Church-Turing thesis, the Halting problem & Turing reductions

Arno Pauly

March 19, 2021

Some terminology

- A function $f : \Sigma^* \to \Sigma^*$ is called *computable*, if there is a TM that will halt for any input $w \in \Sigma^*$, and when halting, $f(w)$ is written on the tape.
- A function $F : \mathbb{N} \to \mathbb{N}$ is called *computable*, if the function $F_2 : \{0,1\}^* \to \{0,1\}^*$ mapping the binary representation of n to the binary representation of $F(n)$ is computable.
- A formal language $A \subseteq \Sigma^*$ is called *decidable* or *computable*, if there is a TM that accepts every $w \in A$ and rejects every $w \in \Sigma^* \setminus A$ (the complement of A).
- A formal language $A \subseteq \Sigma^*$ is called *computable enumerable* if there is a TM that accepts exactly the words $w \in A$ (and may not answer for words outside of A).

The Church-Turing thesis

Church-Turing thesis

The computable functions are exactly those functions that we would intuitively deem to be "computable", i.e. those that have some algorithmic procedure to obtain their outputs.

Church-Turing thesis (Physical)

The computable functions are exactly those functions for which there exists an abstract physical process to transform their input into an output.

Neither of these are mathematical theorems – we cannot prove them. But there is overwhelming evidence.

The existence of undecidability

We already know that there are undecidable languages – for there are only countably many Turing machines, but uncountably many formal languages.

The Halting problem

Theorem

Let & be an otherwise unused separation character. The language

$$\{\langle M \rangle \& w \mid \text{the TM } M \text{ accepts } w\}$$

is undecidable. In other words, there is no TM that reads a description of a TM M and a word w and eventually halts and answers "yes" if M halts on input w, and answers "no" otherwise.

Proof strategy

1. We will assume that there were a TM H that did decide the Halting problem.
2. From this, we will construct another TM G.
3. We will show that the existence of G is absurd.
4. So our initial assumption was wrong, and H cannot exist.

How the construction works

- G starts by writing an & at the end of the input, and then copying its input after the &.
- Then G does whatever H would do (we can just copy the instructions in).
- Except for Halting states: If H would halt and output “no”, G will halt and output “yes”.
- And if H would halt and output “yes’, G will go to a new states and loop there for ever.

Question

What happens if we give $\langle G \rangle$ as input to G?

Transferring results

- I had promised more – that it is undecidable whether the input TM M halts on empty input.
- Assume we could do the latter.
- Given $\langle M \rangle$ and w, we can built a TM M_w that starts by writing w on the tape (hardcoded) and then acts as M does.
- Then M_w halts on empty input if and only if M halts on w. QED.

Turing reductions

Definition (Intuitive idea)

A language A is Turing reducible to a language B (written $A \leq_{\mathrm{T}} B$, if there exists an algorithm to decide membership in A using a hypothetical library function (called "oracle" in the computability theory parlance) for deciding membership in B.

- If $A \leq_{\mathrm{T}} B$, then B is "at least as non-computable" as A is. If A is undecidable, then so is B.
- "Natural problems" tend to be either computable, or we can reduce the Halting problem to them.
- Example: "Given a multivariate integer polynomial, does it have an integral root?" is undecidable (stuff like $2x^y + y^2 - xyz$).