Computability and Reductions

29 November 2022
We’ll do CEQs at the end of class on Thursday.
Designing Collaborative and Playful Technology for Accessible Learning

Vinitha Gadiraju

Abstract

Children with disabilities must develop a multifaceted skill set to gain access to education, future employment, hobbies, and independent living. This dynamic learning process necessitates support in multiple forms and contexts, from peers and educators in the classroom to family members in the home. However, current assistive technologies and educational tools are isolating and focus solely on academic skills without considering other areas of development. My research leverages collaboration and playful learning to explore how technology can support complex skill learning for blind or visually impaired (BVI) children and their parents and teachers.

In this talk, I will show a holistic approach, including qualitative methodology and prototyping, to supporting accessible education. First, I will present BrailleBlocks, a set of interactive blocks that connect with educational games to facilitate playful Braille learning between BVI children and sighted parents. Next, I will discuss an ethnographic and game co-design study I conducted at a specialized school for BVI children. This observational work characterized necessary functional development, such as Independent Living Skills, that educators weave into Braille and other academic subjects. Finally, I will present my ongoing work that builds on these projects by investigating barriers to and strategies for at-home Independent Living Skill learning. I will also contemplate future research directions relating to intersectional learning challenges for people with disabilities and supporting long-term assistive technology use.

About the speaker

Vinitha Gadiraju is a Ph.D. Candidate in Computer Science at the University of Colorado Boulder, advised by Dr. Shaun Kane. Her research develops collaborative educational tools and frameworks for people with disabilities and their support networks through qualitative methodology, rapid prototyping, and community engagement. Vinitha’s work on accessibility and technology-mediated learning has been published in top HCI research conferences, including ACM CHI and ASSETS. She is an NSF Fellow, Chateaubriand Fellow, and ACM Student Research Competition Winner. She has also received multiple research and service awards from University of Colorado’s Department of Computer Science. Vinitha received a B.S. in Computer Science from the University of Oregon in 2018, where she served as President of Women in Computer Science.
Where are we?
The class of Turing-decidable languages (R) represents problems that can truly be solved by a computer.

The class of Turing-recognizable languages (RE) represents problems where “yes” answers can be found but “no” answers can’t be ruled out.
There are languages that Turing machines cannot decide
We saw how to use self-reference to find languages that are *undecidable* – languages that are not in \( R \).
def will_accept(program: str, input: str) -> bool:
    ...
    some implementation...
    ...

def main(my_input: str) -> bool:
    me = my_source()
    if will_accept(me, my_input):
        return False
    else:
        return True

The self-defeating object
Using that object against itself
THEOREM $A_{TM} \notin R.$

PROOF By contradiction; assume that $A_{TM} \in R$. Then there is some decider $D$ for $A_{TM}$, which we can represent in software as a function `will_accept` that takes as input the source code of a program and an input, and returns true if the program accepts the input and false otherwise.

Given this, we could then construct this program $P$:

```python
def main(my_input: str) -> bool:
    me = my_source()
    if will_accept(me, my_input):
        return False
    else:
        return True
```

Choose any string $w$ and trace through the execution of program $P$ on input $w$: If `will_accept(me, my_input)` returns true, this means that $P$ must accept its input $w$, but instead it rejects it. Otherwise, if `will_accept(me, my_input)` returns false, this means that $P$ must not accept its input $w$, but instead it accepts it.

In both cases, we reach a contradiction, so our assumption must have been wrong. Therefore, $A_{TM} \notin R.$ □
THEOREM $A_{TM} \notin R$.

PROOF By contradiction; assume that $A_{TM} \in R$. Then there is some decider $D$ for $A_{TM}$. If this machine is given a TM–string pair, it will determine whether the TM accepts the string and report back the answer.

Given this, we could then construct the following TM:

$M = \text{“On input } w: \text{"}

1. Have $M$ obtain its own description $\langle M \rangle$.

2. Run $D$ on $\langle M, w \rangle$ and see what it says.

3. If $D$ says that $M$ will accept $w$, reject.

4. If $D$ says that $M$ will not accept $w$, accept.”

Choose any string $w$ and trace through the execution of the machine, focusing on the answer given back by the machine $D$. If $D$ says that $M$ will accept $w$, notice that $M$ then proceeds to reject $w$, contradicting what $D$ says. Otherwise, if $D$ says that $M$ will not accept $w$, notice that $M$ then proceeds to accept $w$, contradicting what $D$ says.

In both cases, we reach a contradiction, so our assumption must have been wrong. Therefore, $A_{TM} \notin R$. ■
There are languages that Turing machines cannot even recognize
Languages, TMs, and TM encodings

*Recall:* The language of a TM $M$ is the set

$$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

Some of the strings in this set might – by pure coincidence – be encodings of TMs.

*Idea:* Let’s think about different Turing machines and how they behave when they’re given a Turing machine as input.
\[ M_0 \]
\[ M_1 \]
\[ M_2 \]
\[ M_3 \]
\[ M_4 \]
\[ M_5 \]
\[ \ldots \]
All Turing machines, listed in some order
All descriptions of Turing machines, listed in the same order
<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</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>$\langle M_5 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>...</td>
</tr>
<tr>
<td>---</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>-----</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>$M_0$</td>
<td>$M_1$</td>
<td>$M_2$</td>
<td>$M_3$</td>
<td>$M_4$</td>
<td>$M_5$</td>
<td>...</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Flip all “accept”s to “no”s and vice versa.
<table>
<thead>
<tr>
<th>$\langle M_0 \rangle$</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>$\langle M_5 \rangle$</th>
<th>$\ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>$\ldots$</td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

What TM has this behavior?
<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</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>$\langle M_5 \rangle$</th>
<th>$\ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>( M_0 )</td>
<td>( M_1 )</td>
<td>( M_2 )</td>
<td>( M_3 )</td>
<td>( M_4 )</td>
<td>( M_5 )</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><strong>Acc</strong></td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Acc</strong></td>
<td><strong>Acc</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Acc</strong></td>
<td><strong>Acc</strong></td>
<td><strong>Acc</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td><strong>No</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td><strong>No</strong></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><strong>No</strong></td>
<td>No</td>
<td>No</td>
<td><strong>Acc</strong></td>
<td>No</td>
<td><strong>Acc</strong></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
\[
\begin{array}{ccccccc}
\langle M_0 \rangle & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \langle M_5 \rangle & \ldots \\
\hline
M_0 & \text{Acc} & \text{No} & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \ldots \\
M_1 & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \ldots \\
M_2 & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \ldots \\
M_3 & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \text{Acc} & \text{Acc} & \ldots \\
M_4 & \text{Acc} & \text{No} & \text{Acc} & \text{No} & \text{Acc} & \text{No} & \ldots \\
M_5 & \text{No} & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \text{No} & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\text{No} & \text{No} & \text{No} & \text{Acc} & \text{No} & \text{Acc} & \ldots \\
\end{array}
\]
<table>
<thead>
<tr>
<th></th>
<th>(M_0)</th>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(M_3)</th>
<th>(M_4)</th>
<th>(M_5)</th>
<th>(\cdots)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_0)</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(M_1)</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(M_2)</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(M_3)</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(M_4)</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(M_5)</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>(\cdots)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>...</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>-----</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</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>
<tr>
<td>...</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>(M_0)</td>
<td>(M_1)</td>
<td>(M_2)</td>
<td>(M_3)</td>
<td>(M_4)</td>
<td>(M_5)</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>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>
\[
\begin{array}{cccccccc}
\langle M_0 \rangle & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \langle M_5 \rangle & \ldots \\
M_0 & \text{Acc} & \text{No} & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \ldots \\
M_1 & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \ldots \\
M_2 & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \ldots \\
M_3 & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \text{Acc} & \text{Acc} & \ldots \\
M_4 & \text{Acc} & \text{No} & \text{Acc} & \text{No} & \text{Acc} & \text{No} & \ldots \\
M_5 & \text{No} & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \text{No} & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\text{No} & \text{No} & \text{No} & \text{Acc} & \text{No} & \text{Acc} & \text{Acc} & \ldots \\
\end{array}
\]
<table>
<thead>
<tr>
<th></th>
<th>(M_0)</th>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(M_3)</th>
<th>(M_4)</th>
<th>(M_5)</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_0)</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>(M_1)</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>(M_2)</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>(M_3)</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>(M_4)</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>(M_5)</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</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>
<tr>
<td>...</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>(M_0)</td>
<td>(M_1)</td>
<td>(M_2)</td>
<td>(M_3)</td>
<td>(M_4)</td>
<td>(M_5)</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>(M_0)</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>(M_1)</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>(M_2)</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>(M_3)</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td></td>
</tr>
<tr>
<td>(M_4)</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>(M_5)</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$M_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>$\ldots$</td>
<td></td>
</tr>
</tbody>
</table>

