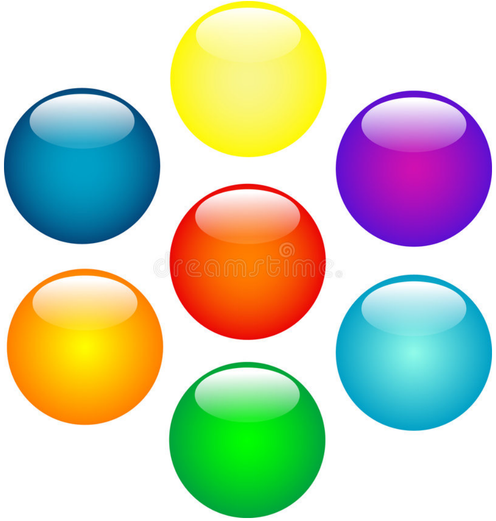
Undecidability, part II



Warm-up question (4/15/22)

What is your favorite walking area in Pullman (on or off campus)?

Send me your response by Canvas email.

HW 7 is out today

- Last homework set!
- Has 8 problems
  - 7 problems (mostly on decidability and reducibility)
    - 1 game-inspired problem
    - One of the problems involves learning about Cantor
  - 1 reflection question
- You have 10 days
  - Out: 4/15/22
  - Due: 4/25/22

In last lecture, we saw…

A_{TM} = {<M,w> | M is a TM and M accepts w}

- A_{TM} is Turing-Recognizable
  - Universal Turing Machine
- Countable and Uncountable sets
  - Correspondence
- The set of real numbers R is uncountable
  - The diagonalization method (Cantor)

Today, we will…

- Show that some languages are not Turing-recognizable
- Prove that A_{TM} is undecidable
- Exhibit an example of a language that is not Turing-recognizable

Some languages are not Turing-recognizable

- Underlying reason:
  - There are **uncountably many languages**
  - And only **countably many Turing Machines**

- To show that the set of all TMs is countable
  - First, observe that the set of all strings Σ^* is countable for any alphabet Σ
    - With only finitely many strings of each length, we may form a list Σ^* by writing down all strings of length 0, length 1, length 2, and so on
  - Each Turing TM M has an encoding into a string <M>

- To show that the set of all languages is uncountable (well, we need another slide, or two; see next)

The set of all languages is uncountable

- First, observe that the set of all infinite binary sequences is uncountable
  - (An *infinite binary sequence* is an unending sequence of 0s and 1s)
  - Let B be the set of all infinite binary sequences
  - We can show that B is uncountable by using a proof by diagonalization similar to the one used to prove the set of real R is uncountable

The set of all languages is uncountable

- Let L be the set of all languages over alphabet Σ
- We show that L is uncountable by giving a correspondence with B
- Let $\Sigma^* = \{s_1, s_2, \ldots\}$
- Each $A \in L$ has a unique sequence in B
- The ith bit of that sequence is a 1 if $s_i \in A$ and is a 0 if $s_i \notin A$
- This is called the characteristic sequence of A
- For example, if A were the language of all strings starting

with a 0 over the alphabet $\{0,1\}$, its characteristic sequence χ_A would be

$$
\begin{array}{rcccccccccccl}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots & \} & ; \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots & \} & ; \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots & & .
\end{array}
$$

The set of all languages is uncountable

- The function $f: L \rightarrow B$, where $f(A)$ equals the characteristic sequence of A, is one-to-one and onto (hence is a correspondence)

- Therefore, as B is uncountable, L is uncountable as well

- Thus we have shown that the set of all languages cannot be put into a correspondence with a set of all Turing machines

- We conclude that some languages are not recognized by any TM

A_{TM} is undecidable: proof

- We assume that A_{TM} is decidable and obtain a contradiction
- Suppose H is a decider for A_{TM} and w is a string
- H halts and accepts if M accepts w;

  H halts and rejects if M fails to accept w
- In other words, we assume that H is a TM, where

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

A_{TM} is undecidable: proof

- Now we construct a new TM D with H as a subroutine
- This new TM calls H to determine what M does when the input to M is its own description <M>
- Once D has determined this information, it does the opposite (i.e. it rejects if M accepts and accepts if M does not accept)

$D =$ "On input $\langle M \rangle$, where M is a TM:
1. Run H on input $\langle M, \langle M \rangle \rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

A_{TM} is undecidable: proof

In summary, we have:

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

What happens when we run D with its own description <D> as input? In that case, we get:

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

No matter what D does, it is forced to do the opposite, a contradiction. Thus neither TM D nor TM H can exist.

