

# BU CS 332 – Theory of Computation

<https://forms.gle/EyQvoGQwPcCZcxuF7>



## Lecture 14:

- More on Undecidable and Unrecognizable Languages
- Reductions

Reading:

Sipser Ch 4.2, 5.1

Alexander Poremba & Mark Bun

March 24, 2026

# Undecidability

**Last time:** Countability, uncountability, and diagonalization

Existential proof that there are undecidable  
and unrecognizable languages

An explicit undecidable language

**Today:** Reductions: Relate decidability / undecidability  
of different problems

# Specializing the proof

**Theorem:** Let  $X$  be the set of all TM deciders. Then there exists an undecidable language in  $P(\{0, 1\}^*)$

- 1) Consider the function  $L: X \rightarrow P(\{0, 1\}^*)$
- 2) “Flip the diagonal” to construct a language  $UD \in P(\{0, 1\}^*)$  such that  $L(M) \neq UD$  for every  $M \in X$
- 3) Conclude that  $UD$  is undecidable, hence  $L$  is not onto

# An explicit undecidable language

Does  $M_2$  accept on input  $\langle M_3 \rangle$ ?

TM $M$	$M(\langle M_1 \rangle)$ ?	$M(\langle M_2 \rangle)$ ?	$M(\langle M_3 \rangle)$ ?	$M(\langle M_4 \rangle)$ ?		$D(\langle D \rangle)$ ?
$M_1$	<del>Y</del> N	N	Y	Y	...	
$M_2$	N	<del>N</del> Y	Y	Y		
$M_3$	Y	Y	<del>Y</del> N	N		
$M_4$	N	N	Y	<del>N</del> Y		
$\vdots$					$\ddots$	
$D$						

$UD = \{ \langle M \rangle \mid M \text{ is a TM that does not accept on input } \langle M \rangle \}$

**Claim:**  $UD$  is undecidable

# An explicit undecidable language

**Theorem:**  $UD = \{\langle M \rangle \mid M \text{ is a TM that does not accept on input } \langle M \rangle\}$  is undecidable

**Proof:** Suppose for contradiction that some TM  $D$  decides  $UD$

By definition of deciders:  $\forall \langle M \rangle$  (a string encoding TM  $M$ )

- $\langle M \rangle \in UD \Rightarrow D$  accepts input  $\langle M \rangle$
- $\langle M \rangle \notin UD \Rightarrow D$  rejects input  $\langle M \rangle$

Imagine running  $D$  on input  $\langle D \rangle$ . Two cases:

1)  $D$  accepts input  $\langle D \rangle \Rightarrow \langle D \rangle \notin UD$  by def of  $UD$   
But this contradicts ~~\*~~

2)  $D$  rejects input  $\langle D \rangle \Rightarrow \langle D \rangle \in UD$  by def of  $UD$   
But this contradicts ~~\*~~

In either case,  $D$  has the wrong behavior on input  $\langle D \rangle$

$\Rightarrow D$  does not decide  $UD$

# A more useful undecidable language

$A_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$

**Theorem:**  $A_{\text{TM}}$  is undecidable

**Proof:** Assume for the sake of contradiction that TM  $H$  decides  $A_{\text{TM}}$ :

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

**Idea:** Show that  $H$  can be used to construct a decider for the (undecidable) language  $UD$  – a contradiction.

# A more useful undecidable language

$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$

**Proof (continued):**

Suppose, for contradiction, that  $H$  decides  $A_{TM}$

Consider the following TM  $D$ :

“On input  $\langle M \rangle$  where  $M$  is a TM:

1. Run  $H$  on input  $\langle M, \langle M \rangle \rangle$
2. If  $H$  accepts, **reject**. If  $H$  rejects, **accept**.”

**Claim:**  $D$  decides  $UD = \{\langle M \rangle \mid \text{TM } M \text{ does not accept } \langle M \rangle\}$

Proof: If  $\langle M \rangle \in UD \Rightarrow M$  does not accept input  $\langle M \rangle$  (def of  $UD$ )  
 $\Rightarrow \langle M, \langle M \rangle \rangle \notin A_{TM}$  (def of  $A_{TM}$ )  
 $\Rightarrow H$  rejects  $\Rightarrow D$  accepts ✓

If  $\langle M \rangle \notin UD \Rightarrow M$  accepts input  $\langle M \rangle$  (def of  $UD$ )  
 $\Rightarrow \langle M, \langle M \rangle \rangle \in A_{TM}$   
 $\Rightarrow H$  accepts  $\Rightarrow D$  rejects ✓

$\langle M, w \rangle \in A_{TM}$   
 $\Rightarrow H(\langle M, w \rangle)$  accepts

$\langle M, w \rangle \notin A_{TM}$   
 $\Rightarrow H(\langle M, w \rangle)$  rejects

...but this language is undecidable!