No TM has this behavior!
<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</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>$\langle M_5 \rangle$</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>…</td>
</tr>
<tr>
<td>---</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>---</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

“The language of all TMs that do not accept their own descriptions”
\[
\begin{array}{cccccc}
\langle M_0 \rangle & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \langle M_5 \rangle & \ldots \\
M_0 & \text{Acc} & \text{No} & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \ldots \\
M_1 & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \ldots \\
M_2 & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \text{Acc} & \ldots \\
M_3 & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \text{Acc} & \text{Acc} & \ldots \\
M_4 & \text{Acc} & \text{No} & \text{Acc} & \text{No} & \text{Acc} & \text{No} & \ldots \\
M_5 & \text{No} & \text{No} & \text{Acc} & \text{Acc} & \text{No} & \text{No} & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\text{No} & \text{No} & \text{No} & \text{Acc} & \text{No} & \text{Acc} & \ldots \\
\end{array}
\]

\{ \langle M \rangle \mid M \text{ is a TM that does not accept } \langle M \rangle \}
What just happened?

Some of the strings in the language of a TM may be descriptions of TMs.

We can draw a matrix showing which descriptions of TMs are accepted by which TMs.

The complemented diagonal cannot be accepted by any TM, because each TM disagrees with itself.

Therefore, there is some language not recognized by any TM.
A deviously tricky problem

The *diagonalization language* $L_D$ is defined as

$$L_D = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ does not accept } \langle M \rangle \}$$

We constructed this language to be different from the language of every TM.

Therefore, $L_D \notin \text{RE}$!
We’ve seen that

$A_{TM}$ and $HALT_{TM}$ are undecidable

$L_D$ is unrecognizable

Can we use these results to show that other problems are also unrecognizable or undecidable?
Reductions
I wonder if I can lift that car…
Nope! It turns out that cars are heavy!
I wonder if I can lift this fully loaded truck!
Nope! Because then I could lift a car!
A *reduction* works by turning an instance of the problem $E$ (easy) into an instance of a problem $H$ (harder).

In this example, we reduce the problem of lifting the car by putting it into a truck and lifting the truck.

Suppose we cannot solve $E$. If we can reduce $E$ to $H$, we cannot solve $H$ either.

We can’t lift the car, so we can’t lift the truck.
$A$ reduces to $B$ and $B \in \mathbb{R} \quad \Rightarrow \quad A \in \mathbb{R}$

$A$ reduces to $B$ and $B \in \mathbb{RE} \quad \Rightarrow \quad A \in \mathbb{RE}$

$A$ reduces to $B$ and $A \not\in \mathbb{R} \quad \Rightarrow \quad B \not\in \mathbb{R}$

$A$ reduces to $B$ and $A \not\in \mathbb{RE} \quad \Rightarrow \quad B \not\in \mathbb{RE}$
Proof by reduction: \textit{DECIDER} is undecidable
Can we tell whether a given TM is a decider or just a recognizer?

Let $DECIDER = \{\langle M \rangle \mid M$ is a decider\}.

That is, $M$ halts on all inputs.
Can we tell whether a given TM is a decider or just a recognizer?

Let $DECIDER = \{ \langle M \rangle \mid M \text{ is a decider}\}$.

That is, $M$ halts on all inputs.

**Idea:** Reduce $HALT_{TM}$ to $DECIDER$.

Show how a decider for $DECIDER$ would also decide $HALT_{TM}$.

Conclude that no such decider can exist.
Assume, for the sake of contradiction, that DECIDER is decidable.
Decider for DECIDER
We’re going to try to show how a decider for DECIDER decides $\text{HALT}_\text{TM}$, so our input must be a TM and a string.
Construct $M'$ from $\langle M, w \rangle$
Construct $M'$ from $\langle M, w \rangle$
Construct $M'$ from $\langle M, w \rangle$

Decider for $\text{DECIDER}$

$x$ (ignored)
Decider for $\langle M, w \rangle$

Construct $M'$ from $\langle M, w \rangle$

Decider for DECIDER

Simulate $M$ on $w$

(ignored)
Construct $M'$ from $\langle M, w \rangle$.

Decider for $\text{DECIDER}$.

Simulate $M$ on $w$.

$x$ (ignored)
$M' = \text{“On input } x:$$

\text{Ignore } x.$

\text{Run } M \text{ on } w.$

\text{If } M \text{ accepts } w, \textit{accept}.\textit{ }

\text{If } M \text{ rejects } w, \textit{reject.”}
Construct $M'$ from $\langle M, w \rangle$

Decider for DECIDER

Simulate $M$ on $w$ if $M$ halts on $w$?
Decider for $\langle M, w \rangle$

Construct $M'$ from $\langle M, w \rangle$

Decider for $\text{DECIDER}$

What does $M'$ do if $M$ halts on $w$?

$M'$ always halts!
Construct $M'$ from $\langle M, w \rangle$

Decider for DECIDER

What does $M'$ do if $M$ loops on $w$?

(Model $x$ ignored)

Simulate $M$ on $w$
Construct $M'$ from $\langle M, w \rangle$ for Decider for $DECIDER$.

Simulate $M'$ on $w$.

If $M$ loops on $w$, $M'$ never halts! What does $M'$ do if $M$ loops on $w$?
Construct $M'$ from $\langle M, w \rangle$.

Simulate $M'$ on $w$.

Decider for DECIDER.
\[
\langle M, w \rangle \xrightarrow{\text{Construct } M'} \langle M' \rangle \xrightarrow{\text{Decider for } DECIDER}
\]

\[
M' \xrightarrow{\text{(ignored)}} \text{Simulate } M \text{ on } w
\]
Decider for $\langle M, w \rangle$

Construct $M'$ from $\langle M, w \rangle$

Simulate $M$ on $w$

$H$

What does $H$ do if $M$ halts on $w$?
Decider for DECIDER

Construct $M'$ from $\langle M, w \rangle$

Decider for DECIDER

$M'$

$\langle M, w \rangle$

(ignored)

Simulate $M$ on $w$

What does $H$ do if $M$ halts on $w$?

