Computability and Reductions

30 November 2021
Where are we?
The class of Turing-decidable languages (R) represents problems that can be solved by a computer.

The class of Turing-recognizable languages (RE) represents problems where “yes” answers can be verified by a computer.
We’ve already seen how to use self-reference to find languages that are *undecidable* – languages that are not in $R$. 
So what?

Problems like the Halting Problem might not seem all that exciting, so who cares if we can’t solve them?

It turns out, this same line of reasoning can be used to show that some very important problems are impossible to solve.
Analogy time!
Engineering problem

Design a diesel engine that doesn’t emit lots of NO$_x$ pollutants.
Engineering problem

Design a diesel engine that doesn’t emit lots of NO\textsubscript{x} pollutants.

Engineering prowess!
Engineering problem

Design a diesel engine that doesn’t emit lots of NO\textsubscript{x} pollutants.

Engineering prowess!

Awesome engine!
**Engineering problem**

Design a diesel engine that doesn’t emit lots of NO\textsubscript{x} pollutants.

**Engine testing regimen**

**Regulatory problem**

Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO\textsubscript{x} pollutants.
**Engineering problem**

*Design a diesel engine that doesn’t emit lots of NO\textsubscript{x} pollutants.*

**Regulatory problem**

*Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO\textsubscript{x} pollutants.*
**Engineering problem**

Design a diesel engine that doesn’t emit lots of NO\textsubscript{x} pollutants.

**Regulatory problem**

Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO\textsubscript{x} pollutants.
FACT: Almost all “regulatory problems” about computer programs are undecidable. That is, almost all problems of the form “does this program have [behavioral property X]” are undecidable.

This can be formalized through a result called Rice’s Theorem. (See the appendix to these slides – and the textbook – for more details!)
Beyond R and RE
All languages

Regular languages

CFLs

$A_{TM}$

$HALT_{TM}$

$R$

$RE$

Anything out here?
Intuitively, a language is \textit{not} in RE if there’s no general way to prove that a given string $w \in L$ actually belongs to $L$.

In other words, even if a string is in the language, you might not be able to convince someone of it!
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.
Languages, TMs, and TM encodings

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

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

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.
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>
</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</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>
</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>
</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</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>
</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>$\ldots$</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\ldots$</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>
</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</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>
</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></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>
</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>
</tbody>
</table>

\( \vdots \)
<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>...</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>...</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>
</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>...</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>
</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>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</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>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Flip all “accept”s to “no”s and vice versa.
<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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>…</td>
<td></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>...</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>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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</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>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>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>...</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>No</td>
<td>Acc</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>...</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>M0</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M1</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M4</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M5</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></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>( \ldots )</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>( \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></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>$M_0$</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>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>No</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></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>No</td>
<td>Acc</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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</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>…</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>M₀</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>M₁</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>M₂</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>M₃</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>M₄</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>M₅</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>

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>( \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>( \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>
</tbody>
</table>
The language of all TMs that do not accept their own descriptions.

<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>(\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></td>
</tr>
</tbody>
</table>
\[ \{ \langle M \rangle \mid M \text{ is a TM that does not accept } \langle M \rangle \} \]

<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>( \text{No} )</td>
<td>( \text{No} )</td>
<td>( \text{No} )</td>
<td>( \text{Acc} )</td>
<td>( \text{No} )</td>
<td>( \text{Acc} )</td>
<td>( \ldots )</td>
<td></td>
</tr>
</tbody>
</table>
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, $L_D \notin \text{RE}$!
$$L_D = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ does not accept } \langle M \rangle \}$$

**THEOREM.** $L_D \notin \text{RE}$. 

**PROOF.** By contradiction; assume that $L_D \in \text{RE}$. This means that there is a Turing machine $R$ such that $L(R) = L_D$.

Since $R$ is a recognizer for $L_D$, we see that

$$R \text{ accepts } \langle R \rangle \text{ if and only if } \langle R \rangle \in L_D.$$ 

By the definition of $L_D$, we know that

$$\langle R \rangle \in L_D \text{ if and only if } R \text{ does not accept } \langle R \rangle.$$ 

Combining the two statements above tell us that

$$R \text{ accepts } \langle R \rangle \text{ if and only if } R \text{ does not accept } \langle R \rangle.$$ 

We’ve reached a contradiction, so our assumption was wrong and $L_D \notin \text{RE}$. ■
The diagram illustrates the relationship between different classes of languages in computational theory:

- **Regular languages** are the smallest class and form the innermost circle.
- **CFLs** (Context-Free Languages) are the next level, encompassing regular languages.
- **RE** (Recursive or Turing-recognizable) languages include CFLs.
- **R** (Turing-computable, Recursive) languages are even more encompassing, including RE.
- **CFLs** and **RE** are subsets of **ALL LANGUAGES**.

