Computability and Reductions

28 November 2023
Computability and Reductions
Computability and Reductions
IMPROVING QUASI-EXPERIMENTS USING MACHINE LEARNING

Many of the questions asked in data science are causal in nature, as the end goal is often to make better decisions. For example, research has previously shown that women who undergo menopausal hormone therapy tend to have a lower risk of cardiovascular disease. Physicians thus believed that the therapy reduced cardiovascular disease risk and widely prescribed it, an intervention that was later found to be incorrect as the therapy and cardiovascular risk were only correlated.

How can we avoid such erroneous causal conclusions? Establishing causality is challenging as we often only have access to observational data (data where we do not get to randomize who is treated), as randomized experiments can be expensive or unethical. Quasi-experiments are a type of study where treatments are “quasi-randomized” naturally in real-world situations, like patients who are randomly assigned to doctors when visiting the hospital. This real-world randomization can help us measure causality. However, it can be difficult to know when and where we can apply these quasi-experiments, which limits their usefulness. In this talk, I’ll show how machine learning can be used to improve quasi-experiments, increasing their feasibility across healthcare and social science settings.

PhD candidate at the University of Pennsylvania,

Tony Liu

Tuesday 11/28/2023

SP 105 4–5 pm

Light refreshments
Assignments:

Optional: Extra-credit assignment (parts 1 and 2)
  Out now
  Due last day of class
Assignment 10
  Coming soon (today?)

End of the semester means we’re off the regular schedule; take note of the due dates for the assignment and corrections.
We’ll do the end of course survey (ECS) at the end of class on Thursday.
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 might be impossible to rule out).
We saw how to use self-reference to find languages that are *undecidable* – languages that are not in $R$. 
def will_accept(fn: str, w: str) -> bool:
    """Returns True if fn(w) returns true. Otherwise, returns False."""
    ...

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

A self-defeating object

Using that object against itself
THEOREM $A_{TM} \notin \mathcal{R}$.

PROOF By contradiction; assume that $A_{TM} \in \mathcal{R}$. Then there is some decider $D$ for $A_{TM}$. We can represent $D$ as a function

\[
\text{will\_accept}(\text{fn: str, w: str}) \rightarrow \text{bool}
\]

that takes in the source code of a function $\text{fn}$ and a string $w$, then returns true if $\text{fn}(w)$ returns true and returns false otherwise.

Given this, consider this function:

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

Choose any string $s$. We consider two cases:

**Case 1:** $\text{will\_accept}(me, input)$ returns true. Since $\text{will\_accept}$ decides $A_{TM}$, this means $\text{trickster}(s)$ returns true. However, given how trickster is written, in this case $\text{trickster}(s)$ returns false.

**Case 2:** $\text{will\_accept}(me, input)$ returns false. Since $\text{will\_accept}$ decides $A_{TM}$, this means $\text{trickster}(s)$ doesn't return true. However, given how trickster is written, in this case $\text{trickster}(s)$ returns true.

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

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

Given this, consider this TM:

$M = \text{"On input } w:\n1. \text{ Have } M \text{ obtain its own description } \langle M \rangle.\n2. \text{ Run } D \text{ on } \langle M, w \rangle \text{ and see what it says.}\n3. \text{ If } D \text{ says that } M \text{ will accept } w, \text{ reject.}\n4. \text{ If } D \text{ says that } M \text{ will not accept } w, \text{ accept."}$

Choose any string $s$. We consider two cases:

**Case 1:** $D$ accepts $\langle M, s \rangle$. Since $D$ decides $A_{TM}$, this means $M$ accepts $s$. However, given how $M$ is written, in this case $M$ rejects $s$.

**Case 2:** $D$ rejects $\langle M, s \rangle$. Since $D$ decides $A_{TM}$, this means $M$ does not accept $s$. However, given how $M$ is written, in this case $M$ accepts $s$.

