Computation and Logic

Generated by LectureMate

February 22, 2025

1 Introduction

This lecture marks the beginning of the Computation and Logic section of the course. After eight weeks of focusing on logic, we will now delve into computation, specifically the simplest model of computation that can still perform useful tasks.

2 Key Figure

Today's notable figure is Stephen Carlini, who, while not a computer scientist, has made significant contributions to early computer science results and is well-known for his work in set theory.

3 Understanding Computation

3.1 What is Computation?

Computation can be understood as a process that involves moving between different states. We will assume a finite number of states for simplicity. The machine we will discuss reads input, which can be any symbol from a defined alphabet, and processes this input to determine its output.

3.2 Types of Machines

There are two fundamental types of machines:

- Machines that classify inputs as good or bad (e.g., recognizing valid sentences).
- Machines that produce outputs based on inputs (e.g., translation machines).

For this section, we will focus on machines that provide a yes or no answer regarding the validity of the input.

4 Principles of Simple Computation

The principles of simple computation include:

- Moving between states based on input.

- Recognizing good and bad states.
- The ability to reboot the machine.

5 Finite Automata

5.1 Definition

A finite automaton (or finite state machine) is a model that consists of:

- A finite set of states.
- A transition function that determines state changes based on input.
- An initial state where computation begins.
- Accepting states that signify good inputs.

5.2 Applications

Finite automata have numerous applications in modern technology, including:

- Washing machines that adjust cycles based on load.
- Central heating systems that respond to temperature inputs.
- Traffic light systems that adapt to traffic conditions.
- Programming language implementations and CPU controllers.

6 Graphical Representation of Automata

6.1 State Diagrams

Automata can be represented graphically using state diagrams, where:

- Circles represent states.
- Arrows indicate transitions based on input.
- Double circles denote accepting states.

6.2 Example

Consider a simple automaton that reads inputs A and B:

- Starting in state 0, reading A transitions to state 1.
- Reading B keeps the machine in state 0.

This automaton accepts strings based on its final state after processing the input.

7 Transition Functions

The transition function, denoted as δ, maps the current state and input symbol to the next state. For example, if the machine is in state q_1 and reads input a, it transitions to state q_2.

8 Deterministic Finite Automata (DFA)

A DFA is characterized by:

- A single start state.
- Exactly one transition for each input symbol from every state.

This determinism ensures that the next state is uniquely determined by the current state and input.

9 Notation and Definitions

9.1 Formal Definition of DFA

A deterministic finite automaton is formally defined as a 5-tuple:

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

where:

- Q is a finite set of states.
- Σ is the input alphabet.
- $\delta : Q \times \Sigma \to Q$ is the transition function.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting states.

10 Building Automata

To construct a DFA that accepts strings with an even number of zeros and an odd number of ones, we can define states based on the parity of the counts of zeros and ones. This leads to a structured approach to automata design.

11 Conclusion

In this lecture, we introduced the concept of finite automata, their applications, and how to represent them graphically. We also discussed the formal definitions and properties of deterministic finite automata. Future lectures will build on these concepts to explore more complex automata and their applications.