Notable languages that are not part of these classes include:
- **L_D**: Languages that are not recursive.
- **HALT_{TM}**: Languages that are undecidable.
- **A_{TM}**: Languages that are recursively enumerable but not recursive.
Unsolvable problems
Undecidability and non-recognizability

We’ve seen languages like

\[ \text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ halts on } w \} \]

which is recognizable but undecidable, and

\[ L_D = \{ \langle M \rangle \mid M \text{ does not accept } \langle M \rangle \} \]

which is non-recognizable.

Can we show other problems are also undecidable or non-recognizable?
The tools so far

How do we prove that languages are not regular?

*Pumping Lemma*: Explicitly prove that some language is not regular.

*Closure properties*: Show that a closure property of regular language does not hold for the language.

How do we prove that languages are not context-free?

*Pumping Lemma*: Explicitly prove that some language is not context-free.

*Closure properties*: Show that a closure property of context-free languages does not hold for the language.
The Pumping Lemma for TMs

Pumping lemma proof works by forcing a model of computation to repeat itself. Having an infinite tape breaks this logic.
Closure properties for R and RE

R is closed under:
- Union
- Intersection
- Complement
- Concatenation
- Kleene star

RE is closed under:
- Union
- Intersection
- Concatenation
- Kleene star
However, we’ll usually prove a language is undecidable or non-recognizable using *proof by reduction*.
Proof by reduction
Suppose you want to build a recognizer for

\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

Do you need to start from scratch?

What Turing machine could you use to recognize this language?
Suppose you want to build a recognizer for

\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

Do you need to start from scratch?

What Turing machine could you use to recognize this language?

The universal TM \( U \) recognizes \( A_{TM} \), i.e., for any TM \( M \) and string \( w \), \( \langle M, w \rangle \in A_{TM} \) iff \( M \) accepts \( w \).
Suppose you want to build a recognizer for

\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

Do you need to start from scratch?

What Turing machine could you use to recognize this language?
Suppose you want to build a recognizer for

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Do you need to start from scratch?

What Turing machine could you use to recognize this language?

$$\langle M \rangle \in L \iff \langle M, \varepsilon \rangle \in A_{TM}$$
Suppose you want to build a recognizer for

\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

Do you need to start from scratch?

What Turing machine could you use to recognize this language?

\[ \langle M \rangle \in L \iff \langle M, \varepsilon \rangle \in A_{TM} \]

Make a TM that accepts \( \langle M \rangle \)

 iff U accepts \( \langle M, \varepsilon \rangle \)
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]
$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

\[ H = \text{“On input } \langle M \rangle:\]
1. Construct the string \( \langle M, \varepsilon \rangle \).
2. Run \( U \) on \( \langle M, \varepsilon \rangle \).
3. If \( U \) accepts \( \langle M, \varepsilon \rangle \), then \textit{accept}. 
4. If \( U \) rejects \( \langle M, \varepsilon \rangle \), then \textit{reject}.”
$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \epsilon \}$

$H = \text{"On input } \langle M \rangle \text{:}
1. \text{Construct the string } \langle M, \epsilon \rangle.
2. \text{Run } U \text{ on } \langle M, \epsilon \rangle.
3. \text{If } U \text{ accepts } \langle M, \epsilon \rangle, \text{ then accept.}
4. \text{If } U \text{ rejects } \langle M, \epsilon \rangle, \text{ then reject."}$

What does $H$ do input $\langle M \rangle$ if $M$ accepts $\epsilon$?
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

\[ H = \text{“On input } \langle M \rangle: \]

1. Construct the string \( \langle M, \varepsilon \rangle \).
2. Run \( U \) on \( \langle M, \varepsilon \rangle \).
3. If \( U \) accepts \( \langle M, \varepsilon \rangle \), then \text{accept.}\n4. If \( U \) rejects \( \langle M, \varepsilon \rangle \), then \text{reject.} \]

