CS1200: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Lecture 23: Unsolvability of the Halting Problem (and More)

Harvard SEAS - Fall 2025 *2025-11-18*

1 Announcements

- Salil's next OH Thu 1pm-1:45pm SEC 3.327, Mon 5:30pm-6:15pm Zoom.
- No staff office hours or sections Wed 11/26-Sun 11/30.
- Photos will be taken of SRE today. Do not sit in center section of the room if you do not want to be photographed.
- Three practice finals posted. Solutions will be released 12/1 (2022 final), 12/4 (2023 final), 12/7 (2024 final). Stay tuned for more review materials and review session dates.
- PS8 feedback
  - Time spent median/75th percentile: College 8hrs/9.75hrs, DCE 12hrs/15hrs.
  - "The pace at the start was just right. It seems to have ramped up quite significantly from I'd say towards the end of the graph section." At same time, 61% percent of respondees find lecture pacing "just right" or "a little too slow." My take: the material is more sophisticated now, we are relying on more tools from earlier than the course and making bigger leaps in lecture. In retrospect, it would have been good to speed up a little bit earlier in the course (e.g. 1 lecture's worth), so we could let things sink in a bit more now.
- PS8 reflection: active learning strategies
  - Teaching concepts to others, at office hours, in friend groups before midterms, etc.
  - Following the SREs and being very engaged as sender/receiver
  - Like that lecture notes aren't entirely prefilled, so forced to do active learning
  - Writing down questions to ask during office hours
  - Going up to Salil after class for any follow up questions
  - DCE Discord group for help / questions / answering

  - Paste the textbook into ChatGPT and have it ask me questions to gauge my understanding
  - Rewinding on videos when don't understand
  - Reading textbook before/after lecture
- SRE6 (Vector Subset Sum) feedback
  - Sender Feedback: Appreciated the more time. Able to get most proof details but didn't hash out specifics, which made partners wish they had more time
  - Receiver Feedback: Thought the exercise was pretty confusing overall, wish they can go into more details and worked with the senders to understand the notation better
  - Our takeaways: will simplify the SRE for future (reducing from Independent Set instead of 3-SAT), and you all should feel free to ask private questions on Ed while preparing for the SRE.

Recommended Reading:

- Hesterberg–Vadhan 26
- MacCormick 6.0, 7.0–7.6

2 One-sided solutions to decision problems

Let us recall what it means to *solve* a decision problem:

Definition 2.1. Let $\Pi = (\mathcal{I}, \{\texttt{yes}, \texttt{no}\}, f)$ be a decision problem and let Q be a RAM program. We say that Q *solves* Π if the following hold:

In the previous class, we gave (without proof) our first example of an *unsolvable problem*, which cannot be solved by any algorithm whatsoever:

Input: A RAM program P and an input x
Output: `yes` if _____, `no` otherwise

Computational Problem Halting Problem–RAM

Today, we will *prove* that this problem can't be solved by any algorithm. Before diving into the proof, it is very useful to look at a weaker notion of solving a problem.

Definition 2.2. Let $\Pi = (\mathcal{I}, \{\texttt{yes}, \texttt{no}\}, f)$ be a decision problem and let Q be a RAM program. We say that Q *one-sided solves* Π if for every $x \in \mathcal{I}$,

- If $f(x) = \{\texttt{yes}\}$, then _____
- If $f(x) = \{\texttt{no}\}$ *and* _____, then _____

This is not really a satisfactory notion of solving a computational problem, because if Q has not halted after we run it for a long time, we don't know whether or not it eventually will halt and give us an answer.

Q: If a RAM program Q always halts and one-sided solves Π, then does it solve Π?

A:

Examples:

- Here is a one-sided algorithm for Halting Problem–RAM.

- Consider the following weird decision problem, which asks what a program does if we run it on itself.

Input: A RAM program P **Output:** `yes` if $P(P) =$ `no`; `no` if $P(P) =$ `yes`; `no` if $P(P)$ does not halt.

Computational Problem Reject Self–RAM

A RAM program that one-sided solves this is as follows:

- A less-contrived example from number theory is a problem about polynomial equations.