In both cases, we reach a contradiction, so our assumption must have been wrong. Therefore, $A_{TM} \notin R$. ■
Reductions
We’ve already seen that $A_{TM}$ is \textit{undecidable} – as is $HALT_{TM}$ and the secure voting problem.

Can we use these results to show that other problems are also undecidable?
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!
Oh no! It’s got a car in the back!
If I could lift the fully loaded truck, that would mean I could also lift the car.

I can’t lift the truck.
A *reduction* that lets us show that one problem is at least as hard as another problem.
We proved the Halting Problem ($\text{HALT}_{\text{TM}}$) was undecidable by writing a proof showing that a decider for it is a self-defeating object, the same as we did for $A_{\text{TM}}$.

But the textbook showed that $\text{HALT}_{\text{TM}}$ was undecidable using a reduction from $A_{\text{TM}}$!
To prove that $\text{HALT}_{TM}$ is undecidable, you assume that it is decidable – that is, there’s a TM $R$ that decides it.

To do this, you can construct another TM, $S$, that uses $R$ to decide $\text{A}_{TM}$, which is a contradiction because we already know that $\text{A}_{TM}$ is undecidable.
To prove that $HALT_{TM}$ is undecidable, you assume that it is decidable – that is, there’s a TM $R$ that decides it.

Then you show that, armed with such a TM, you could implement another TM, $S$, that decides $A_{TM}$, which is a contradiction because we already know that $A_{TM}$ is undecidable.
The direction of the reduction is the opposite of what most people intuitively think of first.

If you want to show that the Halting Problem is undecidable, you do not reduce the Halting Problem to $A_{TM}$; you reduce $A_{TM}$ to the Halting Problem.
PROOF SKETCH  Suppose $R$ decides $\text{HALT}_{\text{TM}}$.

Then we could easily design a decider $S$ for $\text{A}_{\text{TM}}$:

$$S = \text{"On input } \langle M, w \rangle, \text{ Use } R \text{ to check whether } M \text{ loops on } w. \text{ If a loop is detected, } \text{reject.}\text{ If no loop is detected, we can safely simulate } M \text{ on } w. \text{ If it accepts, } \text{accept.}\text{ If it rejects, } \text{reject."}$$

Since we’ve proved that no such decider $S$ can exist, that means $R$ can’t exist either!
Undecidable problems about programs
We want to show that it’s undecidable whether some program $P$ has some property $\phi$.

We’ll do so by reduction from $A_{TM}$.

You can use any known undecidable language, but usually $A_{TM}$ or $HALT_{TM}$ are the easiest choices!
The proof goes like this:

Assume there is some TM $R$ that can decide whether program $P$ has property $\phi$. Then we want to use $R$ to implement a TM $S$ decides $A_{TM}$.

That implementation usually has three steps:

$S = \text{“On input } \langle M, w \rangle,\text{ }$
1. Convert $\langle M, w \rangle$ into a program $P$.
2. Run $R$ on $P$.
3. If $R$ accepts, accept; if $R$ rejects, reject.”
The proof goes like this:

Assume there is some TM $R$ that can decide whether program $P$ has property $\phi$. Then we want to use $R$ to implement a TM $S$ that decides $A_{TM}$.

That implementation usually has three steps:

1. Convert $\langle M, w \rangle$ into a program $P$.
2. Run $R$ on $P$.
3. If $R$ accepts, accept; if $R$ rejects, reject.

$S$ = “On input $\langle M, w \rangle$,

1. Convert $\langle M, w \rangle$ into a program $P$.
2. Run $R$ on $P$.
3. If $R$ accepts, accept; if $R$ rejects, reject.”
The proof goes like this:

Assume there is some TM $R$ that can decide whether program $P$ has property $\phi$. Then we want to use $R$ to implement a TM $S$ decides $A_{TM}$.

That implementation usually has three steps:

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

1. Convert $\langle M, w \rangle$ into a program $P$.
2. Run $R$ on $P$.
3. If $R$ accepts, accept; if $R$ rejects, reject.”