What does \( H \) do input \( \langle M \rangle \) if \( M \) accepts \( \varepsilon \)? \text{Accepts!}
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

\[
H = \text{"On input } \langle M \rangle : \text{"
1. Construct the string } \langle M, \varepsilon \rangle .
2. Run } U \text{ on } \langle M, \varepsilon \rangle .
3. \text{If } U \text{ accepts } \langle M, \varepsilon \rangle , \text{ then } \text{accept.}
4. \text{If } U \text{ rejects } \langle M, \varepsilon \rangle , \text{ then } \text{reject."} \]

What does } H \text{ do input } \langle M \rangle \text{ if } M \text{ accepts } \varepsilon? \text{ Accepts!}

What does } H \text{ do input } \langle M \rangle \text{ if } M \text{ rejects } \varepsilon?
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

\[
H = \text{“On input } \langle M \rangle \text{:} \\
1. \text{Construct the string } \langle M, \varepsilon \rangle. \\
2. \text{Run } U \text{ on } \langle M, \varepsilon \rangle. \\
3. \text{If } U \text{ accepts } \langle M, \varepsilon \rangle, \text{ then accept.} \\
4. \text{If } U \text{ rejects } \langle M, \varepsilon \rangle, \text{ then reject.”} \\
\]

What does \( H \) do input \( \langle M \rangle \) if \( M \) accepts \( \varepsilon \)? **Accepts**!

What does \( H \) do input \( \langle M \rangle \) if \( M \) rejects \( \varepsilon \)? **Rejects**!
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

\begin{align*}
H &= \text{"On input } \langle M \rangle: \\
1. \text{ Construct the string } \langle M, \varepsilon \rangle. \\
2. \text{ Run } U \text{ on } \langle M, \varepsilon \rangle. \\
3. \text{ If } U \text{ accepts } \langle M, \varepsilon \rangle, \text{ then accept.} \\
4. \text{ If } U \text{ rejects } \langle M, \varepsilon \rangle, \text{ then reject."}
\end{align*}

What does \( H \) do input \( \langle M \rangle \) if \( M \) accepts \( \varepsilon \)? \textit{Accepts}!

What does \( H \) do input \( \langle M \rangle \) if \( M \) rejects \( \varepsilon \)? \textit{Rejects}!

What does \( H \) do input \( \langle M \rangle \) if \( M \) loops on \( \varepsilon \)?
$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$

$H = \text{“On input } \langle M \rangle: \text{”}$

1. Construct the string $\langle M, \varepsilon \rangle$.
2. Run $U$ on $\langle M, \varepsilon \rangle$.
3. If $U$ accepts $\langle M, \varepsilon \rangle$, then $\text{accept}$.
4. If $U$ rejects $\langle M, \varepsilon \rangle$, then $\text{reject}$.

What does $H$ do input $\langle M \rangle$ if $M$ accepts $\varepsilon$? $\text{Accepts!}$

What does $H$ do input $\langle M \rangle$ if $M$ rejects $\varepsilon$? $\text{Rejects!}$

What does $H$ do input $\langle M \rangle$ if $M$ loops on $\varepsilon$? $\text{Loops!}$
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

\[ H = \text{“On input } \langle M \rangle : \]
1. Construct the string \( \langle M, \varepsilon \rangle \).
2. Run U on \( \langle M, \varepsilon \rangle \).
3. If U accepts \( \langle M, \varepsilon \rangle \), then accept.
4. If U rejects \( \langle M, \varepsilon \rangle \), then reject.”

What does \( H \) do input \( \langle M \rangle \) if \( M \) accepts \( \varepsilon \)? \textit{Accepts}!

What does \( H \) do input \( \langle M \rangle \) if \( M \) rejects \( \varepsilon \)? \textit{Rejects}!

What does \( H \) do input \( \langle M \rangle \) if \( M \) loops on \( \varepsilon \)? \textit{Loops}!

\( H \) accepts \( \langle M \rangle \) iff U accepts \( \langle M, \varepsilon \rangle \) iff \( M \) accepts \( \varepsilon \) iff \( \langle M \rangle \in L \).

Therefore, \( L(H) = L \).
$H = \text{“On input } \langle M \rangle:\text{“}
1. Construct the string $\langle M, \epsilon \rangle$. 
2. Run $U$ on $\langle M, \epsilon \rangle$.
3. If $U$ accepts $\langle M, \epsilon \rangle$, then \textit{accept}.
4. If $U$ rejects $\langle M, \epsilon \rangle$, then \textit{reject}.”