This means that $M'$ always halts!
Decider for DECIDER

Construct $M'$ from $\langle M, w \rangle$

Simulate $M$ on $w$

Accept!

What does $H$ do if $M$ halts on $w$?

This means that $M'$ always halts!
Construct $M'$ from $\langle M, w \rangle$ and simulate $M'$ on $w$. The result is then fed into the Decider for $DECIDER$. The process is depicted in the diagram with the input $\langle M, w \rangle$ leading to $H$, then $M'$ being constructed and simulated on $w$, and finally the result being fed into the Decider.
Decider for\( \langle M, w \rangle \)

Construct \( M' \) from \( \langle M, w \rangle \)

Decider for \( \text{DECIDER} \)

\[ H \]

\[ \langle M' \rangle \]

What does \( H \) do if \( M \) loops on \( w \)?

\[ M' \]

(ignored)

Simulate \( M \) on \( w \)

\[ x \]
Decider for $\langle M, w \rangle$

Construct $M'$ from $\langle M, w \rangle$

Decider for DECIDER

What does $H$ do if $M$ loops on $w$?

This means that $M'$ never halts!
Construct $M'$ from $\langle M, w \rangle$

Simulate $M$ on $w$

What does $H$ do if $M$ loops on $w$?

This means that $M'$ never halts!
Construct $M'$ from $\langle M, w \rangle$.

Simulate $M'$ on $w$.

Decider for $DECIDER$.
Simulate $M$ on $w$.
Simulate \( M \) on \( w \)

What does \( H \) do if \( M \) halts on \( w \)?
\[(M, w)\] x (ignored)

Simulate \(M\) on \(w\)

Accept!

What does \(H\) do if \(M\) halts on \(w\)?
Simulate $M'$ on $w$
Simulate $M$ on $w$.

What does $H$ do if $M$ loops on $w$?
Simulate $M$ on $w$.

What does $H$ do if $M$ loops on $w$?

$M' \quad (\text{ignored})$

Simulate $M$ on $w$.

Reject!
Decider for DECIDER

Construct $M'$ from $\langle M, w \rangle$

Simulate $M'$ on $w$

(ignored)
Construct $M'$ from $\langle M, w \rangle$ and simulate $M'$ on $w$. $H$ is a decider for \textsc{Halt}_{TM}!
Decider for $\langle M, w \rangle$

Construct $M'$ from $\langle M, w \rangle$

Simulate $M'$ on $w$

$H$ is a decider for $\text{HALT}_{TM}$!
What just happened?

Suppose, for the sake of contradiction, that DECIDER is decidable.
What just happened?

Suppose, for the sake of contradiction, that \textit{DECIDER} is decidable.

Build a TM $H$ that takes $\langle M, w \rangle$ and constructs a TM $M'$ that is a decider iff $M$ accepts $w$.

\begin{align*}
\text{We build a TM that has a property of the new problem (here, DECIDER) based on whether some other TM has a property of the old problem (here, HALT_{TM}).} \\
\text{Deciding whether this TM has the new property thus decides whether some other TM has the old property.}
\end{align*}
What just happened?

Suppose, for the sake of contradiction, that \textsc{Decider} is decidable.

Build a TM $H$ that takes $\langle M, w \rangle$ and constructs a TM $M'$ that is a decider iff $M$ accepts $w$.

Using the decider for \textsc{Decider}, $H$ checks whether $M'$ is a decider:

- If $M'$ is a decider, then $M$ halts on $w$.
- If $M'$ is not a decider, then $M$ does not halt on $w$. 
What just happened?

Suppose, for the sake of contradiction, that \textit{DECIDER} is decidable.

Build a TM $H$ that takes $\langle M, w \rangle$ and constructs a TM $M'$ that is a decider iff $M$ accepts $w$.

Using the decider for \textit{DECIDER}, $H$ checks whether $M'$ is a decider:

- If $M'$ is a decider, then $M$ halts on $w$.
- If $M'$ is not a decider, then $M$ does not halt on $w$.

Conclude that $\text{HALT}_{TM}$ is decidable.
What just happened?

Suppose, for the sake of contradiction, that DECIDER is decidable.

Build a TM \( H \) that takes \( \langle M, w \rangle \) and constructs a TM \( M' \) that is a decider iff \( M \) accepts \( w \).

Using the decider for DECIDER, \( H \) checks whether \( M' \) is a decider:

- If \( M' \) is a decider, then \( M \) halts on \( w \).
- If \( M' \) is not a decider, then \( M \) does not halt on \( w \).

Conclude that \( \text{HALT}_\text{TM} \) is decidable.

We know this is wrong, so our initial assumption must be wrong – DECIDER isn’t decidable!
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that DECIDER is decidable. Let $T$ be a decider for DECIDER. Then consider the following TM: $D = \text{On input } \langle M, w \rangle:$

- Construct the TM $M' = \text{On input } x:$
  - Ignore $x$.
  - Run $M$ on $w$.
  - If $M$ accepts $w$, accept.
  - If $M$ rejects $w$, reject.
- Run $T$ on $\langle M' \rangle$.
  - If $T$ accepts, accept.
  - If $T$ rejects, reject.

We claim that $D$ decides $\text{HALT}_{TM}$. To see this, we show that $D$ is a decider and that $L(D) = \text{HALT}_{TM}$. To see that $D$ is a decider, note that after we construct $M'$, we run $T$ on $\langle M' \rangle$. Since $T$ is a decider, it always halts, so $D$ always halts.

To see that $L(D) = \text{HALT}_{TM}$, note that $D$ accepts $\langle M, w \rangle$ if $T$ accepts $\langle M' \rangle$. Because $T$ is a decider for DECIDER, $T$ accepts $\langle M' \rangle$ if $M'$ halts on all inputs. By construction, $M'$ halts on any input if $M$ halts on $w$. Finally, $M$ halts on $w$ if $\langle M, w \rangle \in \text{HALT}_{TM}$. This means that $D$ accepts $\langle M, w \rangle$ if $\langle M, w \rangle \in \text{HALT}_{TM}$, so $L(D) = \text{HALT}_{TM}$.

We have reached a contradiction because $D$ decides $\text{HALT}_{TM}$, which we know is undecidable. Thus our assumption was wrong and DECIDER is undecidable. ■
THEOREM  \textit{DECIDER} is undecidable.

PROOF  By contradiction; assume that \textit{DECIDER} is decidable. Let $T$ be a decider for $\textit{DECIDER}$.

We claim that $D$ decides $\text{HALT}_{TM}$. To see this, we show that $D$ is a decider and that $L(D) = \text{HALT}_{TM}$. To see that $D$ is a decider, note that after we construct $M'$, we run $T$ on $\langle M' \rangle$. Since $T$ is a decider, it always halts, so $D$ always halts.