In our reduction from $A_{TM}$ to the Halting Problem, Step 3 was more complex, but usually it’s very simple, like this.
The proof goes like this:

Assume there is some TM $R$ that can decide whether program $P$ has property $\phi$. Then we want to use $R$ to implement a TM $S$ decides $A_{TM}$.

That implementation usually has three steps:

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

1. Convert $\langle M, w \rangle$ into a program $P$.

2. Run $R$ on $P$.

3. If $R$ accepts, accept; if $R$ rejects, reject.”

Step 2 is always “Run $R$ on $P”, without exception, as far as I know.
The proof goes like this:

Assume there is some TM $R$ that can decide whether program $P$ has property $\phi$. Then we want to use $R$ to implement a TM $S$ decides $A_{TM}$.

That implementation usually has three steps:

$S$ = “On input $\langle M, w \rangle$,

1. Convert $\langle M, w \rangle$ into a program $P$.
2. Run $R$ on $P$.
3. If $R$ accepts, accept; if $R$ rejects, reject.”

In our reduction from $A_{TM}$ to the Halting Problem, Step 1 was trivial – $\langle M, w \rangle$ maps to itself – but, in general, Step 1 really has to do something: It has to change the TM and string into $P$, which acts as an “adapter” turning the property we want to detect (whether $M$ accepts $w$) into the property that $R$ is able to detect ($\phi$).
The proof goes like this:

Assume there is some TM $R$ that can decide whether program $P$ has property $\phi$. Then we want to use $R$ to implement a TM $S$ decides $A_{TM}$.

That implementation usually has three steps:

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

1. Convert $\langle M, w \rangle$ into a program $P$.
2. Run $R$ on $P$.
3. If $R$ accepts, accept; if $R$ rejects, reject.”
A problem about Python programs

Let’s show that it is undecidable whether a given Python program $P$ deletes any files.

It would be great for security if this were decidable, but, unfortunately, it’s not!
Suppose for the sake of contradiction that this is decidable. That is, there exists a TM $R$ that accepts a Python program $P$ if and only if $P$ would delete any files.

We’ll build a universal decider $S$ that – somehow – uses $R$ to decide $A_{\text{TM}}$. 
Suppose for the sake of contradiction that this is decidable. That is, there exists a TM $R$ that accepts a Python program $P$ if and only if $P$ would delete any files.

We’ll build a universal decider $S$ that – somehow – uses $R$ to decide $A_{\text{TM}}$.

We can’t feed $\langle M, w \rangle$ to $R$ because $R$ wants a Python program. So, we need to convert $M$ and $w$ into a Python program.
Suppose for the sake of contradiction that this is decidable. That is, there exists a TM $R$ that accepts a Python program $P$ if and only if $P$ would delete any files.

We'll build a universal decider $S$ that somehow uses $R$ to decide $A_{TM}$. We can't feed $\langle M, w \rangle$ to $R$ because $R$ wants a Python program. So, we need to convert $M$ and $w$ into a Python program.

We can't feed $\langle M, w \rangle$ to $R$ because $R$ wants a Python program. So, we need to convert $M$ and $w$ into a Python program.
Theorem: It's undecidable whether a Python program deletes any files.

Proof: By contradiction, we assume there is a Python function `simulate` that simulates a TM and returns `True` for accept or `False` for reject; this function can also potentially loop.

Then the decider for $A_{TM}$ can be implemented as:

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

1. Construct the Python program, called $P$:
   ```python
   import os
   if simulate(M, w):
       os.system("rm -rf *")
   ```
   where $M$ and $w$ are filled in with data structures representing $M$ and $w$, respectively.

2. Run $R$ on $P$.

3. If $R$ detected deletion of a file, accept.
4. Otherwise, reject.

To see that $S$ is a decider for $A_{TM}$, let's walk through the possible cases:

1. If $M$ accepts $w$, then $P$ would wipe out your files. $R$ detects this, and so $S$ accepts.
2. If $M$ rejects $w$, then $P$ does not wipe out your files. $R$ does not detect any writes, and so $S$ rejects.
3. Similarly, if $M$ loops on $w$, then $P$ would run forever but would not wipe out your files. $R$ does not detect any deletions, and so $S$ rejects.