$H = \text{“On input } \langle M \rangle:\text{“}
1. Run $M$ on $\epsilon$.
2. If $M$ accepts $\epsilon$, then \textit{accept}.
3. If $M$ rejects $\epsilon$, then \textit{reject}.”
We can use the same approach for lots of problems:

We give a special input to an existing TM and decide whether to accept or reject based on what it does.
From $\text{HALT}_\text{TM}$ to $\text{A}_\text{TM}$
From $\text{HALT}_{TM}$ to $A_{TM}$

- Change $M$ to loop instead of reject
- Decider for $\text{HALT}_{TM}$.
From $\text{HALT}_{\text{TM}}$ to $A_{\text{TM}}$

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

1. Build $M$ into $M'$ so $M'$ loops when $M$ rejects.
2. Run $D$ on $\langle M', w \rangle$.
3. If $D$ accepts $\langle M', w \rangle$, then accept.
4. If $D$ rejects $\langle M', w \rangle$, then reject.”
The general pattern

\[ H \]

\[ H = \text{"On input } w:\text{ }
\]

1. Transform the input } w \text{ into } f(w).
2. Run machine } R \text{ on } f(w).
3. If } R \text{ accepts } f(w), \text{ then } \textit{accept}.
4. If } R \text{ rejects } f(w), \text{ then } \textit{reject}.\]
Reductions

Intuitively, problem $A$ reduces to problem $B$ iff a solver for $B$ can be used to solve $A$.

Reductions can be used to show certain problems are “solvable”:

If $A$ reduces to $B$ and $B$ is “solvable”, then $A$ is “solvable”.
Defining reductions

Formally, a *reduction* from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that for any $w \in \Sigma_1^*$,

- $w \in A$ iff $f(w) \in B$:
  - Every $w \in A$ maps to some $f(w) \in B$.
  - Every $w \notin A$ maps to some $f(w) \notin B$. 
Why reductions matter

If language $A$ reduces to language $B$,

we can use a recognizer for $B$ to recognize $A$ and

we can use a decider for $B$ to decide $A$.

Why is this the case?
Why reductions matter

If language $A$ reduces to language $B$, we can use a recognizer for $B$ to recognize $A$ and we can use a decider for $B$ to decide $A$. Why is this the case?
$w \in A \text{ iff } f(w) \in B$
\[ w \in A \iff f(w) \in B \]
\[ w \in A \iff f(w) \in B \]
$w \in A \text{ iff } f(w) \in B$
\( w \in A \text{ iff } f(w) \in B \)

**Diagram:**

\[ H \]

Compute \( f \) \[ w \rightarrow \text{Compute } f \]

\( f(w) \) \[ f(w) \rightarrow \text{TM for language } B \]

\[ R \]

If \( R \) accepts \( f(w) \), then accept.

If \( R \) rejects \( f(w) \), then reject.

**Text:**

\( H = \) “On input \( w \):

1. Transform the input \( w \) into \( f(w) \).
2. Run machine \( R \) on \( f(w) \).
3. If \( R \) accepts \( f(w) \), then accept.
4. If \( R \) rejects \( f(w) \), then reject.”
$w \in A$ iff $f(w) \in B$

$H = \text{"On input } w:\n\begin{align*}
1. & \text{ Transform the input } w \text{ into } f(w). \\
2. & \text{ Run machine } R \text{ on } f(w). \\
3. & \text{ If } R \text{ accepts } f(w), \text{ then accept.} \\
4. & \text{ If } R \text{ rejects } f(w), \text{ then reject.}\n\end{align*}$
$w \in A$ iff $f(w) \in B$

$H = \text{"On input } w:\n\begin{enumerate}
\item Transform the input } w \text{ into } f(w).
\item Run machine } R \text{ on } f(w).
\item If } R \text{ accepts } f(w), \text{ then } \text{accept.}
\item If } R \text{ rejects } f(w), \text{ then } \text{reject."
\end{enumerate}$

$H \text{ accepts } w$ iff $R \text{ accepts } f(w)$
$w \in A \text{ iff } f(w) \in B$

$H = \text{"On input } w:"

1. Transform the input $w$ into $f(w)$.
2. Run machine $R$ on $f(w)$.
3. If $R$ accepts $f(w)$, then \textit{accept}.
4. If $R$ rejects $f(w)$, then \textit{reject}.