To see that $L(D) = \text{HALT}_{TM}$, note that $D$ accepts $\langle M, w \rangle$ if $T$ accepts $\langle M' \rangle$. Because $T$ is a decider for $\text{DECIDER}$, $T$ accepts $\langle M' \rangle$ if $M'$ halts on all inputs. By construction, $M'$ halts on any input if $M$ halts on $w$. Finally, $M$ halts on $w$ if $\langle M, w \rangle \in \text{HALT}_{TM}$. This means that $D$ accepts $\langle M, w \rangle$ if $\langle M, w \rangle \in \text{HALT}_{TM}$, so $L(D) = \text{HALT}_{TM}$.

We have reached a contradiction because $D$ decides $\text{HALT}_{TM}$, which we know is undecidable. Thus our assumption was wrong and $\text{DECIDER}$ is undecidable. ■
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that
DECIDER is decidable Let $T$ be a decider
for DECIDER. Then consider the following
TM:

$H = \"On input \langle M, w \rangle:\$
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that
DECIDER is decidable Let T be a decider for DECIDER. Then consider the following TM:

\[ H = \text{"On input } \langle M, w \rangle:\]
\[ \text{Construct the TM } M' = \text{"On input } x:\]
\[ \text{Ignore } x.\]
\[ \text{Run } M \text{ on } w.\]
\[ \text{If } M \text{ accepts } w, \text{ accept.}\]
\[ \text{If } M \text{ rejects } w, \text{ reject."} \]

We claim that D decides HALT_TM. To see this, we show that D is a decider and that \( L(D) = \text{HALT}_{TM} \). To see that D is a decider, note that after we construct \( M' \), we run T on \( \langle M' \rangle \). Since T is a decider, it always halts, so D always halts.

To see that \( L(D) = \text{HALT}_{TM} \), note that D accepts \( \langle M, w \rangle \) iff T accepts \( \langle M' \rangle \). Because T is a decider for DECIDER, T accepts \( \langle M' \rangle \) iff \( M' \) halts on all inputs. By construction, \( M' \) halts on any input iff \( M \) halts on \( w \). Finally, \( M \) halts on \( w \) iff \( \langle M, w \rangle \in \text{HALT}_{TM} \). This means that D accepts \( \langle M, w \rangle \) iff \( \langle M, w \rangle \in \text{HALT}_{TM} \), so \( L(D) = \text{HALT}_{TM} \).

We have reached a contradiction because D decides HALT_TM, which we know is undecidable. Thus our assumption was wrong and DECIDER is undecidable. ■

Most reduction proofs work by building a new TM out of an existing TM and a string. The new TM then has some property iff the old TM–string pair has some property.
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that
DECIDER is decidable Let T be a decider for DECIDER. Then consider the following
TM:

\[ H = "\text{On input } \langle M, w \rangle:\]

- Construct the TM \( M' \) = "On input x:
  - Ignore x.
  - Run \( M \) on \( w \).
  - If \( M \) accepts \( w \), accept.
  - If \( M \) rejects \( w \), reject."
- Run \( T \) on \( \langle M' \rangle \).
- If \( T \) accepts, accept.
- If \( T \) rejects, reject."

The behavior of \( H \) depends on what \( T \) does on \( \langle M' \rangle \).

The behavior of \( T \) on \( \langle M' \rangle \) depends on what \( M \) does on \( w \).

So the behavior of TM \( H \) depends on what \( M \) does on \( w \).
THEOREM  \( \text{DECIDER} \) is undecidable.

PROOF  By contradiction; assume that \( \text{DECIDER} \) is decidable. Let \( T \) be a decider for \( \text{DECIDER} \). Then consider the following TM:

\[
H = "\text{On input } \langle M, w \rangle:\n\text{Construct the TM } M' = "\text{On input } x:\n\text{Ignore } x.\n\text{Run } M \text{ on } w.\n\text{If } M \text{ accepts } w, \text{ accept.}\n\text{If } M \text{ rejects } w, \text{ reject.}"
\text{Run } T \text{ on } \langle M' \rangle.\n\text{If } T \text{ accepts, accept.}\n\text{If } T \text{ rejects, reject.}"
\]

We claim that \( H \) decides \( \text{HALT}_{TM} \). To see this, we show that \( H \) is a decider and that \( L(H) = \text{HALT}_{TM} \).
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that DECIDER is decidable. Let T be a decider for DECIDER. Then consider the following TM:

\[ H = \text{"On input } \langle M, w \rangle: \]
\[ \text{Construct the TM } M' = \text{"On input } x: \]
\[ \text{Ignore } x. \]
\[ \text{Run } M \text{ on } w. \]
\[ \text{If } M \text{ accepts } w, \text{ accept.} \]
\[ \text{If } M \text{ rejects } w, \text{ reject."} \]
\[ \text{Run } T \text{ on } \langle M' \rangle. \]
\[ \text{If } T \text{ accepts, accept.} \]
\[ \text{If } T \text{ rejects, reject."} \]

We claim that H decides \( \text{HALT}_{\text{TM}} \). To see this, we show that H is a decider and that \( L(H) = \text{HALT}_{\text{TM}} \). To see that H is a decider, note that \textbf{after we construct } M', \textbf{we run } T \textbf{ on } \langle M' \rangle.

\[ \text{To be completely formal in our proof, we should show that this construction can be done in finite time so that } H \text{ doesn't loop infinitely trying to construct } M'. \]

\[ \text{However, conventionally this is just assumed to be true and you don't need to justify it.} \]
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that 
DECIDER  is decidable Let T be a decider 
for DECIDER. Then consider the following 
TM:

\[
H = \text{"On input } \langle M, w \rangle:\n\]
\[
\text{Construct the TM } M' = \text{"On input } x:\n\]
\[
\text{Ignore } x.
\]
\[
\text{Run } M \text{ on } w.
\]
\[
\text{If } M \text{ accepts } w, \text{ accept.}
\]
\[
\text{If } M \text{ rejects } w, \text{ reject."
}\]
\[
\text{Run } T \text{ on } \langle M' \rangle.
\]
\[
\text{If } T \text{ accepts, accept.}
\]
\[
\text{If } T \text{ rejects, reject."
}\]

We claim that H decides \( \text{HALT}_{TM} \). To see 
this, we show that H is a decider and that 
L(H) = \( \text{HALT}_{TM} \). To see that H is a decider, 
note that after we construct \( M' \), we run 
T on \( \langle M' \rangle \). Since T is a decider, it always 
halts, so H always halts.
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that DECIDER is decidable Let T be a decider for DECIDER. Then consider the following TM:

\[ H = \text{"On input } \langle M, w \rangle: \]
\[ \text{Construct the TM } M' = \text{"On input } x: \]
\[ \text{Ignore } x. \]
\[ \text{Run } M \text{ on } w. \]
\[ \text{If } M \text{ accepts } w, \text{ accept.} \]
\[ \text{If } M \text{ rejects } w, \text{ reject."} \]
\[ \text{Run } T \text{ on } \langle M' \rangle. \]
\[ \text{If } T \text{ accepts, accept.} \]
\[ \text{If } T \text{ rejects, reject."} \]

We claim that H decides \( \text{HALT}_{TM} \). To see this, we show that H is a decider and that \( L(H) = \text{HALT}_{TM} \). To see that H is a decider, note that after we construct \( M' \), we run \( T \) on \( \langle M' \rangle \). Since \( T \) is a decider, it always halts, so H always halts.

