is a finite sequence of symbols from that alphabet, usually written next to one another and not separated by commas.



# STRING AND LANGUAGES

A **string** over an **alphabet** is a finite sequence of symbols from that alphabet, usually written next to one another and not separated by commas.

If  $\Sigma_1 = \{0, 1\}$ , then **01001** is a string over  $\Sigma_1$ .



# STRING AND LANGUAGES

A **string** over an **alphabet** is a finite sequence of symbols from that alphabet, usually written next to one another and not separated by commas.

If  $\Sigma_1 = \{0, 1\}$ , then **01001** is a string over  $\Sigma_1$ .

A **language** is a set of strings.



# Thank You