$H$ accepts $w$ if $R$ accepts $f(w)$ if $f(w) \in B$
$w \in A \iff f(w) \in B$

**$H$**

1. **Compute $f$**
2. **Transform the input $w$ into $f(w)$**
3. **Run machine $R$ on $f(w)$**
4. **If $R$ accepts $f(w)$, then accept.**
   
5. **If $R$ rejects $f(w)$, then reject.**

$H$ accepts $w$ iff $R$ accepts $f(w)$
iff $f(w) \in B$
iff $w \in A$
$w \in A$ iff $f(w) \in B$

$H = \text{"On input } w:"
1. Transform the input $w$ into $f(w)$.
2. Run machine $R$ on $f(w)$.
3. If $R$ accepts $f(w)$, then accept.
4. If $R$ rejects $f(w)$, then reject.$$

L(H) = A$
A problem

We said $f$ is a reduction from $A$ to $B$ iff

$$w \in A \iff f(w) \in B$$

However, under this definition, any language $A$ reduces to any language $B$ unless $B = \emptyset$ or $\Sigma^*$.

There must be some string $w_{yes} \in B$ and some string $w_{no} \notin B$, so define $f : \Sigma_1^* \rightarrow \Sigma_2^*$ as follows:

$$f(w) = \begin{cases} w_{yes} & \text{if } w \in A \\ w_{no} & \text{if } w \notin A \end{cases}$$

Then $f$ is a reduction from $A$ to $B$. 
A problem

*Example*: Let’s reduce $L_D$ to $0^*1^*$.

Take $w_{yes} = 01$, $w_{no} = 10$.

Then $f(w)$ is defined as

$$f(w) = \begin{cases} 
01 & \text{if } w \in L_D \\
10 & \text{if } w \notin L_D 
\end{cases}$$

What’s wrong?
A problem

*Example:* Let’s reduce $L_D$ to $0^*1^*$.  

Take $w_{\text{yes}} = 01$, $w_{\text{no}} = 10$. 

Then $f(w)$ is defined as 

$$f(w) = \begin{cases} 
01 & \text{if } w \in L_D \\
10 & \text{if } w \not\in L_D 
\end{cases}$$

What’s wrong?

There’s no TM that can actually evaluate the function $f(w)$ on all inputs, since no TM can decide whether or not $w \in L_D$!
We need to introduce the requirement that \( f : \Sigma_1^* \rightarrow \Sigma_2^* \) is a \textit{computable function}.

That is, there is some TM \( M \) with the following behavior:

On input \( w \):
1. Compute \( f(w) \) and write it on the tape.
2. Move the tape head to the start of \( f(w) \).
3. Halt.
Computable functions

\[ f(1^n) = 1^{3n+1} \]
Computable functions

\[ f(1^n) = 1^{3n+1} \]
Computable functions

\[ f(w) = \begin{cases} 1^{mn} & \text{if } w = 1^n1^m \\ \varepsilon & \text{otherwise} \end{cases} \]
Computable functions

\[ f(w) = \begin{cases} \ 1^{mn} & \text{if } w = 1^n1^m \\ \varepsilon & \text{otherwise} \end{cases} \]
Mapping reductions

A function \( f : \Sigma_1^* \rightarrow \Sigma_2^* \) is a *mapping reduction* from \( A \) to \( B \) iff

For any \( w \in \Sigma_1^* \), \( w \in A \) iff \( f(w) \in B \), and

\( f \) is a computable function.

Intuitively, a mapping reduction from \( A \) to \( B \) says that a computer can transform any instance of \( A \) into an instance of \( B \) such that the answer to \( B \) is the answer to \( A \).
If there is a mapping reduction from language $A$ to language $B$, we say that language $A$ is **mapping reducible** to language $B$.

Notation: $A \leq_m B$ iff language $A$ is mapping reducible to language $B$. 

**Note that we reduce languages, not machines!**
$A \leq_m B$

$H = \text{“On input } w:\$

1. Compute $f(w)$.
2. Run machine $R$ on $f(w)$.
3. If $R$ accepts $f(w)$, then accept.
4. If $R$ rejects $f(w)$, then reject.
If $R$ is a decider for $B$, then $H$ is a decider for $A$. 