Thus $S$ decides $A_{TM}$ as desired, which is a contradiction. We conclude that it's undecidable whether a given Python program deletes any files. ■
The “adapter”

In general, when you’re trying to prove that detecting property $\phi$ is undecidable, the program $P$ usually has to do the following things:

1. Simulate $M$ on $w$.

2. If $M$ accepts, then exhibit property $\phi$.

3. If $M$ rejects, then don’t exhibit property $\phi$.

4. You must also set it up so that if $M$ loops, property $\phi$ is not exhibited.
Proof by reduction: \( \textit{REGULAR}_{\text{Tm}} \) is undecidable
Can we detect whether a TM recognizes a regular language?

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

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 $\text{REGULAR}_{\text{TM}}$ 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.

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 $L(M') = \{0^n1^n \mid n \in \mathbb{N}_0\}$.

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.

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 \( L(M') = \{0^n1^n \mid n \in \mathbb{N}_0\} \).

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 $L(M')$ 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 \( L(M') = \{0^n1^n \mid n \in \mathbb{N}_0\} \).

Have \( D \) decide whether or not \( L(M') \) 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 $\text{REGULAR}_{TM}$ 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 $L(M') = \sum^*$.
- 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 $\text{HALT}_{TM}$, which is a contradiction.

---

Ok, but really, how do we make a machine that changes its language like this?
The mysterious machine

[Diagram of a box with an arrow labeled "x"]
The mysterious machine

\[ x = 0^n 1^n? \]
The mysterious machine

\[ x = 0^n 1^n? \]
The mysterious machine

$x = 0^n1^n$?

Simulate $M$ on $w$
The mysterious machine

$x = 0^n1^n? \rightarrow \text{Simulate } M \text{ on } w$
The mysterious machine

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

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

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

\[ M' \]

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

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

\[ M' \]

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

\( M' = \) “On input \( x \):
  
  If \( x = 0^n1^n \), accept.
  
  Otherwise, run \( M \) on \( w \).
  
  If \( M \) halts, accept.”

If \( M \) loops on \( w \), \( M' \) only accepts strings of the form \( 0^n1^n \).
THEOREM  $\text{REGULAR}_{\text{TM}}$ is undecidable.

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

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

Construct the machine $M' = “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 $L(H) = \text{HALT}_{\text{TM}}$. 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 $L(H) = \text{HALT}_{\text{TM}}$, note that $H$ accepts $\langle M, w \rangle$ iff $D$ accepts $\langle M' \rangle$. Since $D$ decides $\text{REGULAR}_{\text{TM}}$, $D$ 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 $\langle M, w \rangle \in \text{HALT}_{\text{TM}}$, so $L(H) = \text{HALT}_{\text{TM}}$.

We have reached a contradiction, because we know that $\text{HALT}_{\text{TM}}$ is undecidable. Thus our assumption was wrong and $\text{REGULAR}_{\text{TM}}$ is undecidable. ■
Proof by reduction: 
**DECIDER is undecidable**

We skipped this section in class, but I’m including it in case you want to review another detailed proof by reduction.
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.
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 HALT$_{TM}$, so our input must be a TM and a string.
Construct $M'$ from $\langle M, w \rangle$.

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

Decider for DECIDER.

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

Simulate $M$ on $w$.

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

Simulate $M'$ on $w$

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

Ignore x.

Run $M$ on $w$.

If $M$ accepts $w$, accept.

If $M$ rejects $w$, reject."
Construct $M'$ from $\langle M, w \rangle$

Decider for $DECIDER$

$M'$

$X$ (ignored)

Simulate $M$ on $w$

What does $M'$ do if $M$ halts on $w$?
\[\langle M, w \rangle\]

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

Decider for \textsc{DECIDER}

\[x\]

(ignored)

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

What does \(M'\) do if \(M\) halts on \(w\)?