Let us review the steps of the proof we just saw

- Assume that a TM H decides A_{TM}
- Use H to build a TM D that takes an input <M>,
where D accepts its input <M> exactly when <M> does not accept its input <M>
- Finally, run D on itself
- Thus, the machine takes the following actions, with the last line being the contradiction.

- H accepts $\langle M, w \rangle$ exactly when M accepts w.
- D rejects $\langle M \rangle$ exactly when M accepts $\langle M \rangle$.
- D rejects $\langle D \rangle$ exactly when D accepts $\langle D \rangle$.

But where is the diagonalization in the proof?

- Becomes apparent when we examine the tables of behavior for TMs H and D
- In these tables, we list all TMs down the rows, and all their descriptions across the columns
- The entries tell whether the machine in a given row accepts the input in a given column
- The entry is accept if the machine accepts the input, but is blank if it rejects or loops on that input
- Example:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

But where the diagonalization in the proof?

Entry i,j is the value of H on input $<M_i, <M_j>>$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$		$\vdots$			

But where the diagonalization in the proof?

If D is in the figure, a contradiction occurs at ?

$$
\begin{array}{c|ccccccc}
 & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \cdots & \langle D \rangle & \cdots \\
\hline
M_1 & \underline{accept} & reject & accept & reject & & accept & \\
M_2 & accept & \underline{accept} & accept & accept & \cdots & accept & \cdots \\
M_3 & reject & reject & \underline{reject} & reject & & reject & \\
M_4 & accept & accept & reject & \underline{reject} & & accept & \\
\vdots & & \vdots & & & \ddots & & \\
D & reject & reject & accept & accept & & \underline{\ ?\ } & \\
\vdots & & \vdots & & & & & \ddots
\end{array}
$$

A Turing-unrecognizable language

Definition:

A language is co-Turing-recognizable if it is the complement of a Turing-recognizable language.

Theorem:

A language is decidable iff
it is Turing-recognizable and co-Turing-recognizable.

IOW: a language is decidable exactly when both it and its complement are Turing-recognizable.

Proof

- First direction
  - If A is decidable, we can easily see that both A and its complement $\overline{A}$ are Turing-recognizable
    - Any decidable language is Turing-recognizable,

    and the complement of a decidable language also is decidable
- Second direction
  - If both A and its complement $\overline{A}$ are Turing-recognizable, we let M_1 be the recognizer for A and M_2 be the recognizer for $\overline{A}$.
  - The following TM M is decider for A

Proof

$M =$ "On input w:
1. Run both M_1 and M_2 on input w in parallel.
2. If M_1 accepts, *accept*; if M_2 accepts, *reject*."

- Running the two machines in parallel means that M has two tapes, one for simulating M_1 and another for simulating M_2
- Next, we show that M decides A
  - Every string w is either in A or $\overline{A}$
  - Therefore, either M_1 or M_2 must accept w
  - Because M halts whenever M_1 or M_2 accepts, M always halts, and so it is a decider
  - Furthermore, it accepts all strings in A and rejects all strings not in A. So M is decider for A, and thus A is decidable.

A Turing-unrecognizable language: exhibit

Corollary:

The complement of A_{TM} is not Turing-recognizable

Proof:

- We know that A_{TM} is Turing-recognizable.
- If A_{TM} complement also were Turing-recognizable, A_{TM} would be decidable.
- But we know that A_{TM} is not decidable.
- So A_{TM} complement must not be Turing-recognizable.