To see that \( L(H) = \text{HALT}_{TM} \), note that \( H \) accepts \( \langle M, w \rangle \) iff \( T \) accepts \( \langle M' \rangle \). Because \( T \) is a decider for DECIDER, \( T \) accepts \( \langle M' \rangle \) iff \( M' \) halts on all inputs. By construction, \( M' \) halts on any input iff \( M \) halts on \( w \). Finally, \( M \) halts on \( w \) iff \( \langle M, w \rangle \) \( \in \text{HALT}_{TM} \). This means that \( H \) accepts \( \langle M, w \rangle \) iff \( \langle M, w \rangle \) \( \in \text{HALT}_{TM} \), so \( L(H) = \text{HALT}_{TM} \).
THEOREM  DECIDER is undecidable.

PROOF  By contradiction; assume that
DECIDER is decidable Let T be a decider
for DECIDER. Then consider the following
TM:

\[ H = \text{"On input } \langle M, w \rangle:\]
\[ \text{Construct the TM } M' = \text{"On input } x:\]
\[ \text{Ignore } x.\]
\[ \text{Run } M \text{ on } w.\]
\[ \text{If } M \text{ accepts } w, \text{ accept.}\]
\[ \text{If } M \text{ rejects } w, \text{ reject."} \]
\[ \text{Run } T \text{ on } \langle M' \rangle.\]
\[ \text{If } T \text{ accepts, accept.}\]
\[ \text{If } T \text{ rejects, reject."} \]

We claim that \( H \) decides \( \text{HALT}_{TM} \). To see
this, we show that \( H \) is a decider and that
\( L(H) = \text{HALT}_{TM} \). To see that \( H \) is a decider,
note that after we construct \( M' \), we run
\( T \) on \( \langle M' \rangle \). Since \( T \) is a decider, it always
halts, so \( H \) always halts.

To see that \( L(H) = \text{HALT}_{TM} \), note that \( H \)
accepts \( \langle M, w \rangle \) iff \( T \) accepts \( \langle M' \rangle \). Because
\( T \) is a decider for \( \text{DECIDER} \), \( T \) accepts
\( \langle M' \rangle \) iff \( M' \) halts on all inputs. By
construction, \( M' \) halts on any input iff \( M \)
halts on \( w \). Finally, \( M \) halts on \( w \) iff \( \langle M, w \rangle \in \text{HALT}_{TM} \). This means that \( H \) accepts \( \langle M, w \rangle \) iff \( \langle M, w \rangle \in \text{HALT}_{TM} \), so \( L(H) = \text{HALT}_{TM} \).

We have reached a contradiction because
\( H \) decides \( \text{HALT}_{TM} \), which we know is
undecidable. Thus our assumption was
wrong and \( \text{DECIDER} \) is undecidable. ■
Proof by reduction: $\text{REGULAR}_\text{TM}$ is undecidable
Can we detect whether a TM recognizes a regular language?

Let $\text{REGULAR}_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular} \}$. 
Can we detect whether a TM recognizes a regular language?

Let $\text{REGULAR}_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular} \}$. 

This is undecidable, even though it doesn’t have a simple mapping to $A_{\text{TM}}$ or $\text{HALT}_{\text{TM}}$. 
Proof idea: Suppose $\text{REGULAR}_{\text{TM}}$ is decidable by some machine $D$. 

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M')$ is regular.
- If $M$ loops on $w$, then $L(M')$ is not regular.

Have $D$ decide whether or not $M'$ is regular.

- If $M'$ is regular, $M$ halts on $w$.
- If $M'$ is not regular, $M$ loops on $w$.

We can use $D$ to decide $\text{HALT}_{\text{TM}}$, which is a contradiction.
Proof idea: Suppose $\text{REGULAR}_{\text{TM}}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M')$ is regular.
- If $M$ loops on $w$, then $L(M')$ is not regular.

We can use $D$ to decide $\text{HALT}_{\text{TM}}$, which is a contradiction.
Proof idea: Suppose $\text{REGULAR}_{TM}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M')$ is regular.
- If $M$ loops on $w$, then $L(M')$ is not regular.

How do we build a machine with these properties?
Proof idea: Suppose \( \text{REGULAR}_{\text{TM}} \) is decidable by some machine \( D \).

Given a TM \( M \) and a string \( w \), construct a TM \( M' \) with these properties:

- If \( M \) halts on \( w \), then \( L(M') = \Sigma^* \).
- If \( M \) loops on \( w \), then \( L(M') \) is not regular.

How do we build a machine with these properties?
Proof idea: Suppose $\text{REGULAR}_{TM}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

If $M$ halts on $w$, then $L(M') = \Sigma^*$.  
If $M$ loops on $w$, then $L(M') = \{0^n1^n \mid n \in N_0\}$.

Have $D$ decide whether or not $M'$ is regular.  
If $M'$ is regular, $M$ halts on $w$.  
If $M'$ is not regular, $M$ loops on $w$.

We can use $D$ to decide $\text{HALT}_{TM}$, which is a contradiction.

How do we build a machine with these properties?
Proof idea: Suppose $\text{REGULAR}_{\text{TM}}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M') = \Sigma^*$.
- If $M$ loops on $w$, then $L(M') = \{0^n1^n | n \in \mathbb{N}_0\}$.

Have $D$ decide whether or not $M'$ is regular.

If $M'$ is regular, $M$ halts on $w$.

If $M'$ is not regular, $M$ loops on $w$.

We can use $D$ to decide $\text{HALT}_{\text{TM}}$, which is a contradiction.
Proof idea: Suppose $\text{REGULAR}_\text{TM}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M') = \Sigma^*$.
- If $M$ loops on $w$, then $L(M') = \{0^n1^n \mid n \in \mathbb{N}_0\}$.

Have $D$ decide whether or not $M'$ is regular.

If $M'$ is regular, $M$ halts on $w$.

If $M'$ is not regular, $M$ loops on $w$.

We can use $D$ to decide $\text{HALT}_\text{TM}$, which is a contradiction.
**Proof idea:** Suppose \( \text{REGULAR}_{\text{TM}} \) is decidable by some machine \( D \).

Given a TM \( M \) and a string \( w \), construct a TM \( M' \) with these properties:

- If \( M \) halts on \( w \), then \( L(M') = \Sigma^* \).
- If \( M \) loops on \( w \), then \( L(M') = \{0^n1^n | n \in \mathbb{N}_0\} \).

Have \( D \) decide whether or not \( M' \) is regular.

- If \( L(M') \) is regular, \( M \) halts on \( w \).
- If \( L(M') \) is not regular, \( M \) loops on \( w \).
**Proof idea:** Suppose $\text{REGULAR}_{\text{TM}}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M') = \Sigma^*$.
- If $M$ loops on $w$, then $L(M') = \{0^n1^n | n \in \mathbb{N}_0\}$.

Have $D$ decide whether or not $M'$ is regular.

- If $L(M')$ is regular, $M$ halts on $w$.
- If $L(M')$ is not regular, $M$ loops on $w$.

We can use $D$ to decide $\text{HALT}_{\text{TM}}$, which is a contradiction.
Proof idea: Suppose $\text{REGULAR}_{\text{TM}}$ is decidable by some machine $D$.

