CS120: Intro. to Algorithms and their Limitations Anshu & Hesterberg

Lecture 24: Uncomputable problems

Harvard SEAS - Fall 2023 *2023-11-30*

1 Announcements

Recommended Reading:

- MacCormick Ch. 3 and §7.7–7.9

Announcements:

- 3 late days for SRE.
- No late penalties for participation survey.
- No revision videos for pset 9. All revision videos due by 12/8.

2 One-sided solutions to decision problems

In lecture 2, we defined what it means for an algorithm to solve a computational problem. Let us restate this definition, focusing on decision problems and using RAM programs as models of our algorithms. Note that RAM programs are not crucial here - we could have used Word-RAM programs, Python, C, etc to model our algorithms (recall Lecture 17).

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

- For every $w \in \mathcal{I}$ such that $f(w) = \texttt{yes}$, $P(w)$ halts and returns `yes`.
- For every $w \in \mathcal{I}$ such that $f(w) = \texttt{no}$, $P(w)$ halts and returns `no`.

Last time we stated our first example of an *unsolvable problem*, which can be solved by no algorithm.

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

Computational Problem Halting Problem

Today, we will see why this problem can't be solved by any algorithm. We will also see other decision problems with same property. Before delving 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 P be a RAM program. We say that P *one-sided solves* Π if

- For every $w \in \mathcal{I}$ such that $f(w) = \texttt{yes}$, $P(w)$ halts and returns `yes`.

- For every $w \in \mathcal{I}$ such that $f(w) =$ `no`, if $P(w)$ halts then it returns `no`.

In other words, for a RAM program P to one-sided solve Π, it is only required that P halts and gives correct answer in the `yes` case. In the `no` case, P is allowed to not halt - but if it halts then it must output correctly.

Concept-check question: If a RAM program P always halts and one-sided solves Π, then does it solve Π?

Here are some examples:

- Here is a one-sided algorithm for the Halting problem. It makes use of the universal RAM program U that can simulate any RAM program on any input.

```
OnesidedSolve-Halt($P, x$):
Run $U(P, x)$;
return yes;
```

- Another example is the following weird decision problem.

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 RejectSelf

The RAM program to one-sided solve this is as follows.

```
Onesidedsolve-RejectSelf($P$):
Run $U(P, P)$;
if $U(P, P) = no$ then
    return yes
```

- Above two are somewhat contrived problems about programs. Lets look at more interesting examples. The first one 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 are natural numbers $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ such that $p(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = 0$, `no` otherwise

Computational Problem Diophantine Equations

Fermat's last theorem is an instance of this problem, with $n = 3$ and $p(x_0, x_1, x_2) = x_0^k + x_1^k - x_2^k$ (Andrew Wiles famously proved that there are no solutions to this instance when $k \geq 3$). 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 $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ lie in $\mathbb{N}$. Here is a RAM program to one-sided solve Diophantine Equations.

```
Onesidedsolve-Diophantine(multivariate polynomial $p$):
foreach $\ell = 0, 1, 2, \ldots$ do
    foreach $x_0, x_1, \ldots x_{n-1}$: $x_0 + x_1 + \ldots x_{n-1} = \ell$ do
        if $p(x_0, x_1, \ldots x_{n-1}) = 0$ then
            return yes
```

It searches over all tuples with the hope of setting the polynomial to 0. If polynomial is never 0, the algorithm will never halt.

- 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	**:** `no` if the entire 2-d plane be tiled using tiles from T so that colors of abutting edges match, `yes` otherwise

Computational Problem Tiling

There is a RAM program that one-sided solves Tiling. It proceeds by searching over larger and larger square regions of 2-d plane and trying all tilings within these regions to see if tiling fails. If the tiling is not possible (the `yes` case), then this will be seen in some large enough square region and the program will stop. If the tiling is possible (the `no` case), the algorithm will run forever.

3 The Halting Problem is unsolvable

Our goal is to prove the following theorem. A word of caution that the theorem is about solving the Halting problem, rather than one-sided solving the Halting problem. The latter is doable - as we saw above.

