Lecture 6
The Acceptance Problem for TMs

\[ A_{TM} = \{ <M,w> \mid M \text{ is a TM} \& w \in L(M) \} \]

Theorem: \( A_{TM} \) is Turing recognizable

\[ \text{Pf: It is recognized by a TM } U \text{ that, on input } <M,w>, \text{ simulates } M \text{ on } w \text{ step by step. } U \text{ accepts iff } M \text{ does.} \]

\( U \) is called a \textit{Universal Turing Machine}
(Ancestor of the stored-program computer)

Note that \( U \) is a recognizer, not a decider.
**A\textsubscript{TM} is Undecidable**

\[ A\textsubscript{TM} = \{ <M,w> | M \text{ is a TM} \& w \in L(M) \} \]

Suppose it’s decidable, say by TM H. Build a new TM D:

“on input <M> (a TM), run H on <M,<M>>; when it halts, halt & do the opposite, i.e. accept if H rejects and vice versa”

D accepts <M> iff H rejects <M,<M>> \hspace{1cm} (by construction)

iff M rejects <M> \hspace{1cm} (H recognizes A\textsubscript{TM})

D accepts <D> iff D rejects <D> \hspace{1cm} (special case)

**Contradiction!**
A specific non-Turing-recognizable language

Let $M_i$ be the TM encoded by $w_i$, i.e., $<M_i>$ = $w_i$. If $w_i$ is an illegal code, $M_i$ = some default machine.

The $i, j$ entry tells whether $M_i$ accepts $w_j$.

Then $L_D$ is not recognized by any TM.

Note: The above TM $D$, if it existed, would recognize exactly the language $L_D$ defined in this diagonalization proof (which we already know is not recognizable).

Let $M_i$ be the TM encoded by $w_i$. If $w_i$ is a legal code, $M_i$ is a Turing machine.

Then $L_D$ is not recognized by any TM.

<table>
<thead>
<tr>
<th></th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
<th>W5</th>
<th>W6</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;M1&gt;</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&lt;M2&gt;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;M3&gt;</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>&lt;M4&gt;</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;M5&gt;</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;M6&gt;</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>L_D</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Decidable $\subsetneq$ Recognizable
Decidable = $\text{Rec} \cap \text{co-Rec}$

$L$ decidable iff both $L$ & $L^c$ are recognizable

Pf:
($\iff$) on any given input, dovetail a recognizer for $L$ with one for $L^c$; one or the other must halt & accept, so you can halt & accept/reject appropriately.

($\Rightarrow$): from last lecture, decidable languages are closed under complement (flip acc/rej)
Reduction

“A is reducible to B” means I could solve A if I had a subroutine for B

Ex:
Finding the max element in a list is reducible to sorting
pf: sort the list in increasing order, take the last element
(A big hammer for a small problem, but never mind...)
The Halting Problem

\[ \text{HALT}_\text{TM} = \{ <M,W> \mid \text{TM M halts on input w} \} \]

Theorem: The halting problem is undecidable

Proof:

A = A_\text{TM}, B = \text{HALT}_\text{TM} Suppose I can reduce A to B. We already know A is undecidable, so must be that B is, too.

Suppose TM R decides \text{HALT}_\text{TM}. Consider S:

\begin{align*}
on \text{input } <M,w>, & \text{ run } R \text{ on it. If it rejects, halt & reject; if it accepts, run } M \text{ on } w; \text{ accept/reject as it does.} \\
\text{Then } S \text{ decides } A_\text{TM}, \text{ which is impossible. } R \text{ can’t exist.}
\end{align*}