Given a TM $M$ and a string $w$, construct a TM $M'$ with these properties:

- If $M$ halts on $w$, then $L(M') = \Sigma^*$.
- If $M$ loops on $w$, then $L(M') = \{0^n1^n \mid n \in \mathbb{N}_0\}$.

Have $D$ decide whether or not $M'$ is regular.

- If $L(M')$ is regular, $M$ halts on $w$.
- If $L(M')$ is not regular, $M$ loops on $w$.

We can use $D$ to decide $\text{HALT}_{\text{TM}}$, which is a contradiction.
The mysterious machine
The mysterious machine

\[ x = \theta^n 1^n ? \]
The mysterious machine

$x = \emptyset^n 1^n$?
The mysterious machine

\[ x = 0^n1^n? \]

Simulate \( M \) on \( w \)
The mysterious machine

x = $\theta^n1^n$?

Simulate $M$ on $w$
The mysterious machine

\[ x = \theta^n 1^n \]
The mysterious machine

\[ x = 0^n1^n? \]

\[ M' = \text{"On input } x: \]
\[ \text{If } x = 0^n1^n, \text{ accept.} \]
\[ \text{Otherwise, run } M \text{ on } w. \]
\[ \text{If } M \text{ halts, accept."} \]
The mysterious machine

If $M$ halts on $w$, $M'$ accepts all strings – its language is $\Sigma^*$.

$M' = "On input x:"

If $x = \theta^n1^n$, accept.

Otherwise, run $M$ on $w$.

If $M$ halts, accept."
The mysterious machine

$M' = \text{"On input } x:"

If $x = \theta^n1^n$, accept.
Otherwise, run $M$ on $w$.
If $M$ halts, accept.

If $M$ halts on $w$, $M'$ accepts all strings – its language is $\Sigma^*$.

If $M$ loops on $w$, $M'$ only accepts strings of the form $0^n1^n$. 

$M'$ simulates $M$ on $w$.

If $x = \theta^n1^n$, accept.
Otherwise, run $M$ on $w$. If $M$ halts, accept.
THEOREM $\textit{REGULAR}_{\text{TM}}$ is undecidable.

PROOF By contradiction; assume $D$ decides $\textit{REGULAR}_{\text{TM}}$. Consider the following machine $H$:

$$H = \text{"On input } \langle M, w \rangle:\$$

Construct the machine $M' = \text{"On input } x:\$

If $x$ has the form $0^n1^n$, accept.

Otherwise, run $M$ on $w$.

If $M$ halts on $w$, accept.

Run $D$ on $\langle M' \rangle$.

If $D$ accepts, accept; if $D$ rejects, reject.”

We claim that $H$ is a decider and that $L(H) = H\textit{ALT}_{\text{TM}}$. To see that $H$ is a decider, note that after $H$ constructs $M'$, $H$ runs $D$ on $\langle M' \rangle$. Since $D$ is a decider, $D$ always halts. If $D$ accepts, $H$ accepts, and if $D$ rejects, $H$ rejects. Thus $H$ halts on all inputs.

To see that $L(H) = H\textit{ALT}_{\text{TM}}$, note that $H$ accepts $\langle M, w \rangle$ iff $D$ accepts $\langle M' \rangle$.

Since $D$ decides $\textit{REGULAR}_{\text{TM}}$, $D$ accepts $\langle M' \rangle$ iff $L(M')$ is regular. We claim that $L(M')$ is regular iff $M$ halts on $w$. To see this, note that if $M$ halts on $w$, $M'$ accepts all strings, either because the string has form $0^n1^n$ or because it accepts in the final step after $M$ halts. Thus $L(M') = \{0^n1^n \mid n \in \mathbb{N}_0\}$, which is not regular. Thus $H$ accepts $\langle M, w \rangle$ iff $M$ halts on $w$ iff $\langle M, w \rangle \in H\textit{ALT}_{\text{TM}}$, so $L(H) = H\textit{ALT}_{\text{TM}}$.

We have reached a contradiction, because we know that $H\textit{ALT}_{\text{TM}}$ is undecidable. Thus our assumption was wrong and $\textit{REGULAR}_{\text{TM}}$ is undecidable.

$\blacksquare$
Rice’s Theorem
The story so far

Consider the following problems:

Does $M$ accept $w$?
Does $M$ halt on $w$?
Does $M$ halt on all inputs?
Is $L(M)$ regular?
Does $M$ reject $\langle M \rangle$?
The story so far

Consider the following problems:

- Does $M$ accept $w$? \textit{Undecidable!}
- Does $M$ halt on $w$?
- Does $M$ halt on all inputs?
- Is $L(M)$ regular?
- Does $M$ reject $\langle M \rangle$?
The story so far

Consider the following problems:

- Does $M$ accept $w$? *Undecidable!*
- Does $M$ halt on $w$? *Undecidable!*
- Does $M$ halt on all inputs?
- Is $L(M)$ regular?
- Does $M$ reject $\langle M \rangle$?
The story so far

Consider the following problems:

- Does $M$ accept $w$? \textit{Undecidable!}
- Does $M$ halt on $w$? \textit{Undecidable!}
- Does $M$ halt on all inputs? \textit{Undecidable!}
- Is $L(M)$ regular?
- Does $M$ reject $\langle M \rangle$?
The story so far

Consider the following problems:

- Does $M$ accept $w$? \textit{Undecidable!}
- Does $M$ halt on $w$? \textit{Undecidable!}
- Does $M$ halt on all inputs? \textit{Undecidable!}
- Is $L(M)$ regular? \textit{Undecidable!}
- Does $M$ reject $\langle M \rangle$?
The story so far

Consider the following problems:

- Does \( M \) accept \( w \)? \( \text{Undecidable!} \)
- Does \( M \) halt on \( w \)? \( \text{Undecidable!} \)
- Does \( M \) halt on all inputs? \( \text{Undecidable!} \)
- Is \( L(M) \) regular? \( \text{Undecidable!} \)
- Does \( M \) reject \( \langle M \rangle \)? \( \text{Undecidable (and unrecognizable)!} \)
The story so far

Consider the following problems:

Does $M$ accept $w$? \hspace{1cm} \textit{Undecidable!}
Does $M$ halt on $w$? \hspace{1cm} \textit{Undecidable!}
Does $M$ halt on all inputs? \hspace{1cm} \textit{Undecidable!}
Is $L(M)$ regular? \hspace{1cm} \textit{Undecidable!}
Does $M$ reject $\langle M \rangle$? \hspace{1cm} \textit{Undecidable (and unrecognizable)!}

There seems to be a trend here.
It turns out that most interesting questions about properties of Turing machines (and thus computer programs) are undecidable!
A *property of an RE language* is some trait that may apply to RE languages.

For example:

- Does $L = \emptyset$?
- Is $L$ regular?
- Is $L$ context-free?
- Does $L$ contain any string of length exactly 137?
We can describe a property of an RE language as the set of RE languages with that property. 

If $P$ is a property of RE languages, consider the language 

$$L_P = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \in P \}$$

Note that membership in $L_P$ depends only on the language of a TM, not the description of that TM.

If $L(M_1) = L(M_2)$, then $\langle M_1 \rangle \in L_P$ iff $\langle M_2 \rangle \in L_P$. 

$P$ is the set of RE languages with property $P$. 

