Parul® University

NAAC A++

Church Turing thesis
Chapter 5: Undecidability

Prof. Riddhi Atulkumar Mehta
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

INDEX

Learning Goals

- Understand what the Church-Turing Thesis asserts
- Explore multiple models of computation
- Learn historical developments that led to the thesis
- Appreciate the philosophical and theoretical implications
- Recognize its limits and open questions

Introduction

- What is a computation? What does it mean to “compute” a function?
- Informally, computation involves step-by-step execution of a finite procedure (algorithm) to solve a problem.
- Church-Turing Thesis provides a formal answer to what is computable.

Historical Background

- **Alonzo Church** (1936): Introduced **λ-calculus**, a formal system for expressing computation via function abstraction and application.
- **Alan Turing** (1936): Independently developed the **Turing Machine**, a theoretical machine to model algorithmic processes.
- Both proved the same class of computable functions—leading to the **Church-Turing Thesis**.

The Church-Turing Thesis

- “A function is effectively computable if and only if it is computable by a Turing machine.”
- “Effectively computable” = can be computed by a human or machine using an algorithm, without intuition or guesswork.
- The thesis is not a formal theorem, but a philosophical hypothesis supported by overwhelming evidence.

Formal Models of Computation

Model	Inventor	Year	Description
Turing Machine	Alan Turing	1936	Machine with infinite tape and head for reading/writing symbols
λ-Calculus	Alonzo Church	1936	Formal system based on variable binding and substitution
Recursive Functions	Gödel/Kleene	1930s	Functions built using basic operations and recursion
Post Systems	Emil Post	1943	Production rules on strings (rewriting systems)

Turing Machine

- A Turing Machine is a **mathematical model** of computation that operates on an infinite tape with a finite set of rules.
- Capable of simulating any algorithm.
- Forms the basis of modern computing models.

λ-Calculus – A Functional Perspective

- Uses variable binding and substitution.
- Core operations: abstraction (functions), application (function calls).
- Foundation for many functional programming languages (e.g., Lisp, Haskell).

What Does "Computable" Mean?

- A function f: N→N is **computable** if there exists an algorithm (or TM) that produces f(n) for every input n.
- **Example**:
  - Computable: Addition, multiplication, sorting
  - Non-computable: Halting problem, truth of arbitrary mathematical statements

Implications of the Thesis

- **Defines the boundary** of what can be computed using any physical device or algorithm.
- Provides a **unified model** of computation used in:
- Programming language theory
- Complexity theory
- Logic and formal verification

Strong Church-Turing Thesis

“Any reasonable model of computation can be efficiently simulated by a Turing machine.”

- Focuses not only on computability but also **efficiency (time/space)**.
- Forms the basis of complexity theory (P, NP, etc.)

Quantum Church-Turing Thesis

- Proposes that any physically realizable computational process can be efficiently simulated by a **quantum computer**.
- Still under investigation.
- Shor’s algorithm (factoring) suggests quantum computers can outperform classical TMs in some tasks.

Parul® University

NAAC GRADE A++

https://paruluniversity.ac.in/