Theorem 3.1. *There is no algorithm (modelled as a RAM program) that solves the Halting Problem.*

We will soon see that RejectSelf is also unsolvable. Further ahead, we will mention that Diophantine Equations and Tiling are all unsolvable. But Halting problem has a special place among these unsolvable problems, due to the following claim.

Claim 3.2. *Suppose there is a RAM program P that solves the Halting problem. Let $\Pi = (\mathcal{I}, \{\texttt{yes}, \texttt{no}\}, f)$ be a decision problem that is one-sided solved by a RAM program Q. Then there is a RAM program R that solves Π.*

Proof. We construct the RAM program R as follows.

```
R:
Input   : $w \in \mathcal{I}$
Output  : yes if $f(w) =$ yes, no if $f(w) =$ no
if $P(Q, w) = no$ then
    return $no$;
if $P(Q, w) = yes$ then
    return $U(Q,w)$;
```

The program uses P to check if Q halts on w. By the one-sided property, if Q does not halt on w, then $f(w)$ must be no. If Q halts on w, then we can simply run Q on w - using the universal RAM program - and output the result. □

The claim shows - using Lemma 4.2 from last lecture - that if the Halting problem were solvable, then all of RejectSelf, Diophatine Equations, Tiling would be solvable. Thus, the claim gives us a reduction from these computational problems to the Halting problem. This is very reminiscent of NP-complete problems. The class of decision problems which can be one-sided solved is called RE; and Halting is complete for this class.

Now we are in a position to prove Theorem 3.1.

Proof. Suppose for contradiction that there is a RAM program P that solves the halting problem. Then there is a RAM program R that solves RejectSelf, due to Claim 3.2.

Now we show that no RAM program solves RejectSelf, which leads to contradiction. Suppose R solves RejectSelf - observe that this forces R to halt on all inputs. Further, $R(P) =$ no if $P(P) =$ yes AND $R(P) =$ yes if $P(P) =$ no or if $P(P)$ does not halt. But what happens if $P = R$? We find that $R(R) =$ no iff $R(R) =$ yes, which is not possible. Thus, R does not exist.

□

The self-contradictory nature of R is very related to the paradoxical sentence "This sentence is false" and also the paradoxical set of all sets that don't contain themselves $\{S : S \notin S\}$. It comes out of "Cantor's diagonalization" argument very similar as the one used to prove that the real numbers are uncountable.

3.1 Diophantine Equations and Tiling are also unsolvable

The phenomenon of unsolvability is not limited to problems about programs. Indeed, there's a sense in which *almost all* computational problems are unsolvable: it can be shown that there are "uncountably many" decision problems over any infinite set $\mathcal{I}$ of inputs, but there are only "countably many" RAM programs P, so most decision problems cannot be solved by any RAM program.

But it's more interesting and useful to identify natural examples of unsolvable problems. Diophantine Equations introduced earlier is a striking example. This problem was posed by Hilbert in a famous address to the International Congress of Mathematicians in 1900, as part of a list of 23 problems that Hilbert laid out as challenges for mathematicians to tackle in the 20th century. Several of Hilbert's problems were part of a general project to fully formalize and mechanize mathematics. Hilbert's 2nd problem was to prove consistency of the axioms of mathematics, which was shown to be impossible by Gődel's Incompleteness Theorem in 1931. Turing's work on the undecidability of the Halting Problem was also motivated by Hilbert's program, and was used by Turing to show that there is no algorithm to decide to truth of well-defined mathematical statements (since whether or not a Turing machine halts on a given input is a well-defined mathematical statement), resolving Hilbert's Entscheidungsproblem ("Decision Problem," posed by Hilbert and Ackermann in 1928). Hilbert's 10th problem asks about a very special case of the Entscheidungsproblem, and was finally resolved in 1970 through the work of several mathematicians:

Theorem 3.3 (Matiyasevich, Robinson, Davis, Putnam)**.** *Diophantine Equations is unsolvable.*

Tiling is yet another example of unsolvable problem.

Theorem 3.4 (Wang, Berger)**.** *Tiling is unsolvable.*

The tiling problem has a beautiful theory that dates back to Persian architectures, with more recent surprises being Penrose's aperiodic tiling. See here and here for nice optional reads.