Church's Thesis

Ayush Tandon (2K22/CO/133)

October 29, 2024

1 Introduction

The **Church-Turing Thesis** is a fundamental concept in the field of computer science and mathematical logic. Proposed independently by Alonzo Church and Alan Turing in the 1930s, it posits that any function which can be computed algorithmically can also be computed by a *Turing machine*. This thesis has profound implications for the theory of computation and defines the limits of what can be computed in the physical universe.

2 Definition of the Church-Turing Thesis

The Church-Turing Thesis, also referred to as **Church's Thesis** or **Turing's Thesis**, suggests that:

> "**Every computation that can be carried out in the real world can be effectively performed by a Turing Machine.**"

This implies that any problem which can be solved by a set of well-defined instructions or algorithms can be computed by a Turing machine.

3 Formalization of Computability

Before the Church-Turing Thesis, there was no formal definition of computability. Church introduced the concept through *lambda calculus*, while Turing introduced the *Turing machine*. Both formalizations were later proven to be equivalent in their computational power.

3.1 Turing Machines

A *Turing machine* is an abstract computational model defined by Alan Turing. It consists of:

- An infinite tape, divided into cells that can hold symbols.
- A head that can read and write symbols on the tape and move left or right.
- A finite set of states, with a set of rules for transitioning between them.

The Turing machine's ability to simulate any algorithm forms the basis for the thesis.

3.2 Lambda Calculus

Lambda calculus, introduced by Alonzo Church, is a formal system that uses function abstraction and application to perform computation. In lambda calculus, every function is represented as an expression composed of variables, and computation proceeds by substituting values into these expressions.

Turing Machine and Lambda Calculus are equivalent in power. The lambda calculus can be called the **smallest** universal programming language of the world.

4 Examples of Computable and Non-Computable Problems

The Church-Turing Thesis helps distinguish between **decidable** and **undecidable** problems.

4.1 Example: Computable Problem

An example of a computable problem is integer addition. We can define an algorithm to compute the sum of two integers, which a Turing machine can execute.

4.2 Example: Non-Computable Problem

A classic example of a non-computable problem is the **Halting Problem**. Given a description of a Turing machine and an input, there is no algorithm that can determine whether the machine will halt or continue running indefinitely.

5 Implications of the Church-Turing Thesis

The thesis has several important implications:

- **Computational Universality**: Any function that can be computed by any mechanical process can be computed by a Turing machine.
- **Limits of Computation**: Certain problems, such as the Halting Problem, cannot be solved by any algorithm and are therefore undecidable.
- **Equivalence of Models**: Models like Turing machines, lambda calculus, and recursive functions are all equivalent in terms of computational power.

6 Conclusion

- Anything that can be done on any existing digital computer can also be done by a Turing Machine.
- No one has yet been able to suggest a problem that is intuitively considered solvable by an algorithm for which a Turing Machine program cannot be written.
- Alternative models have been proposed for mechanical computation, but none of them is more powerful than the Turing Machine model.
- Despite its lack of formal proof, the thesis has been widely accepted due to the lack of counterexamples and has shaped the field of computer science.