CS 361 Meeting 25 — 4/27/20

A Recognizable, but Undecidable Language

(Click for video)

1. Last class, I presented a brief, somewhat inscrutable proof that the language

$$A_{BTM} = \{\langle M\rangle w \mid \langle M\rangle \text{ is a binary encoding of a binary TM, \& } w \in \mathcal{L}(M).\ \} \subset \{0,1\}^*$$

which is just the language A_{TM} restricted to Turing machine's with binary input alphabets, is not decidable.

Theorem: A_{BTM} is undecidable.

Proof: Suppose that A_{BTM} was decidable. Then there would exist some TM N that always halted such that $A_{BTM} = L(N)$.

- Given N, we could construct another TM D which on any input w, made a copy of w after its original input to form ww and then ran N on the result. This machine would decide the language

$$L(D) = \{\langle M\rangle \mid \langle M\rangle \text{ is an encoding of a binary TM } \& \langle M\rangle \in \mathcal{L}(M).\ \}$$

- Now, suppose that we alter D just a bit to produce a new machine named $\overline{D}$. $\overline{D}$ will be identical to D except its accept and reject states will be interchanged. Since all of these machines are deciders, we can say

$$L(\overline{D}) = \{\langle M\rangle \mid \langle M\rangle \text{ is not an encoding of a binary TM or } \langle M\rangle \notin \mathcal{L}(M).\ \}$$

- Now, consider what happens when we apply $\overline{D}$ to its own description. That is, we apply $\overline{D}$ to the input $\langle\overline{D}\rangle$. Since $\langle\overline{D}\rangle$ is clearly an encoding of a binary TM, we can see that $< \overline{D} > \in L(\overline{D}) \equiv < \overline{D} > \notin L(\overline{D})$.
- This is nonsense! Or better yet a contradiction. As a result, we can state that our original assumption that A_{BTM} was decidable must be false.

Click here to view the slides for this class

What Diagonal?

(Click for video)

1. The proof that A_{BTM} is undecidable is short and each step is quite simple and undebatable. At the end, however, it feels a bit more like a magic trick than a proof. Therefore, it is worth taking some time to understand what it says in a different way.
2. The proof that A_{BTM} is undecidable is described as a diagonalization proof.
3. You may (or may not!) recall that on the first day of class we used a diagonalization argument to show that there were more reals than integers.
   - We assumed that there was a mapping from the natural numbers to the reals. That is, that there was some list that included every real number in such a way that we could identify some real as number 1, some real as number 2, and so on.
   - We then described a real number that could not be in this list by stipulating that we would select the ith digit of the decimal expansion of our number to be different from the ith bit of the ith number in our list of reals.
   - Since a real number constructed in this way could not be in our list, this contradicted the assumption that such a list could exist.
   - To see the diagonal in this argument, consider the following table which is supposed to show the first few digits of each of the first few real numbers in some potential numbered list of all real numbers.

			1	2	3	4	5	6	7	...
1)	3	.	**1**	4	1	5	9	2	6	...
2)	2	.	7	**1**	8	2	8	1	8	...
3)	2	.	9	9	**7**	9	2	4	5	...
4)	1	.	4	1	4	**2**	1	3	5	...
5)	1	.	7	3	2	0	**5**	0	8	...
6)	0	.	3	3	3	3	3	**3**	3	...
7)	4	.	0	0	0	0	0	0	**0**	...
8)	I	.	$d_{8,1}$	$d_{8,2}$	$d_{8,3}$	$d_{8,4}$	$d_{8,5}$	$d_{8,6}$	$d_{8,7}$	...

The i digit of the number we construct to show that such a table cannot contain every real number is chosen to be different from $d_{i,i}$, the ith digit of the ith number. These are the numbers found along the diagonal of this table (if you ignore any portion of one of the real numbers shown before the decimal point).

- Note that it is not necessary for each real to appear just once in the purported enumeration of all reals to make this work. Even if some one real number appeared infinitely often, the argument would still work as long as we assumed every real appeared at least once.

4. The machine D we constructed in our proof that A_{BTM} is undecidable can be viewed as diagonalizing over a list of all TMs in the same way we formed a real number by diagonalizing over a purported list of all reals.

5. We can imagine an infinite table whose rows correspond to TMs in some ordering and whose columns correspond to binary encodings of these same Turing machines.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	...
M_1	**reject**	reject	reject	reject	reject	reject	...
M_2	reject	**reject**	reject	reject	reject	reject	...
M_3	accept	reject	**accept**	reject	reject	accept	...
M_4	accept	accept	accept	**accept**	accept	accept	...
M_5	reject	reject	reject	reject	**reject**	reject	...
M_6	reject	accept	reject	accept	reject	**reject**	...
M_7	accept	accept	reject	reject	reject	accept	...
M_8	reject	accept	reject	accept	reject	accept	...

