Lecture 5: Diagonalization and the Incompleteness Theorem

Anup Rao

January 17, 2023

Last time, we discussed the fact that there are functions that require large circuits. Today, we use a similar strategy to argue that there are functions that cannot be computed by Turing machines. A very similar idea gives an impossibility result for Turing Machines.

Theorem 1. *There is a function that is not computed by any Turing Machine.*

Before we see the the simple proof, let us point out that this is philosophically a very powerful fact. A consequence of it is that assuming the Church-Turing Thesis is true, there are some ways to manipulate information that can never occur in the universe. It seems hard to imagine a physical process that violates the Church-Turing thesis, and it also seems hard to stomach the fact that the universe cannot manipulate information in a particular way, yet one of those two (admittedly wishy washy) strange things must happen.

We shall need some notation before discussing the proof. Given a string α, we write M_α to denote the Turing Machine whose code is α.

Proof Consider the function $f : \{0,1\}^* \to \{0,1\}$ defined as follows:

$$f(\alpha) = \begin{cases} 1 & \text{if } M_\alpha(\alpha) = 0 \\ 0 & \text{else.} \end{cases}$$

No Turing Machine can compute this function, for if there was some machine that could, then let γ denote the binary encoding of its code. Then we have that $M_\gamma(\gamma) = f(\gamma)$, but this contradicts the definition of f, since if $f(\gamma) = 0$, then $M_\gamma(\gamma)$ cannot be 0, and if $f(\gamma) = 1$, $M_\gamma(\gamma)$ cannot be 1. ■

You may object that the uncomputable f that we found above is very unnatural, but actually it is not hard to come up with natural examples that are also impossible to compute using Turing Machines.

For example, we can define the function $\mathsf{HALT} : \{0,1\}^* \to \{0,1\}$ that takes as input two strings α, x, and then decides whether $M_\alpha(x)$ halts or runs forever. This seems like a very useful function to compute, but it is also uncomputable.

Theorem 2. HALT *is not computable by a Turing Machine.*

Proof Suppose it was. Then consider the machine M that on input α first simulates $\mathsf{HALT}(\alpha, \alpha)$. If the answer is that $M_\alpha(\alpha)$ halts, then M simulates $M_\alpha(\alpha)$ and outputs the opposite of its output. If $M_\alpha(\alpha)$ does not halt, then M outputs 0. Then M computes the uncomputable function f above. ■

Gödel's Incompleteness Theorem

Diagonalization was also used to prove Gödel's famous incompleteness theorem. The theorem is a statement about proof systems. We sketch a simple proof using Turing machines here.

A proof system is given by a collection of axioms. For example, here are two axioms about the integers:

1. For any integers a, b, c, $a > b$ and $b > c$ implies that $a > c$.

2. For any integer a, $a + 1 > a$.

Given a list of such axioms, a proof is a sequence of statements that uses the axioms to prove that a statement is true. For example, to prove that $a > b$ implies that $a + 1 > b$, we can combine the assumption $a > b$ with the axiom $a + 1 > a$ and the first axiom, to prove $a + 1 > b$.

Prior to Gödel's work, mathematicians were trying to axiomatize all of mathematics. They were looking for a set of finite axioms that could be combined to prove any proof statement. Godel proved that this a doomed project.

A set of axioms is *consistent* if the axioms don't contradict each other. The set of axioms is complete if every true statement can be derived from the set of axioms. Godel proved:

Theorem 3. *Every consistent finite set of axioms is incomplete.*

We give an alternate proof due to Chaitin. Given $x \in \{0,1\}^*$, its Kolmogorov complexity $K(x)$ is the length of the shortest program α such that $M_\alpha(.) = x$. Namely it is the length of the shortest program that outputs x. For each $x \in \{0,1\}^*, N \in \mathbb{N}$, let $S_{x,N}$ be the statement

$$K(x) > N.$$

Fact 4. *For every N, there is an x for which $S_{x,N}$ is true.*

Proof There are only a finite number of programs of length N, so for each N, there are only a finite number of x's such that $K(x) \leq N$. This means that almost all statements $S_{x,N}$ are true. ■

To prove Godel's theorem, suppose there is some finite set of axioms A. Consider the following program M_N:

- Enumerate over all pairs (x, α), where $x \in \{0,1\}^*$, $\alpha \in \{0,1\}^*$. If α describes a proof of $S_{x,N}$ using the axioms A, output x.

If the finite set of axioms were complete, M_N would always halt, since it would find some string x and a proof α proving $S_{x,N}$. But the program M_N can be described using just $O(\log N)$ bits, and it outputs a string x for which $K(x) > N$. For N large enough, this is a contradiction, and so A must be incomplete.