CMSC 28100

Introduction to Complexity Theory

Spring 2024
Instructor: William Hoza

Which problems can be solved through computation?

Which languages are decidable?

Is every language decidable?

The liar paradox

Are you selecting option B as your answer to this question?

A: Yes

B: No

C: Yes

D: Yes

Respond at PollEv.com/whoza or text “whoza” to 22333

Self-rejecting Turing machines

- Let M be a TM (with a large enough input alphabet)
- A strange-but-legal thing we can do: Run M on $\langle M \rangle$
- Three possibilities:
  - M accepts $\langle M \rangle$
  - M rejects $\langle M \rangle$
  - M loops on $\langle M \rangle$
- **Definition:** We say that a Turing machine M is self-rejecting if M rejects $\langle M \rangle$

Self-rejecting Turing machines

- Let $\text{SELF-REJECTORS} = \{\langle M \rangle : M \text{ is a self-rejecting Turing machine}\}$

Lemma: SELF-REJECTORS is undecidable

- **Proof:** Let M be any TM. We'll show that M does not decide SELF-REJECTORS
  - Case 1: M is self-rejecting: $\langle M \rangle \in \text{SELF-REJECTORS}$, but M rejects $\langle M \rangle$ ❌
  - Case 2: M isn't self-rejecting: $\langle M \rangle \notin \text{SELF-REJECTORS}$, but M doesn't reject $\langle M \rangle$ ❌
  - Either way, M fails!

Contrived vs. natural

- Admittedly, SELF-REJECTORS is a contrived language, cooked up purely for the sake of proving an undecidability result
- Are there undecidable languages that are natural/well-motivated/interesting?
- Yes! Key example: The halting problem

The halting problem

Here is an attempt at designing an algorithm that solves the halting problem:

Given M and w:
1. Simulate M on w.
2. If it halts, accept; if it loops, reject.

- **Problem:** Given a Turing machine M and an input w, determine whether M halts on w.
- Roughly speaking, this is the problem of identifying bugs in someone else's code! 🐞

Does the proposed algorithm work?

A: No. Step 1 isn't legal, so the algorithm isn't well-defined

B: No. Step 2 isn't legal, so the algorithm isn't well-defined

C: Yes

D: No. The algorithm behaves incorrectly in some cases

Respond at PollEv.com/whoza or text "whoza" to 22333

The halting problem is undecidable

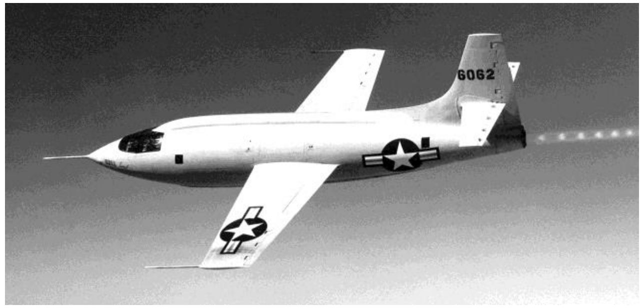
- Let $\mathrm{HALT} = \{\langle M, w\rangle : M \text{ is a Turing machine that halts on input } w\}$

Theorem: HALT is undecidable.

- **Proof by contradiction:** Assume that H decides HALT
- Let's design an algorithm that decides SELF-REJECTORS. Given $\langle M\rangle$:

1. Construct $\langle M'\rangle$, where M' is a modified version of M in which the accept state has been replaced with a looping state
2. Simulate H on $\langle M', \langle M\rangle\rangle$. If it accepts, accept; if it rejects, reject.

The Church-Turing thesis, revisited



- Let L be a language

Church-Turing Thesis:

There exists an “algorithm” / “procedure” for figuring out whether a given string is in L if and only if there exists a Turing machine that decides L.

- Computation is an intuitive notion rooted in everyday human experience
- Could it be possible to solve the halting problem using science and technology?

Hypercomputers

- A hypercomputer is a hypothetical device that can solve some computational problem that cannot be solved by Turing machines, such as the halting problem
- Could it be possible that there are hypercomputers at the centers of stars? Inside black holes?
- Could it be possible to build a hypercomputer?

The Physical Church-Turing Thesis

- Let L be a language

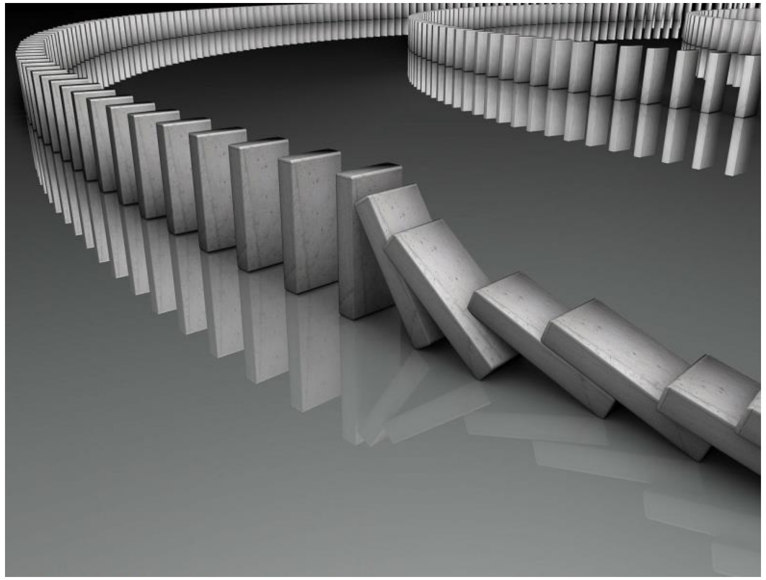
Physical Church-Turing Thesis:

It is physically possible to build a device that decides L if and only if there exists a Turing machine that decides L.

The Physical Church-Turing Thesis

- The standard Church-Turing thesis is a philosophical statement
- The Physical Church-Turing thesis is a scientific law
- Conceivably, it could be disproven by future discoveries… but that would be very surprising
- Analogy: Second Law of Thermodynamics
- Analogy: Cannot travel faster than the speed of light

Undecidability



- First, we proved that SELF-REJECTORS is undecidable
- Then, we used the fact that SELF-REJECTORS is undecidable to prove that HALT is undecidable
- Next, let's use the fact that HALT is undecidable to prove that other interesting languages are undecidable

Complement of the halting problem

- Let $\overline{\mathrm{HALT}} = \{\langle M, w\rangle : M \text{ does not halt on } w\}$
- **Claim:** $\overline{\mathrm{HALT}}$ is undecidable
- **Proof by contradiction:** Assume that H decides $\overline{\mathrm{HALT}}$
- Let's design an algorithm that decides HALT. Given $\langle M, w\rangle$:

1. Run H on $\langle M, w\rangle$
2. If it accepts, reject; if it rejects, accept.

Technicality: The first step of our algorithm should be to confirm that the input is "valid," i.e., confirm that the input has the form $\langle M, w\rangle$ for some Turing machine M and some input w to

M

The "acceptance problem"

- Let $\mathrm{A_{TM}} = \{\langle M, w \rangle : M \text{ is a Turing machine that accepts input } w\}$
- **Claim:** $\mathrm{A_{TM}}$ is undecidable
- **Proof by contradiction:** Assume that A decides $\mathrm{A_{TM}}$
- Let's design an algorithm that decides HALT. Given $\langle M, w \rangle$:

1. Construct $\langle M' \rangle$, where M' is a modified version of M in which all rejecting transitions have been changed into accepting transitions
2. Simulate A on $\langle M', w \rangle$. If it accepts, accept; if it rejects, reject.