CS120: Intro. to Algorithms and their Limitations	Prof. Salil Vadhan

Active Learning Exercise 7: Reading for Receivers

Harvard SEAS - Fall 2021 — *Nov. 16, 2021*

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.

To prepare for this exercise as a receiver, you should try to understand the theorem statement and definition in Section 4 below, and review the material on Unsolvability covered in class on November 9. Your partner sender will communicate the proof of Theorem 4.1. Section 6 contains questions for you and your sender to think about if you finish the active learning exercise early; there is no need to prepare anything in advance for that.

1 The Result

In class, we saw 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	**:** `accept` if P has running time $O(n)$ on inputs of length n, `reject` otherwise

Computational Problem IsLinearTime

Theorem 1.1. *IsLinearTime() is unsolvable.*

2 The Proof

The approach to proving unsolvability.

The reduction.

Correctness of the reduction.

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	**:** `accept` if P has running time $O(n)$, `reject` otherwise

Computational Problem IsLinearTimePromise

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

CS120: Intro. to Algorithms and their Limitations	Prof. Salil Vadhan
Active Learning Exercise 7: Reading for Senders	
Harvard SEAS - Fall 2021	*Nov. 16, 2021*

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 4 and 6 are also in the reading for receivers. Your goal will be to communicate the *proof* of Theorem 4.1 (i.e. the content of Section 5) to the receivers. Section 6 contains questions for you and your receiver to think about if you finish the active learning exercise early; there is no need to prepare anything in advance for that.

4 The Result

In class, we saw 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 : `accept` if P has running time $O(n)$ on inputs of length n, `reject` otherwise

Computational Problem IsLinearTime

Theorem 4.1. *IsLinearTime() is unsolvable.*

5 The Proof

By Lemma 2.4 and Theorem 3.1 from Lecture 20, it suffices to prove that HaltOnEmpty $\leq$ 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 : `accept` if P halts on ε, `reject` 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 `accept` or `reject` P;

Algorithm 1: Template for reduction from HaltsOnEmpty to IsLinearTime

How can we construct Q_P from P? One idea is to have, for every input x, $Q_P(x)$ run P on ε for up to $t(n)$ steps where $n = \text{length}(x)$. If P halts on ε, say within t_0 time steps, then this will stop after $O(\min\{t(n), t_0\}) = O(1)$ steps of P. On the other hand, if P doesn't halt on x, then this simulation will always take time $t(n)$. Thus, taking $t(n) = n^2$, whether or not Q_P runs in time $O(n)$ will depend on whether or not P halts on ε, as desired.

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) ;        /* like done in PS10 */
```

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 $P(\varepsilon)$.

Claim 5.1. *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, if P does halt on ε, then Line 6 will take time $O(\min\{n^2, t_0\})$, where t_0 is the time that P takes to halt on ε and the $O(\cdot)$ encompasses the overhead for doing a timed execution of P (as you are showing in PS10). Thus, $Q_P(x)$ will have runtime $O(n) + O(\min\{n^2, t_0\}) = O(n)$. ☐

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

```
A(P):
   Input  : A RAM program P
   Output : accept if P halts on ε, reject otherwise
Construct from P the program Q_P shown in Algorithm 6;
Feed Q_P to the IsLinearTime oracle, return whatever the oracle returns;
```

Algorithm 3: The Reduction from HaltOnEmpty to IsLinearTime

The correctness of the reduction A follows from Claim 5.1, and thus we conclude that IsLinearTime is unsolvable.

6 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	**:** `accept` if P has running time $O(n)$, `reject` otherwise

Computational Problem IsLinearTimePromise

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