Undecidability, part II

April 8, 2020
In last lecture, we saw...

\[ A_{TM} = \{<M,w> | M \text{ is a TM and } M \text{ accepts } w \} \]

- \( A_{TM} \) is Turing-Recognizable
  - Universal Turing Machine
- Countable and Uncountable sets
  - Correspondence
- The set of real numbers \( \mathbb{R} \) 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
- ...and talk a bit about Mid Term 2
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 $\Sigma^*$ is countable for any alphabet $\Sigma$
    - With only finitely many strings of each length, we may form a list $\Sigma^*$ 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 $\langle M \rangle$

- 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 numbers $\mathbb{R}$ is uncountable
The set of all languages is uncountable

- Let \( L \) be the set of all languages over alphabet \( \Sigma \)
- 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 \( i \)th bit of that sequence is a 1 if \( s_i \in A \) and is a 0 if \( s_i \not\in A \)
- 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 \( \chi_A \) would be

\[
\Sigma^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \} \\
A = \{ 0, 00, 01, 000, 001, \ldots \} \\
\chi_A = \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\end{array} \ldots
\]
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

\[ H(\langle M, w \rangle) = \begin{cases} 
  \text{accept} & \text{if } M \text{ accepts } w \\
  \text{reject} & \text{if } M \text{ does not accept } w.
\end{cases} \]
$A_{TM}$ 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 $\langle M \rangle$
- Once $D$ has determined this information, it does the opposite (i.e. it rejects if $M$ accepts and accepts if $M$ does not accept)

$$D = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

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:

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

What happens when we run D with its own description \langle D \rangle as input?

In that case, we get:

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

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 $A_{\text{TM}}$
- 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 $<M, w>$ exactly when $M$ accepts $w$.
- $D$ rejects $<M>$ exactly when $M$ accepts $<M>$.
- $D$ rejects $<D>$ exactly when $D$ accepts $<D>$. 
But where 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:

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>
But where the diagonalization in the proof?

Entry $i,j$ is the value of $H$ on input $\langle M_i, <M_j> \rangle$
But where the diagonalization in the proof?

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

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>\ldots</th>
<th>$\langle D \rangle$</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>...</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>...</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>...</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 $A\overline{\phantom{0}}$ 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 $A\overline{\phantom{0}}$ are Turing-recognizable, we let $M_1$ be the recognizer for $A$ and $M_2$ be the recognizer for $A\overline{\phantom{0}}$.
  - The following TM $M$ is decider for $A$
Proof

\[ M = \text{“On input } w:\]
\[ 1. \text{ Run both } M_1 \text{ and } M_2 \text{ on input } w \text{ in parallel.}
2. \text{ If } M_1 \text{ accepts, } \text{accept}; \text{ if } M_2 \text{ accepts, } \text{reject.”} \]

- 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 \( A^- \)
  - 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.
Mid Term 2 Info

- **Topics**
  - **Context free grammars** (sec 2.1 of the book)
    - Lectures: 2/14, 2/19
  - **Pushdown Automata** (sec 2.2 of the book)
    - Lectures: 2/21, 2/24, 3/2
  - **Turing machines** (sec 3.1, 3.2 and 3.3 of the book)
  - **Decidable languages** (sec 4.1 of the book)
    - Lectures: 4/1, 4/3

- **Format**
  - Take home (to be turned within 24 hours)
  - Out on **Friday April 10 at 10am**, due on **Sat April 11 by 2pm**
  - Submitted on OSBLE in PDF (generated or hand-written and scanned)
  - We will have class on Friday 4/10 for clarification/review
Schedule at a glance…

   - Mon 3/23
   - Wed 3/25
   - Fri 3/27: HW6 out ✔

2. Week 3/30 – 4/3 ✔
   - Mon 3/30
   - Wed 4/1
   - Fri 4/3: HW6 in ✔

3. Week 4/6 – 4/10
   - Mon 4/6
   - Wed 4/8
   - Fri 4/10: Mid Term 2 (take-home)

4. Week 4/13 – 4/17
   - Mon 4/13
   - Wed 4/15
   - Fri 4/17: HW7 out

5. Week 4/20 – 4/24
   - Mon 4/20
   - Wed 4/22
   - Fri 4/24: HW 7 in, HW8 out

6. Week 4/27 – 5/1
   - Mon 4/27
   - Wed 4/29
   - Fri 5/1: HW 8 in, Wrap-up

7. Week 5/4 – 5/8
   - Thurs 5/7: Final due (take-home)