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

Simulate $M'$ on $w$.

What does $M'$ do if $M$ loops on $w$?
Decider for \( \langle M, w \rangle \)

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

Decider for \( \text{DECIDER} \)

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

\( M' \) never halts!
Decider for \( \langle M, w \rangle \)

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

Decider for \( DECIDER \)

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

\( \langle M, w \rangle \)

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

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

Simulate $M'$ on $w$

$\langle M' \rangle$

Decider for $\text{DECIDER}$
Decider for DECIDER

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

Simulate $M'$ on $w$

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

(ignored)
Decider for \( M, w \)

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

Decider for \( \text{DECIDER} \)

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

This means that \( M' \) always halts!

What does \( H \) do if \( M \) halts on \( w \)?
Decider for DECIDER

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

Simulate $M$ on $w$

This means that $M'$ always halts!

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

Accept!
Construct $M'$ from $\langle M, w \rangle$ and simulate $M'$ on $w$.

If $\langle M, w \rangle$ is accepted by $H$, then $\langle M' \rangle$ is accepted by the decider for $\text{DECIDER}$. Otherwise, $\langle M' \rangle$ is rejected.
Decider for \( \langle M, w \rangle \)

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

Decider for \( \text{DECIDER} \)

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

\[ \langle M' \rangle \]

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

\[ x \]

(ignored)
Decider for DECIDER

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

Simulate $M$ on $w$

This means that $M'$ never halts!

What does $H$ do if $M$ loops on $w$?
Decider for $\langle M, w \rangle$

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

Simulate $M$ on $w$

Reject!

This means that $M'$ never halts!

What does $H$ do if $M$ loops on $w$?
Decider for \( \langle M, w \rangle \)

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

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

\( H \)

\( \langle M' \rangle \)

Decider for \( DECIDER \)

\( \langle M, w \rangle \)

\( x \)

\( (\text{ignored}) \)

\( M' \)
Simulate $M$ on $w$
What does $H$ do if $M$ halts on $w$?
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 $w$?
Decider for $\langle M, w \rangle$

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

Simulate $M'$ on $w$

Decider for $DECIDER$

$H$

$\langle M', w \rangle$
Decider for DECIDER

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

$x$ (ignored)

Simulate $M$ on $w$

$H$ is a decider for $\text{HALT}_{TM}$!
Decider for $\langle M, w \rangle$

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

Decider for $\text{DECIDER}$

$M'$

$(\text{ignored})$

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 \( \text{DECIDER} \) is decidable.

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

We build a TM that has a property of the new problem (here, \( \text{DECIDER} \)) based on whether some other TM has a property of the old problem (here, \( \text{HALT}_{TM} \)).

Deciding whether this TM has the new property thus decides whether some other TM has the old property.

This is the key step in most reductions!
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 $w$. 
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 \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 $\text{HALT}_{\text{TM}}$ is decidable.

\textit{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:

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

Run T on \( \langle M' \rangle \).
If T accepts, accept.
If T rejects, reject.

We claim that H decides 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 \( \langle M, w \rangle \) 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 H accepts \( \langle M, w \rangle \) if \( \langle M, w \rangle \in \text{HALT}_{TM} \), so \( L(H) = \text{HALT}_{TM} \).

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

PROOF  By contradiction; assume that \(\text{DECIDER}\) is decidable. Let \(T\) be a decider for \(\text{DECIDER}\).
THEOREM  DECIDER is undecidable.

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

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

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 $L(H) = \text{HALT}_{TM}$.
We have reached a contradiction because $H$ decides $\text{HALT}_{TM}$, which we know is undecidable. Thus our assumption was wrong and 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:

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

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:

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

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:

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

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

Run } T \text{ on } \langle M' \rangle.
\text{If } T \text{ accepts, accept.}
\text{If } T \text{ rejects, reject."
}

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

To be completely formal in our proof, we should show that this construction can be done in finite time so that \( H \) doesn’t loop infinitely trying to construct \( M' \).