$L_P$ is the set of TMs that recognize a language with property $P$. 

\[ L_{\text{even}} = \{ \langle M \rangle | L(M) \text{ is finite and } |L(M)| \text{ is even} \} \]

This is a property of \textbf{RE} languages, because it depends \textit{purely} on the language of the TM and not on the TM itself.

Specifically, if \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{\text{even}} \) iff \( \langle M_2 \rangle \in L_{\text{even}} \).
\[ L_{\text{even}} = \{ \langle M \rangle \mid L(M) \text{ is finite and } |L(M)| \text{ is even} \} \]

This is a property of \textbf{RE} languages, because it depends \textit{purely} on the language of the TM and not on the TM itself.

Specifically, if \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{\text{even}} \) iff \( \langle M_2 \rangle \in L_{\text{even}} \)

\[ L_{\text{evenQ}} = \{ \langle M \rangle \mid M \text{ has an even number of states} \} \]

This is \textit{not} a property of \textbf{RE} languages, because it does not depend purely on the language of the TM.

Specifically, if \( L(M_1) = L(M_2) \), then it may be possible for \( \langle M_1 \rangle \in L_{\text{evenQ}} \) but \( \langle M_2 \rangle \notin L_{\text{evenQ}} \)
A property of **RE** languages is called *trivial* if *all* **RE** languages have the property or *no** **RE** languages have the property, e.g.,

\[ \{\langle M \rangle \mid L(M) \text{ is } \text{RE}\} \text{ is trivial} \]
\[ \{\langle M \rangle \mid L(M) \text{ is not } \text{RE}\} \text{ is trivial} \]

A property of **RE** languages is called *nontrivial* if there exist TMs $M_1$ and $M_2$ such that $\langle M_1 \rangle \in L_P$, but $\langle M_2 \rangle \notin L_P$, e.g.,

\[ \{\langle M \rangle \mid L(M) \text{ is infinite}\} \text{ is nontrivial} \]
\[ \{\langle M \rangle \mid L(M) \text{ is regular}\} \text{ is nontrivial} \]
\[ \{\langle M \rangle \mid L(M) \text{ is decidable}\} \text{ is nontrivial} \]
Rice’s Theorem

Any nontrivial property of the RE languages is **undecidable**.
Can we apply Rice’s Theorem to this language?

\[ L_{\text{ne}} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]
Can we apply Rice’s Theorem to this language?

\[ L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]

We can apply Rice’s Theorem if two conditions hold:
Can we apply Rice’s Theorem to this language?

\[ L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]

We can apply Rice’s Theorem if two conditions hold:

\text{\textbf{L}_ne \text{ is nontrivial:}}

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{ne} \land \langle M_2 \rangle \notin L_{ne} \]
Can we apply Rice’s Theorem to this language?

\[ L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]

We can apply Rice’s Theorem if two conditions hold:

✅ \textit{\textbf{L}}_{\textit{ne}} \textit{ is nontrivial:}

\[ \exists M_1 \cdot \exists M_2. \langle M_1 \rangle \in L_{ne} \land \langle M_2 \rangle \notin L_{ne} \]
Can we apply Rice’s Theorem to this language?

\[ L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]

We can apply Rice’s Theorem if two conditions hold:

- **L\(_{ne}\) is nontrivial:**

  \[ \exists M_1 \cdot \exists M_2 : \langle M_1 \rangle \in L_{ne} \land \langle M_2 \rangle \notin L_{ne} \]

- **L\(_{ne}\) is a property of RE languages:**

  If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{ne} \) iff \( \langle M_2 \rangle \in L_{ne} \)
Can we apply Rice’s Theorem to this language?

\[ L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]

We can apply Rice’s Theorem if two conditions hold:

✅ **\( L_{ne} \) is nontrivial:**

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{ne} \land \langle M_2 \rangle \notin L_{ne} \]

✅ **\( L_{ne} \) is a property of RE languages:**

If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{ne} \) iff \( \langle M_2 \rangle \in L_{ne} \)
Can we apply Rice’s Theorem to this language?

\[ L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \} \]

We can apply Rice’s Theorem if two conditions hold:

1. **\( L_{ne} \) is nontrivial:**
   \[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{ne} \land \langle M_2 \rangle \notin L_{ne} \]

2. **\( L_{ne} \) is a property of RE languages:**
   If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{ne} \) iff \( \langle M_2 \rangle \in L_{ne} \)

Rice’s Theorem applies, so \( L_{ne} \) is undecidable!
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{ \langle M \rangle \mid M \text{ has an even number of states} \} \]
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{ \langle M \rangle \mid M \text{ has an even number of states} \} \]

We can apply Rice’s Theorem if two conditions hold:
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{\langle M \rangle \mid M \text{ has an even number of states} \} \]

We can apply Rice’s Theorem if two conditions hold:

**\( L_{es} \text{ is nontrivial:} \)**

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \not\in L_{es} \]
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{\langle M \rangle \mid M \text{ has an even number of states} \} \]

We can apply Rice’s Theorem if two conditions hold:

✅ *Les is nontrivial:*

\[ \exists M_1 \cdot \exists M_2 \cdot \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \notin L_{es} \]
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{ \langle M \rangle \mid M \text{ has an even number of states} \} \]

We can apply Rice’s Theorem if two conditions hold:

✅ *L_{es} is nontrivial:*

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \notin L_{es} \]

*L_{es} is a property of RE languages:*

If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{es} \) iff \( \langle M_2 \rangle \in L_{es} \)
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{ \langle M \rangle \mid M \text{ has an even number of states} \} \]

We can apply Rice’s Theorem if two conditions hold:

✅ **\( L_{es} \) is nontrivial:**

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \notin L_{es} \]

❌ **\( L_{es} \) is a property of RE languages:**

If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{es} \) iff \( \langle M_2 \rangle \in L_{es} \)
Can we apply Rice’s Theorem to this language?

\[ L_{es} = \{ \langle M \rangle \mid M \text{ has an even number of states} \} \]

We can apply Rice’s Theorem if two conditions hold:

✅ \textit{\( L_{es} \) is nontrivial:}

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \not\in L_{es} \]

❌ \textit{\( L_{es} \) is a property of RE languages:}

If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{es} \iff \langle M_2 \rangle \in L_{es} \)

Rice’s Theorem does not apply, so we can’t draw any conclusion!
Can we apply Rice’s Theorem to this language?

\[ L_{\text{small}} = \{\langle M \rangle \mid \text{There is a five-state TM that recognizes } L(M) \} \]
Can we apply Rice’s Theorem to this language?

\[ L_{\text{small}} = \{\langle M \rangle \mid \text{There is a five-state TM} \]
\[ \quad \text{that recognizes } L(M) \}\]

We can apply Rice’s Theorem if two conditions hold:
Can we apply Rice’s Theorem to this language?

\[ L_{\text{small}} = \{ \langle M \rangle \mid \text{There is a five-state TM that recognizes } L(M) \} \]

We can apply Rice’s Theorem if two conditions hold:

**\( L_{\text{small}} \text{ is nontrivial:} \)**

\[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{\text{es}} \land \langle M_2 \rangle \notin L_{\text{es}} \]
Can we apply Rice’s Theorem to this language?

