Logic, Computability and Incompleteness

Turing Computability

Wolfgang Schwarz

31 October 2025

Turing Machines

Turing Machines

We'd like a precise, formal account of **algorithms**, so that we can study whether there is an algorithm for a given task.

Informally, an algorithm is a mechanical, step-by-step procedure for transforming inputs into outputs.

Turing's idea: We can define a formal model of **following an algorithm**.

Following an algorithm involves reading and writing symbols etc. All this is condensed into the **Turing machine** model.

Turing Machines

A Turing machine consists of

- an unbounded tape divided into cells;
- a head that, at each step:
  1. reads the current cell,
  2. overwrites it with a stroke or a blank,
  3. moves one cell left or right;
- a finite set of internal states.

A **program** for a Turing machine specifies for each state/scanned symbol pair, what to write, which way to move, and which state to enter next.

Church-Turing Thesis

Any algorithm can be implemented by a Turing machine.

Turing Machines

https://turingmachine.io/

Universal Turing machines

Universal Turing machines

A **universal Turing machine** U simulates any other machine M on any input I.

Input to U: a **code** for M plus the data I.

Output: whatever M would output on I.

By the Church-Turing Thesis, U must exist.

We can also construct it explicitly.

The Halting Problem

The Halting Problem

The Halting Problem:

Decide, given a machine M and input I, whether M halts on I.

No Turing machine solves the Halting Problem.

Proof by diagonalization:

- Suppose $H(M, I)$ solves the Halting Problem.
- Build $D(M)$:
  - on input <code of M>,
  - ask H if M halts on input <code of M>;
  - if yes, loop forever; if no, halt.
- Run D on its own code.
- D halts iff it doesn't halt.

The undecidability of first-order logic

The undecidability of first-order logic

The Entscheidungsproblem: Is there an algorithm to decide whether any first-order sentence is valid?

Turing's strategy: reduce the Halting Problem to the Entscheidungsproblem.

- For any machine M and input I, build two first-order sentences:
  - $S_{M,I}$ describes the behaviour of M on I,
  - $H_{M,I}$ says "M halts on I".
- M halts on I iff $S_{M,I} \models H_{M,I}$.
- If we could decide validity, we could decide Halting.