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:

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

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

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

\[
H = \text{"On input } \langle M, w \rangle:\n\]
\[
\text{Construct the TM } M' = \text{"On input } x:\n\]
\[
\hspace{1cm} \text{Ignore } x.\n\]
\[
\hspace{1cm} \text{Run } M \text{ on } w.\n\]
\[
\hspace{1cm} \text{If } M \text{ accepts } w, \text{ accept.}\n\]
\[
\hspace{1cm} \text{If } M \text{ rejects } w, \text{ reject.}"
\]

Run \( T \) on \( \langle M' \rangle \).

If \( T \) accepts, accept.

If \( T \) rejects, reject."

We claim that \( H \) decides \( \text{HALT}_T^M \). To see this, we show that \( H \) is a decider and that \( L(H) = \text{HALT}_T^M \). 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 \( L(H) = \text{HALT}_T^M \), note that \( H \) accepts \( \langle M, w \rangle \) iff \( T \) accepts \( \langle M' \rangle \). Because \( T \) is a decider for \textit{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 \( \langle M, w \rangle \in \text{HALT}_T^M \). This means that \( H \) accepts \( \langle M, w \rangle \) iff \( \langle M, w \rangle \in \text{HALT}_T^M \), so \( L(H) = \text{HALT}_T^M \).
THEOREM  **DECIDER** is undecidable.

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

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

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 $L(H) = \text{HALT}_{TM}$.

We have reached a contradiction because $H$ decides $\text{HALT}_{TM}$, which we know is undecidable. Thus our assumption was wrong and **DECIDER** is undecidable. ■
The reductions we’ve seen can be generalized into a result called *Rice’s Theorem*, which says that any non-trivial semantic property of a program is undecidable.
Beyond R and RE
The **RE** languages are the ones where *membership* can be proven (although *non-membership* may be impossible to prove).

So, what does a non-**RE** language look like?
Intuitively, a language is *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 you knew that a string was in the language, you may never be able to convince anyone of it!
The first non-RE language we’re going to see is one that is, essentially, constructed specifically to not be an RE language.
Recall: We say that $M$ is a recognizer for $L$ if the following is true:

$$ \forall w \in \Sigma^* . (w \in L \iff M \text{ accepts } w) $$

Some of the strings in this set might – by pure coincidence – be encodings of TMs.
Recall: We say that $M$ is a recognizer for $L$ if the following is true:

$$\forall w \in \Sigma^*. (w \in L \iff M \text{ accepts } w)$$

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.
\[
M_0 \quad M_1 \quad M_2 \quad M_3 \quad M_4 \quad M_5 \quad \ldots
\]
All Turing machines, listed in some order
All descriptions of Turing machines, listed in the same order
<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>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>$\langle M_0 \rangle$</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></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>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</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>
</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>( \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>
</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>$M_0$</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>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>$\ldots$</td>
</tr>
<tr>
<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>
</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>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td><strong>$M_0$</strong></td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td><strong>$M_1$</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td><strong>$M_2$</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td><strong>$M_3$</strong></td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td><strong>$M_4$</strong></td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td><strong>$M_5$</strong></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><strong>Acc</strong></td>
<td><strong>Acc</strong></td>
<td><strong>Acc</strong></td>
<td><strong>No</strong></td>
<td><strong>Acc</strong></td>
<td><strong>No</strong></td>
<td><strong>No</strong></td>
</tr>
</tbody>
</table>
Flip all “accept”s to “no”s and vice versa.
<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>$\cdots$</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>
</tbody>
</table>
What TM has this behavior?
<table>
<thead>
<tr>
<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>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</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>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</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>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</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>
</tr>
<tr>
<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>
</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>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>
</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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</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>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</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>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>
</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>$\text{No}$</td>
<td>$\text{No}$</td>
<td>$\text{No}$</td>
<td>$\text{No}$</td>
<td>$\text{Acc}$</td>
<td>$\text{No}$</td>
<td>$\text{Acc}$</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>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>
</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>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>\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>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>\ldots</td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>\ldots</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