Each cell indicates whether the TM for the row accepts the input corresponding to the encoding of the TM for that column.

6. The machine D we described in our proof that A_{BTM} is undecidable corresponds to the list of accept/reject results listed along the diagonal of this table (all shown in bold font).

7. If the machine $\overline{D}$ we described in our proof by contradiction existed, its language $L(\overline{D})$ would be described by the opposite of the sequence of "reject"s and "accept"s found along the diagonal of this table. Thus, it will be different from every row of the table in at least one position, but all TMs are included in the rows of this table so no such TM could exist.

8. Unlike the diagonalization proof that Cantor used to show that the reals were not countable, the construction of the machine $\overline{D}$ does not lead to the conclusion that the set of TMs cannot be enumerated. We know that TMs can be enumerated! Instead, it leads to the conclusion that $\overline{D}$ cannot be part of the list of all TMs. That is, $\overline{D}$ does not exist.

Reduction

(Click for video)

1. With regular languages and context-free languages, the appropriate pumping lemma was used over and over again to show that languages did not belong to the class in question.

2. The diagonalization technique is not used repeatedly like the pumping lemmas. Instead, we use the fact that A_{BTM} is now known to be recognizable but undecidable together with closure properties and reductions to show almost all other similar results.

3. As a first example, we can show that it is not necessary to restrict our attention to machines with binary input alphabets encoded in binary. Instead, we can consider the broader language.

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } w \in \mathcal{L}(M)\}$$

   - Suppose that A_{TM} were decidable. In this case, there would be some machine M that decided A_{TM}.

- The definitions of both A_{TM} and A_{BTM} would include schemes for encoding TM descriptions using a finite input alphabet. Suppose we constructed a machine M' with a binary input alphabet that first rejected its input if it was not a valid binary encoding of a binary TM and its input, $\langle B \rangle w$ using the scheme associated with A_{BTM}. Otherwise, it would translate its input into an encoding of the same TM M using the encoding for A_{TM} and then run M.
- Since we assumed M decided A_{TM}, M' would have to decide A_{BTM}. We just proved, however, that it is impossible to decide A_{BTM}. Based on this contradiction, we can conclude that no decider for A_{TM} exists. That is, we have shown that A_{TM} is undedicable.

4. Using diagonalization and reduction, we have now shown that A_{BTM} and A_{TM} are not decidable, but both of these languages are recognizable.
5. The last bubble in our Venn diagram that we need to fill with an example is the area for languages that are not recognizable. This turns out to be easy.
6. Earlier I pointed out that if both a language and its complement are recognizable then they must both also be decidable.
   - If both A and $\overline{A}$ are recognizable, they must be recognized by some pair of machines M and $\overline{M}$.
   - We can build a machine that simulates both of these machines in parallel on the same input (it should be easy to see how to do this on a two-tape TM).
   - We can then decide A by accepting if the simulation of M accepts and rejecting if the simulation of $\overline{M}$ accepts. One of the two must happen eventually.
7. As a result, now that we know that A_{BTM} and A_{TM} are not decidable we can immediately conclude that $\overline{A_{BTM}}$ and $\overline{A_{TM}}$ are not recognizable.

Reductio ad Nauseum

(Click for video)

1. The reduction technique used to extend our knowledge that A_{BTM} is undecidable to also knowing that A_{TM} is undecidable can be applied to a wide range of questions about decidable languages.

   In general:

   > If by assuming a Turing machine M decides some language B, we can describe how to build another Turing machine M' that decides A then:
   >
   > - if B is known to be decidable then A must also be decidable, and
   > - if A is know to be undecidable, then B must also be undecidable.

2. It can also be used to show that a language is or is not recognizable since:

   > If by assuming a Turing machine M recognizes some language B, we can describe how to build another Turing machine M' that recognizes A then:
   >
   > - if B is known to be recognizable then A must also be recognizable, and
   > - if A is know to not be recognizable, then B must also not be recognizable.

3. As another example of reduction, consider how we can show where:

   $$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } \mathcal{L}(M) = \emptyset\}$$

   fits in our classification scheme.

4. It should be fairly clear that the complement of this language

   $$\overline{E_{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } \mathcal{L}(M) \neq \emptyset\}$$

   is at least recognizable.

- A machine could recognize descriptions of TMs with non-empty languages by simulating instances of the machine on more and more possible inputs (using the dovetailing technique discussed in our explanation of recursively enumerable languages). If any of these concurrent simulations found a string that was accepted, the TM description would be accepted.

5. If both E_{TM} and $\overline{E_{TM}}$ are recognizable then they must both be decidable. This seems unlikely, so it seems likely that E_{TM} is unrecognizable. We could demonstrate this by either showing that $\overline{E_{TM}}$ is not decidable, or by using a reduction to directly show that E_{TM} is not recognizable. We will take the second approach.