> **Input:** A (multivariate) polynomial $p(x_0, x_1, \ldots, x_{n-1})$ with integer coefficients
>
> **Output:** `yes` if there exist positive integers $\alpha_0, \alpha_1, \ldots, \alpha_{n-1} \in \mathbb{N}$ such that $p(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = 0$, `no` otherwise

Computational Problem DIOPHANTINE EQUATIONS

Fermat's Last Theorem (a conjecture of mathematician Pierre de Fermat from 1637) concerns instances of this problem, with $n = 3$ and $p(x_0, x_1, x_2) = x_0^k + x_1^k - x_2^k$. (For $k = 2$, solutions are Pythagorean triples, so the answer is `yes`. Andrew Wiles famously proved in 1995 that each instance of that problem corresponding to an integer $k \geq 3$ has the answer `no`, i.e. the equation has no solutions.)

For univariate polynomials $p(x) = ax^2 + bx + c$ of degree 2, there *is* an algorithm to solve this problem; namely by checking whether either of the roots $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ lies in $\mathbb{Z}^+$, but solving DIOPHANTINE EQUATIONS is in general more difficult—for instance, the youngest tenured professor in Harvard's history, Noam Elkies, is best-known for his work on them.

Here is a RAM program to one-sided solve DIOPHANTINE EQUATIONS.

- Another example is that of tiling a plane with squares.

> **Input:** A finite set T of square tiles with each of the four edges colored (using an arbitrarily large palette of colors)
>
> **Output:** `yes` if the entire 2-d plane can be tiled using tiles from T so that colors of abutting edges match, `no` otherwise

Computational Problem TILING

There is a RAM program that one-sided solves $\overline{\text{TILING}}$, the problem of TILING with `yes` and `no` answers switched. It proceeds by searching over larger and larger square regions of the plane and trying all tilings within these regions. If any of them, the tiling is not possible (the `yes` case of $\overline{\text{TILING}}$), and the program can return `yes` and stop. If every finite-size square region can be tiled, then a tiling of the whole infinite plane is possible (a surprisingly subtle statement to prove!), so the algorithm will run forever only in the `no` case.

3 The HALTING PROBLEM is unsolvable

We now formally show the unsolvability of the HALTING PROBLEM. Importantly, the theorem is about solving the HALTING PROBLEM in the usual sense, rather than one-sided solving, which we saw above was doable.

Theorem 3.1 (Turing, 1936)**.** *There is no algorithm that solves* HALTING PROBLEM–RAM*.*

Turing actually proved that there is no algorithm which computes HALTING PROBLEM–TM; the statement above for the RAM version is equivalent to it (as discussed last time).HALTING PROBLEM has a special place among problems that are one-sided solvable, due to the following lemma.

Lemma 3.2. *Let* $\Pi = (\mathcal{I}, \{\texttt{yes}, \texttt{no}\}, f)$ *be a decision problem that is one-sided solvable. Then* Π *reduces to* HALTING PROBLEM–RAM*.*

Proof. Let Π be one-sided solvable by a RAM program Q. The reduction algorithm A is as follows:

```
  Red(x):
0
```

Algorithm 3.1: Reduction from Π that is one-sided solved by Q to the HALTING PROBLEM

□

Q: What is this reminiscent of?

A:

The class of decision problems that can be one-sided solved is called RE (the "recursively enumerable" problems).

We are now in a position to prove Theorem 3.1.

Proof. Our strategy is as follows. We will show that Reject Self–RAM is unsolvable. Then Lemma 3.2 shows that Halting Problem–RAM is also unsolvable.

Formally,

□

The self-contradictory nature of R is a version, in the language of computational problems, of the of the paradoxical sentence "This sentence is false" or the paradoxical set $\{S : S \notin S\}$ of all sets that don't contain themselves. It was first discovered as a version of the "Cantor's diagonalization" argument used to prove that the real numbers are uncountable.

4 Natural unsolvable problems

The phenomenon of unsolvability is not limited to problems about programs.

- "Almost all" problems are unsolvable.
- Hilbert's Program to Formalize and Mechanize Mathematics (and his 23 Problems for Mathematics in the 20th Century, 1900 lecture at International Congress of Mathematicians)
  - Prove consistency of the axioms of mathematics (Hilbert's 2nd Problem): impossible by Gődel's Incompleteness.
  - Algorithm to decide truth of mathematical statements (Hilbert and Ackermann's Entscheidungsproblem): impossible by Turing's work (unsolvability of Halting Problem).

  - Algorithm for DIOPHANTINE EQUATIONS (Hilbert's 10th problem): proved impossible in 1970's.

- TILING is unsolvable.