$H = \text{“On input } w:\n$  
1. Compute $f(w)$.  
2. Run machine $R$ on $f(w)$.  
3. If $R$ accepts $f(w)$, then accept.  
4. If $R$ rejects $f(w)$, then reject.”
A \leq_m B

H = “On input w:
   1. Compute f(w).
   2. Run machine R on f(w).
   3. If R accepts f(w), then accept.
   4. If R rejects f(w), then reject.”

If R is a decider for B, then H is a decider for A.
If R is a recognizer for B, then H is a recognizer for A.
Why mapping reducibility matters

THEOREM. If $B \in \mathbf{R}$ and $A \leq_m B$, then $A \in \mathbf{R}$.

THEOREM. If $B \in \mathbf{RE}$ and $A \leq_m B$, then $A \in \mathbf{RE}$.

*Intuitively:* $A \leq_m B$ means “$A$ is not harder than $B$.”
Why mapping reducibility matters

**THEOREM.** If $B \not\in R$ and $A \leq_m B$, then $A \not\in R$.

**THEOREM.** If $B \not\in RE$ and $A \leq_m B$, then $A \not\in RE$.

*Intuitively:* $A \leq_m B$ means “$B$ is at least as hard as $A$.”
A \leq_m B

If this one is “easy” \((R, RE)\) …

… then this one is “easy” \((R, RE)\) too.
If this one is "hard" (not R, not RE) …

\[ A \leq_m B \]

… then this one is "hard" (not R, not RE) too.
Proof by reduction

Suppose that we are given a language $L$ that we believe is undecidable.

We can prove that this is true using the following technique:

Assume, for the sake of contradiction, that $L$ is decidable.

Show how a decider for $L$ could be used to construct a decider for an undecidable language.

Conclude that $L$ must not be decidable.
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 that a problem is undecidable or unrecognizable by relating it to a problem where this is already known.
Appendix: Rice’s Theorem
In practice, we don’t need to write out full reduction proofs.

Instead, we can use Rice’s Theorem.
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

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

(The set of TMs that recognize a language with property $P$.)

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 $\langle M_2 \rangle \in L_P$
\[ L_{\text{even}} = \{ \langle M \rangle \mid L(M) \text{ is finite and } |L(M)| \text{ is even} \} \]

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{even}} = \{\langle M \rangle \mid L(M) \text{ is finite and } |L(M)| \text{ is even}\}**

This is a property of **RE** languages, because it depends **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 **not** a property of **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.,

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

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?

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

$L_{ne}$ is nontrivial:
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:

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

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

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

- **\( L_{\text{ne}} \) is nontrivial:**

  \[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{\text{ne}} \land \langle M_2 \rangle \notin L_{\text{ne}} \]
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:**
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:**
  \[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{ne} \land \langle M_2 \rangle \not\in L_{ne} \]

- **\( L_{ne} \) 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 $\langle M_1 \rangle \in L_{ne}$ if and only if $\langle M_2 \rangle \in L_{ne}$
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 $\langle M_2 \rangle \in L_{ne}$

Rice’s Theorem applies; $L_{ne}$ 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:

$L_{es}$ is nontrivial:
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:

$\exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \notin L_{es}$
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:*

$$\exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \notin L_{es}$$
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:
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:*

$$\exists M_1 \cdot \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \not\in L_{es}$$

❌ *$L_{es}$ 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 \( \langle M_2 \rangle \in L_{es} \)

Rice’s Theorem does not apply.
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?

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

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:

\[ L_{\text{small}} \text{ is nontrivial:} \]
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:

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

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

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

✅ **$L_{small}$ is nontrivial:**

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

$L_{small}$ is a property of RE languages:
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:

1. $L_{\text{small}}$ is nontrivial:
   \[ \exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{\text{es}} \land \langle M_2 \rangle \notin L_{\text{es}} \]

2. $L_{\text{small}}$ 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?

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

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

- **$L_{small}$ is nontrivial:**
  
  $\exists M_1 . \exists M_2 . \langle M_1 \rangle \in L_{es} \land \langle M_2 \rangle \notin L_{es}$

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

Rice’s Theorem applies; $L_{small}$ 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 class, but the general idea is: If $L_P$ is a nontrivial property of RE languages, show that we can reduce $\text{HALT}_{TM}$ to $L_P$. 
Summary

Reductions from known undecidable problems like $A_{TM}$ and $HALT_{TM}$ can be used to prove that certain languages are undecidable.

Rice’s Theorem can be used to 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*