6. Assume that E_{TM} is recognized by some TM M_E. We will show that we could use M_E as a subroutine to construct a larger machine $M_{\overline{A_{TM}}}$ that recognizes $\overline{A_{TM}}$. Since we know that $\overline{A_{TM}}$ is not recognizable, we can then conclude that M_E must not actually exist and therefore E_{TM} is not recognizable.

7. Consider the following description of a potential machine $M_{\overline{A_{TM}}}$.

   - On any input that is not of the form $< M, w >$ (i.e. a valid encoding of a Turing machine and input), accept the input.[1]
   - On input $< M, w >$, construct a description of a new machine M' that behaves as follows:
     - On any input w', ignore w' and instead simulate M on w. If M accepts w, then accept w', otherwise loop or reject just as the simulation does.
   - Use M_E, the machine that recognizes E_{TM} as a subroutine by applying it to the description of the new machine M'. If M_E accepts this machine's description, accept $< M, w >$, otherwise reject.

[1]This may seem counter-intuitive, but recall that because A_{TM} only includes strings of the form $< M, w >$, a machine that recognizes its complement must accept all such badly formed inputs.

8. The machine we have just described would recognize $\overline{A_{TM}}$ because the language of the machine M' we construct for the input $< M, w >$ will be empty exactly when $w \notin L(M)$ which is exactly when $< M, w > \in \overline{A_{TM}}$. We know that this is impossible. Therefore, the assumption that E_{TM} is recognizable must be false. E_{TM} must not be recognizable.

A Closer Look
(Click for video)

And a Little Bit More
(Click for video)

1. I want to explore the structure of the proof that E_{TM} is not recognizable very carefully. Reduction is such an important proof technique in this context that I want to make sure you are all clear about how the proof work.

   - This proof involve four different Turing machines and/or their descriptions. Keeping the roles of these machines very clear in your mind is essential to understanding the proof.
   - The first machine, M_E is what we assume to exist based on the (false) assumption that E_{TM} is recognizable. We do not assume anything about the structure of this machine. We better not because we believe it cannot exist!
   - $M_{\overline{A_{TM}}}$ is a machine we describe how to construct that would accomplish something we know is impossible to do by exploiting the (false) assumption that M_E exists. In most proofs of this form, the last thing the impossible machine we are trying to construct ($M_{\overline{A_{TM}}}$ in this case) will do is run M_E (or its equivalent).
   - As a somewhat silly analogy to illustrate the relationship between these two machines, think about "Mechanical Turk" — not Amazon Mechanical Turk but the device that inspired Amazon to use this name for its service.
     - The Mechanical Turk was a elaborate hoax/magic trick constructed almost 250 years ago.

- It appeared to be a machine that could play chess. It included a mechanized model dressed in Turkish garb with a movable arm. The model was seated at a large desk with a chess board on top and mysterious gears and other mechanisms inside that appeared to "compute" the Turks moves.
- In fact, the mechanisms inside did nothing more that provide room in the desk for the human who was really controlling the model to hide.
- The person inside is like the first TM in our list of four TMs involved in this (and most) reduction proofs. Even though the point of the proof is to show this machine could not exist, in our construction we are relying on it to actually do the hard work of implementing the "outer machine" which appears to be able to do the impossible (recognize $M_{\overline{A_{TM}}}$ or the equivalent).
- The best part about this analogy is the recognition that especially for its time, the construction of the turk was something of a bit of magic. To pull off this illusion, the designer had to build complex mechanisms that would enable the person hiding inside to see the other player's moves and to control the model to make its own moves.
- Similarly, in these reduction proofs, the trick is really the process of finding a way to convert any input provided to the "outer" machine (the one that would solve $M_{\overline{A_{TM}}}$), into a suitable input to the "inner" machine (the one we want to prove cannot exist).

* Given the Turk analogy, the last two machines in our proof that M_E is not recognizable (and in all similar proofs) are all about providing the linkage between the outer machine and the inner machine.

* M isn't really "a" machine. It can actually be any machine. We are trying to show that we could construct $M_{\overline{A_{TM}}}$ if M_E existed. $M_{\overline{A_{TM}}}$ takes an input composed of the description of a Turing machine and of a possible input to that Turing Machine. M is the name we use to refer to the machine described in some input to $M_{\overline{A_{TM}}}$ as we try to describe how $M_{\overline{A_{TM}}}$ would process its input. We never actually run or simulate M. Instead, we simply process its encoded description in various ways.

* M' is another machine we never actually run. Instead we argue that $M_{\overline{A_{TM}}}$ would be able to construct a **description** of M' (i.e., $< M' >$) on its tape. The form of this machine depends on the input to $M_{\overline{A_{TM}}}$ in such a way that $< M' >$ will belong to the language of M_E exactly when the input to $M_{\overline{A_{TM}}}$ belongs to $\overline{A_{TM}}$.