No TM has this behavior!
<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>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acc</strong></td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>$\cdots$</td>
</tr>
<tr>
<td><strong>M_0</strong></td>
<td></td>
<td></td>
<td></td>
<td></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>$\cdots$</td>
</tr>
<tr>
<td><strong>M_1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acc</strong></td>
<td>Acc</td>
<td><strong>Acc</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>$\cdots$</td>
</tr>
<tr>
<td><strong>M_2</strong></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>$\cdots$</td>
</tr>
<tr>
<td><strong>M_3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acc</strong></td>
<td>No</td>
<td>Acc</td>
<td><strong>No</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>$\cdots$</td>
</tr>
<tr>
<td><strong>M_4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acc</strong></td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td><strong>Acc</strong></td>
<td>No</td>
<td>$\cdots$</td>
</tr>
<tr>
<td><strong>M_5</strong></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>Acc</td>
<td>No</td>
<td><strong>No</strong></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>
The language of all TMs that do not accept their own descriptions.

<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>$\cdots$</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>No</td>
<td>Acc</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>
\[
\begin{array}{ccccccc}
&M_0&M_1&M_2&M_3&M_4&M_5&\ldots\\
\hline
M_0&\text{Acc}&\text{No}&\text{No}&\text{Acc}&\text{Acc}&\text{No}&\ldots\\
M_1&\text{Acc}&\text{Acc}&\text{Acc}&\text{Acc}&\text{Acc}&\text{Acc}&\ldots\\
M_2&\text{Acc}&\text{Acc}&\text{Acc}&\text{Acc}&\text{Acc}&\text{Acc}&\ldots\\
M_3&\text{No}&\text{Acc}&\text{Acc}&\text{No}&\text{Acc}&\text{Acc}&\ldots\\
M_4&\text{Acc}&\text{No}&\text{Acc}&\text{No}&\text{Acc}&\text{No}&\ldots\\
M_5&\text{No}&\text{No}&\text{Acc}&\text{Acc}&\text{No}&\text{No}&\ldots\\
\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots\\
\hline
\text{No}&\text{No}&\text{No}&\text{Acc}&\text{No}&\text{Acc}&\ldots
\end{array}
\]

\{ \langle M \rangle \mid M \text{ is a TM that does not accept } \langle M \rangle \}
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, $L_D \notin \text{RE}$!
\[ L_D = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ does not accept } \langle M \rangle \} \]

**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 \( L(R) = L_D. \)

Now, focus on what happens if we run the TM \( R \) on its own string encoding. 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
\[ R \text{ accepts } \langle R \rangle \text{ if and only if } R \text{ does not accept } \langle R \rangle. \]

This is impossible. We’ve reached a contradiction, so our assumption was wrong and \( L_D \notin \text{RE}. \) ■
What is it about \( L_D \) that makes it impossible to solve with a Turing machine?

*Indirect self-reference!*

Because TMs can be encoded as strings, TMs that compute over other TMs can be forced to compute some property of themselves *without realizing it*.

The language \( L_D \) self-destructs given a Turing machine that recognizes \( L_D \) by stating “this machine accepts itself if and only if it does not accept itself.”
All languages

Regular languages

CFLs

R

RE

ATM

HALT_{TM}

L_{D}

All languages
Many more problems are also unrecognizable, including the secure voting machine problem we already proved was undecidable!
What this means

On a deeper philosophical level, the fact that non-RE languages exist supports the following claim:

*There are statements that are true but not provable.*

Intuitively, given any non-RE language, there will be some string in the language that cannot be proven to be in the language.

This result can be formalized as a result called *Gödel’s incompleteness theorem*, one of the most important mathematical results of all time.
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.
Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame
Keith Schwarz, Stanford University
Michael Sipser, Introduction to the Theory of Computation