#1 "Lecture" Notes

There are two ways we commonly show undecidability: diagonalization and reduction. We will see both today. Because everything we are doing today is nothing strictly new to you, I am going to try something a little different and "invert" the classroom a little. I would like for us to focus mostly (entirely?) on your lab. I will first introduce the problems, and then you may work in groups together for a bit. Then we'll go over each lab question when it feels like everyone is mostly done with that section. (So, first do #2, and then we will try to go over it.)

These exercises are designed to help you understand Sipser §4.2 on pp 207-209 and Sipser §5.1 pp 216-217. The last problem (optional) helps you through Sipser §5.1 pp 217-218.

#2 Our first undecidability proof

Consider the following language.

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts} w\}.$$

The next series of short questions are designed to make you think and tinker with the proof that A_{TM} is undecidable (Sipser pp. 207).

1. Towards a contradiction, we assume A_{TM} is decidable. This means we may assume the TM H decides it, where H has the following behavior.

$$H(\langle M, w\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

   Suppose M' is a TM that loops forever on input 0101.

   (a) What is the behavior, then, of $H(\langle M', 0101\rangle)$? Does it accept, reject, or run forever?
   (b) Recall that the difference between a decider and a TM is that a decider always halts. Briefly describe how we could use H to turn *any* regular old TM into a decider.

2. In the proof, we next construct a new TM D specified as follows.

   (a) D takes input $\langle M\rangle$, where M is a TM.
   (b) Run H on input $\langle M, \langle M\rangle\rangle$.
   (c) Output the opposite of what H outputs. That is, if H accepts, reject; if H rejects, accept.

   In other words, our assumption of the machine H let's us build a TM D with this behavior.

$$D(\langle M\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M\rangle \end{cases}$$

   Suppose M_1, M_2, and M_2 each take as input Turing machines, and that

   - $M_1(\langle M_1\rangle)$ accepts,

- $M_2(\langle M_2 \rangle)$ rejects, and
- $M_3(\langle M_3 \rangle)$ runs forever, and

What is the output, then, of $D(\langle M_1 \rangle)$, $D(\langle M_2 \rangle)$, $D(\langle M_3 \rangle)$?

3. Finally, consider $D(\langle D \rangle)$, the result of running D on itself. Describe why such a computation is a contradiction.

#3 Back to diagonalization

4. Figure 4.19 has a table where *all* the Turing Machines that could ever exist (!) are listed as columns and rows. Why is this possible? That is, why is it mathematically valid to enumerate all such TMs?

5. Describe, in Figure 4.21, how D diagonalizes the set of all Turing machines. Compare and contrast this table with the proof that the real numbers are uncountable on pp. 205 (or see lecture notes 8B).

1. (a) It will reject.
   (b) Given arbitrary M, build a new TM M' as follows. On input string w, M' runs $H(\langle M, w \rangle)$ and accepts/rejects based on this run. Now, if the input w were to cause M to run forever, suddenly it rejects! So M is a decider.
2. accepts, rejects, rejects.
3. If we have such a D, then $D(\langle D \rangle)$ will accept if D does not accept $\langle D \rangle$ and reject if D accepts $\langle D \rangle$. But this breaks the behavior we first ascribed to D! Classically, we might say that "if D accepts $\langle D \rangle$ then D does not accept $\langle D \rangle$, and if D does not accept $\langle D \rangle$ then D accepts $\langle D \rangle$." Read this out loud and you will hear the contradiction.
4. Because the set of all TMs is countably infinite.
5. If we suppose that H can "fill in the blanks" of our table 4.19, then we can build a new TM D that cannot be in such a table!

#4 Our second undecidability proof

Consider the following language:

$$\mathsf{HALT}_{\mathrm{TM}} = \{ (M, w) \mid \text{Turing machine } M \text{ halts on input } w \}.$$

By "halts", we mean that the Turing machine either accepts or rejects. Adapt the diagonalization proof that A_{TM} is undecidable from the end of chapter 4.2 to show that $\mathsf{HALT}_{\mathrm{TM}}$ is undecidable.

Build your proof according to these steps.

1. Assume for the sake of contradiction that $\mathsf{HALT}_{\mathrm{TM}}$ is decidable. Let H decide it.

2. Define a Turing machine D that takes another TM M as input. D should use H on input $\langle M \rangle$. What should it do when $H(\langle M \rangle)$ accepts? What should it do when $H(\langle M \rangle)$ rejects?

3. Consider, again what happens when we run D on $\langle D \rangle$.

Assume for the sake of contradiction that $\mathsf{HALT}_{\mathrm{TM}}$ is decidable; let H decide it. Define Turing machine D that takes another Turing machine M as input as follows.

1. D runs H on input $\langle M \rangle$.
2. If H accepts, go into an infinite loop.
3. If H rejects, then accept.

Now observe what happens when we run D on $\langle D \rangle$.

1. If D accepts $\langle D \rangle$ then it was because H rejected $\langle D \rangle$, implying D went into an infinite loop on itself.
2. If D goes into an infinite loop on $\langle D \rangle$ then it was because H accepted $\langle D \rangle$, implying D halted on itself.

Both cases induce a contradiction, so our original assumption that $\mathsf{HALT}_{\mathrm{TM}}$ was decidable must be incorrect.

#5 Our n^{th} undecidability proof

Show that $\mathsf{HALT}_{\mathrm{TM}}$ is undecidable by reducing $\mathsf{HALT}_{\mathrm{TM}}$ to A_{TM}. Hint: consider your answers to 1a and 1b.

Your proof should follow roughly these steps.

1. Towards a contradiction, assume that $\mathsf{HALT}_{\mathrm{TM}}$ is decidable. Call the TM that decides it R.
2. Construct a new TM S that decides A_{TM}.
3. Argue (briefly) how S decides A_{TM}. Then, conclude that, as we know A_{TM} to be undecidable, so too must $\mathsf{HALT}_{\mathrm{TM}}$.

See the Proof on Sipser pp. 217.

#6 (Optional) Our $n + 1^{th}$ undecidability proof

(Please work on this problem if you finish the others before your peers.)

Let's do a slightly more involved reduction, following Sipser Theorem 5.2. Consider the following language of Turing machines.

$$E_{\mathrm{TM}} = \{\langle M \rangle \mid L(M) = \emptyset\}$$

That is, M is a Turing machine whose language is empty, and E_{TM} is the set of all such M (encoded as strings). We want to prove that E_{TM} is undecidable by a reduction to A_{TM}.

1. The strategy is this: let R be the TM that decides E_{TM}. Use R to construct TM S that decides A_{TM}. Let's brainstorm some ideas. How will S work when it receives input $\langle M, w \rangle$? How can S use R to decide A_{TM}. *Think for a bit and write down whatever strategies you think might work.*
2. After some brainstorming above, you may have come to the conclusion that it isn't quite so simple. (Or: after reading ahead, you now know it is not so simple). We need some more advanced machinery to solve this one.

Instead of running R on $\langle M \rangle$, we will run R on a *modification* of $\langle M \rangle$. In particular, let's build another machine called M_1—for which the input string w in $\langle M, w \rangle$ is fixed—that takes as input another string x. Specifically, given (separate) input $\langle M, w \rangle$ to the machine S, M_1 is constructed as follows.

(a) M_1 takes string input x.

(b) M_1 checks if $x = w$. If $x \neq w$, reject.

(c) If $x = w$, run M on input w and accept if M accepts w.

Finally, use M_1 and R (the TM that decides E_{TM}) to build a TM S that decides A_{TM}. Then conclude that E_{TM} must be undecidable.

See Sipser pp. 218.