CSCI3130 Formal Languages and Automata Theory

Ryan Chan

December 2, 2025

Abstract

This is a note for **CSCI3130 Formal Languages and Automata Theory**.
Contents are adapted from the lecture notes of CSCI3130, prepared by Tsung-Yi Ho, as well as some online resources.
This note is intended solely as a study aid. While I have done my best to ensure the accuracy of the content, I do not take responsibility for any errors or inaccuracies that may be present. Please use the material thoughtfully and at your own discretion.
If you believe any part of this content infringes on copyright, feel free to contact me, and I will address it promptly.
Mistakes might be found. So please feel free to point out any mistakes.

Contents

Chapter 1

Introduction

1.1 Theory of Computation

In the theory of computation, we study the following:

1. **Formal Languages**

This is the abstraction of the general characteristics of programming languages. It consists of a set of symbols and some rules, i.e. strings and grammar of formation, and these symbols are then combined into sentences.

2. **Automata Theory**

This is the study of the dynamic behaviours of "discrete-parameter information systems" in the form of "abstract computing devices."

Types of automata are distinguished by their temporary memory.

For finite automata, there is no temporary memory. Pushdown automata use a stack, and Turing machines use random-access memory. The computational power of finite automata is the smallest, while Turing machines have the highest, with pushdown automata in between.

There are three major models of automata:

- generator: with output only

- acceptor: with input only

- transducer: with both input and output

3. **Computability**

This is the study of the problem-solving capabilities of computational models.

We can classify problems based on resources:

- Impossible problems

- Possible with unlimited resources but impossible with limited resources

- Possible-with-limited-resources problems

Or by time:

- Undecidable problems

- Intractable problems

- Tractable problems

4. **Computational Complexity**

This is the study of the efficiency of problem-solving. To unify comparison, we use an abstract model for problem execution. A Turing machine is usually used, since although simple, it has been proved to be able to simulate any problem-solving steps designed by human beings.

Problems can be classified into:

- P: Polynomial-time problems. These are problems that can be solved quickly (in polynomial time) by a normal computer.

- NP: Non-deterministic Polynomial time. These are problems where we do not know how to solve them quickly, but if someone gives us a solution, we can verify it quickly (in polynomial time).

- NP-hard: at least as hard as every NP problem.

- NP-complete: both NP and NP-hard.

1.2 Mathematical Preliminaries and Notation

> **Remark.** Here only contain partial contents adopted from lecture note.

1.3 Three Basic Concepts

Chapter 2

Finite Accepter

Chapter 3

Regular Language

Chapter 4

Context-Free Language

Chapter 5

Pushdown Automata

Chapter 6

Turing Machine