CS1200: Intro. to Algorithms and their Limitations Anshu & Vadhan

Sender–Receiver Exercise 7: Reading for Senders

Harvard SEAS - Fall 2024 *Nov. 21, 2024*

The goals of this exercise are:

- to develop your skills at understanding, distilling, and communicating proofs and the conceptual ideas in them,
- to practice reductions for proving unsolvability, and gain more intuition for what kinds of problems about programs are unsolvable.

Sections 1 and 3 are also in the reading for receivers. Your goal will be to communicate the *proof* of Theorem 1.1 (i.e. the content of Section 2) to the receivers. Section 3 contains questions for you and your receiver to think about if you finish the exercise early; there is no need to prepare anything in advance for that.

1 The Result

In class, we mentioned Rice's Theorem, which says that all nontrivial problems about the input–output behavior of programs (i.e. about the program's semantics) are unsolvable.

Here we will see an example of a computational problem that is not about the input–output behavior of programs but is nevertheless unsolvable:

Input	**:** A RAM program P
Output	**:** `yes` if P has running time $O(n)$, `no` otherwise

Computational Problem IsLinearTime

The statement "P has running time $O(n)$" means that there are constants c and n_0 such that for all $n \geq n_0$ and all inputs x of length at most n, $P(x)$ halts within $c \cdot n$ steps. Note that the constants c and n_0 are allowed to depend on P.

Theorem 1.1. *IsLinearTime is unsolvable.*

2 The Proof

By Lemma 4.2 (about using reductions to prove unsolvability) and Theorem 5.1 (about the unsolvability of HaltsonEmpty-RAM) from Lecture 22, it suffices to prove that HaltsOnEmpty-RAM $\leq$ IsLinearTime. That is, we need to give an algorithm A that can decide whether a program P halts on ε using an oracle for IsLinearTime. The template for this reduction A is as usual:

```
1 A(P):
  Input   : A RAM program P
  Output  : yes if P halts on ε, no otherwise
2 Construct from P a program Q_P such that whether or not Q_P runs in time O(n) will tell
    us whether or not P halts on ε;
3 Feed Q_P to the IsLinearTime oracle, and use the result to decide whether to output yes or
    no;
```

Algorithm 1: Template for reduction from HaltsOnEmpty-RAM to IsLinearTime

How can we construct Q_P from P? One idea is to have, for every input x of length n, $Q_P(x)$ run P on ε for up to n^2 steps. That way, if P does not halt on ε, then Q_P takes time $\Omega(n^2)$, but if P does halt on ε, then Q_P's execution will stop earlier.

In more detail (but still pseudocode rather than formal RAM code):

```
1 Q_P(x):
2 Let n = input_len;
3 foreach i = 0 to n − 1 do
4     M[i] = 0
5 input_len = 0;
6 Run P for upto n^2 steps (unless it halts sooner)
```

Algorithm 2: The RAM program Q_P constructed from P

The commands of Q_P before Line 6 are to set up the configuration of memory and `input_len` to correspond to input ε, so that we faithfully simulate P on ε

Claim 2.1. *Q_P runs in time $O(n)$ if and only if P halts on ε.*

Proof. If P does not halt on ε, then the execution of P in Line 6 will always take at least n^2 steps, so Q_P does not have runtime $O(n)$.

Conversely, suppose that P does halt on ε, say in $t_0(P)$ steps. Note that $t_0(P)$ depends only on P, not on the input x given to Q_P. Let's now analyze the runtime of Q_P on an input x of length n. The loop for erasing the input before running P takes time $O(n)$. The simulation of P in Line 6 will take time $O(n^2)$ if $n^2 \leq t_0(P)$ and will take time $O(t_0(P))$ if $n^2 > t_0(P)$, where the $O(\cdot)$ is to allow extra time for a counter to make sure that we don't run P for more than n^2 steps. Thus, for all $n \geq t_0(P)^{1/2}$, the overall runtime of $Q_P(x)$ is

$$O(n) + O(t_0(P)) = O(n) + O(1) = O(n),$$

which means that Q_P runs in time $O(n)$. □

With this claim, we can fill in the details of our reduction from HaltOnEmpty-RAM to IsLinearTime:

```
1 A(P):
  Input   : A RAM program P
  Output  : yes if P halts on ε, no otherwise
2 Construct from P the program Q_P shown in Algorithm 6;
3 Feed Q_P to the IsLinearTime oracle, return whatever the oracle returns;
```

Algorithm 3: The Reduction from HaltOnEmpty-RAM to IsLinearTime

The correctness of the reduction A follows from Claim 2.1, and thus we conclude that IsLinearTime is unsolvable.

Tips for understanding the proof:

1. Remember that we are looking at asymptotic runtime of Q_P so think about what would happen when we fix P and then take n arbitrarily large.

2. You may find it helpful to sketch a graph of Q_P's runtime when P halts and when it does not. (Assume $t_0(P) = 100$) What is the asymptotic behavior of each graph?

3. The oracle IsLinearTime only takes in a program as an input. We don't care how the oracle works (it is just assumed to solve the problem in one time step) but it might be helpful to think about the mechanism as running Q_P on all (infinitely many) inputs x, graphing its worst-case runtime $T(n)$, and seeing if the function $T(n)$ is $O(n)$.

3 Food for Thought

If you and your partner(s) finish early, here are some additional questions or issues you can think about:

1. We constructed Q_P so that Q_P has running time $O(n)$ if and only if P halts on ε. Can you think of how to construct Q_P so that Q_P has running time $O(n)$ if and only if P *doesn't* halt on ε? If you used such a construction, how would the reduction A change?

2. So far, our intuition for unsolvability has been that it comes from the possibility that RAM programs don't halt. However, the programs Q_P constructed in the above reduction always halt in time $O(n^2)$. Thus, the same reduction proves unsolvability of the following variant of IsLinearTime, where we *promise* that the input program halts in time $O(n^2)$:

Input	**:** A RAM program P with running time $O(n^2)$
Output	**:** `yes` if P has running time $O(n)$, `no` otherwise

Computational Problem IsLinearTimePromise

Try to develop some of your own intuition for what makes a problem like this, on always-halting programs, unsolvable.