Introduction to the Theory of Computation

Foundations and Motivation

Sara Sorahi

Introduction to the Theory of Computation

Course Note

These slides introduce the central ideas of the course and serve as the conceptual foundation for everything that follows. Students are encouraged to return to these ideas throughout the semester.

What Do We Mean by Computation?

In everyday language, *computation* often means numerical calculation.

In theoretical computer science, computation means:

- a precise, step-by-step procedure,
- operating mechanically on input,
- producing output according to fixed rules.

[1]

[1] Formally, an algorithm is a finite description of a mechanical procedure.

Example: A Simple Computable Task

Problem: Given a number n, determine whether n is even.
Why this problem is computable: - The procedure is clearly defined. - It always terminates. - It produces the correct answer for all inputs.

Why We Use Abstract Models

Real computers are:

- hardware-dependent,
- complex,
- constantly evolving.

Key idea: We study idealized computational models that capture power without implementation details.

Central question:

What is the simplest machine that can solve a given problem?

Languages as Objects of Study

Instead of informal problems, we study **languages**.

- An **alphabet** Σ is a finite set of symbols.
- A **string** is a finite sequence of symbols from Σ.
- A **language** is a set of strings over Σ.

Example: A Formal Language

Alphabet: Sigma $=$ a, b
Language: L $=$ w $|$ w contains an even number of a's
Examples: - In L: bb, abba, aa - Not in L: a, bab

Machines as Language Recognizers

A computational model is viewed as a **language recognizer**.

Given an input string, a machine:

- accepts if the string is in the language,
- rejects otherwise.

Example: Language Recognition

Language: L = w in 0,1* | w ends with 01

To recognize L, a machine must: - read the entire input, - remember the last two symbols, - accept only if they are 01.

Grammars as Language Generators

Languages can also be described using **grammars**.

Grammar rules: $S \to aS \mid b$
Generated language: $a^n b | n 0$

Grammars emphasize structure; machines emphasize process.

Computational Models and Memory

Different models differ mainly in their **memory capacity**.
2tablegraywhite

Model	Memory	Capability
Finite Automaton	Finite control	Pattern recognition
Pushdown Automaton	One stack	Nested structure
Turing Machine	Unbounded tape	General computation

Limits of Computation

The Halting Problem (informal):
Given a program and an input, determine whether the program eventually stops.
Fact: There is no algorithm that can solve this problem for all inputs.

This limitation is mathematical, not technological.

Why This Course Matters

The theory of computation:

- forms the foundation of computer science,
- explains the limits of automation,
- informs programming language design,
- connects computation with logic and linguistics.

Takeaway: This course teaches you to reason precisely about what computation can and cannot do.