$L_{\text{small}} = \{ \langle M \rangle \mid \text{There is a five-state TM that recognizes } L(M) \}$

We can apply Rice’s Theorem if two conditions hold:

✅ $L_{\text{small}}$ is nontrivial:

\[
\exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{\text{es}} \land \langle M_2 \rangle \not\in L_{\text{es}}
\]
Can we apply Rice’s Theorem to this language?

\[ L_{\text{small}} = \{ \langle M \rangle \mid \text{There is a five-state TM that recognizes } L(M) \} \]

We can apply Rice’s Theorem if two conditions hold:

- **\( L_{\text{small}} \) is nontrivial:**
  \[
  \exists M_1 \cdot \exists M_2 . \langle M_1 \rangle \in L_{\text{es}} \land \langle M_2 \rangle \notin L_{\text{es}}
  \]

- **\( L_{\text{small}} \) is a property of RE languages:**
  If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{\text{small}} \) iff \( \langle M_2 \rangle \in L_{\text{small}} \)
Can we apply Rice’s Theorem to this language?

\[ L_{\text{small}} = \{ \langle M \rangle \mid \text{There is a five-state TM that recognizes } L(M) \} \]

We can apply Rice’s Theorem if two conditions hold:

- **\( L_{\text{small}} \) is nontrivial:**
  \[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{\text{es}} \land \langle M_2 \rangle \notin L_{\text{es}} \]

- **\( L_{\text{small}} \) is a property of RE languages:**
  
  If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in L_{\text{small}} \) iff \( \langle M_2 \rangle \in L_{\text{small}} \)
Can we apply Rice’s Theorem to this language?

\[ L_{\text{small}} = \{ \langle M \rangle \mid \text{There is a five-state TM that recognizes } L(M) \} \]

We can apply Rice’s Theorem if two conditions hold:

- **\( L_{\text{small}} \) is nontrivial:**
  \[ \exists M_1 \cdot \exists M_2 . \langle M_1 \rangle \in L_{\text{es}} \land \langle M_2 \rangle \notin L_{\text{es}} \]

- **\( L_{\text{small}} \) is a property of RE languages:**
  \[ \text{If } L(M_1) = L(M_2), \text{ then } \langle M_1 \rangle \in L_{\text{small}} \text{ iff } \langle M_2 \rangle \in L_{\text{small}} \]

Rice’s Theorem applies, so \( L_{\text{small}} \) is undecidable!
Rice’s Theorem tells us that all of the following problems are undecidable:

$L_{\text{palindrome}} = \{ \langle M \rangle \mid \text{every string in } L(M) \text{ is a palindrome} \}$

$L_{\text{allodd}} = \{ \langle M \rangle \mid \text{every string in } L(M) \text{ has odd length} \}$

$L_{\text{CFL}} = \{ \langle M \rangle \mid L(M) \text{ is a context-free language} \}$

$L_{\text{short}} = \{ \langle M \rangle \mid L(M) \text{ has no strings of length greater than 5} \}$

$L_{\text{decidable}} = \{ \langle M \rangle \mid L(M) \text{ is decidable} \}$

$E_{\text{TM}} = \{ \langle M \rangle \mid L(M) = \emptyset \}$
The proof of Rice’s theorem is a generalization of the reductions we’ve seen so far.

We won’t have time to go through the proof in detail, but the general idea is: If $L_P$ is a nontrivial property of RE languages, show that we can reduce $\text{HALT}_{TM}$ to $L_P$. 
THEOREM (Rice) Any nontrivial property of the RE languages is undecidable.

PROOF Let $M_\emptyset$ be a TM that accepts the empty language. Then either $\langle M_\emptyset \rangle \notin L_P$ or $\langle M_\emptyset \rangle \in L_P$. In the former case, we prove that $L_P$ is undecidable directly. In the latter case, we prove that $L_P$ is undecidable; this means that $L_P$ is undecidable as well. So assume without loss of generality that $\langle M_\emptyset \rangle \notin L_P$. Assume for the sake of contradiction that $L_P$ is a decidable property of the RE languages.

Since $L_P$ is a nontrivial property of the RE languages and $\langle M_\emptyset \rangle \notin L_P$, there must be some TM $M_{yes}$ such that $\langle M_{yes} \rangle \in L_P$. Moreover, $L(M_{yes}) \neq L(M_\emptyset)$, because otherwise if $L(M_{yes}) = L(M_\emptyset)$, we would have that either $\langle M_\emptyset \rangle \in L_P$ and $\langle M_{yes} \rangle \in L_P$ or $\langle M_\emptyset \rangle \notin L_P$ and $\langle M_{yes} \rangle \notin L_P$, both of which are false.

Because $L_P$ is decidable, let $D$ be a decider for $L_P$. Then consider the following TM:

$$H = \text{"On input } \langle M, w \rangle:\n$$
$$\text{Construct the TM } M' = \text{"On input } x:\n$$
$$\text{Run } M \text{ on } w, \text{ and if } M \text{ halts, run } M_{yes} \text{ on } x.\n$$
$$\text{Accept if } M_{yes} \text{ accepts; reject if } M_{yes} \text{ rejects."
}$$
$$\text{Run } D \text{ on } \langle M' \rangle \text{ and accept if } D \text{ accepts and reject if } D \text{ rejects."}
$$

We claim that $H$ is a decider for $HALT_{TM}$. If we can show this, then we have reached a contradiction because $HALT_{TM}$ is undecidable. Thus our assumption was wrong, so no nontrivial property of the RE languages is decidable.

To see that $H$ is a decider, note that since $D$ is a decider, after we construct $M'$ and run $D$ on $\langle M' \rangle$, $D$ always halts, so $H$ always halts. Thus $H$ halts on all inputs.

To see that $L(H) = HALT_{TM}$, note that if $M$ loops on $w$, then $L(M') = \emptyset$, and so $D$ rejects $\langle M' \rangle$ because $L(M') = \emptyset = L(M_\emptyset)$ and $\langle M_\emptyset \rangle \notin L_P$. Otherwise, if $M$ halts on $w$, then $M'$ accepts $x$ iff $M_{yes}$ accepts $x$, so $L(M') = L(M_{yes})$, and since $\langle M_{yes} \rangle \in L_P$, $D$ accepts $\langle M' \rangle$. Thus $H$ accepts $\langle M, w \rangle$ iff $D$ accepts $\langle M' \rangle$ iff $M$ halts on $w$ iff $\langle M, w \rangle \in HALT_{TM}$, so $L(H) = HALT_{TM}$. $\blacksquare$
Where we stand

*The Church–Turing thesis* tells us that TMs give us a mechanism for studying computation in the abstract.

*Universal computers* – computers as we know them – are not just a stroke of luck. The existence of the universal TM ensures that such computers exist.

*Self-reference* is an inherent consequence of computational power.

*Undecidable problems* exist partially as a consequence of the above and indicate that there are statements whose truth can’t be determined by computational processes.

*Unrecognizable problems* are out there and can be discovered via diagonalization. They imply there are limits to mathematical proof.

*Reductions* let us prove a connection between problems, showing they’re of the same difficulty (decidable, undecidable, recognizable, unrecognizable).

*Rice’s Theorem* can be used to easily prove that large classes of languages are also undecidable.
Acknowledgments

This lecture incorporates material from:

W. Daniel Hillis, *The Pattern on the Stone*
Nancy Ide, Vassar College
Keith Schwarz, Stanford University
Michael Sipser, *Introduction to the Theory of Computation*