# Unrecognizable Languages

**Theorem:** A language  $L$  is decidable if and only if  $L$  and  $\bar{L}$  are both Turing-recognizable.

**Corollary:**  $\overline{A_{TM}}$  is unrecognizable

Proof:

- $A_{TM}$  is Turing-recognizable
- $A_{TM}$  is undecidable

If  $\overline{A_{TM}}$  were recognizable, then  $TM \Rightarrow A_{TM}$  is decidable  $\times$

**Proof of Theorem:**

$\Rightarrow$  | WTS: If  $L$  is decidable, then  $L$  and  $\bar{L}$  are Turing-recognizable

If  $L$  decidable  $\Rightarrow L$  is Turing-recognizable

$\hookrightarrow \bar{L}$  is decidable  $\Rightarrow \bar{L}$  is Turing-recognizable

# Unrecognizable Languages

**Theorem:** A language  $L$  is decidable if and only if  $L$  and  $\bar{L}$  are both Turing-recognizable.

**Proof continued:**

$\Leftarrow$  WTS: If  $L$  and  $\bar{L}$  both Turing-recognizable, then  $L$  is decidable

$L$  Turing-recognizable  $\Rightarrow \exists$  a TM  $M_1$  recognizing  $L$   
 $\bar{L}$  " "  $\Rightarrow$  "  $M_2$  recognizing  $\bar{L}$

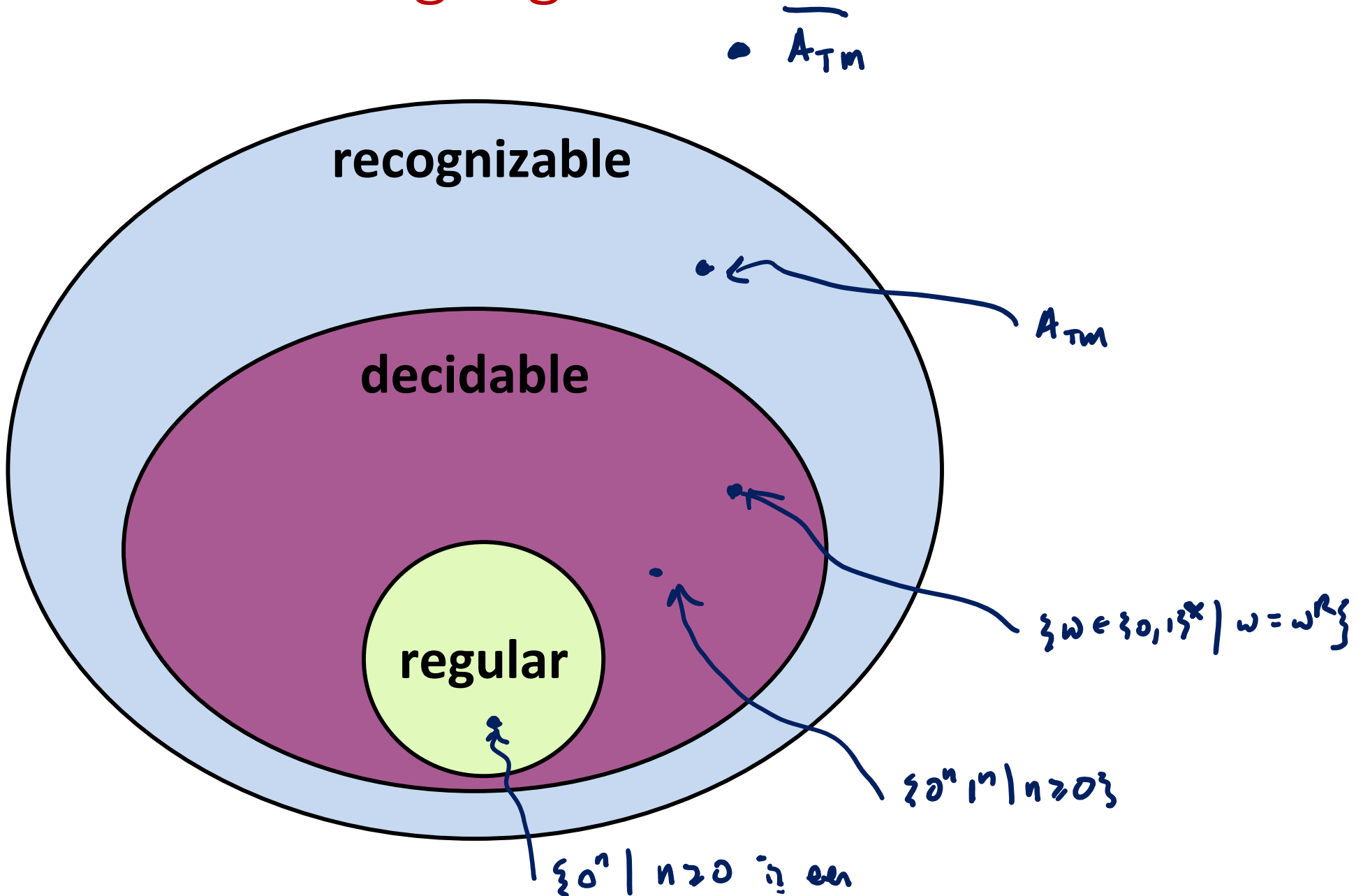
The following TM  $N$  decides  $L$ :

On input  $x$ :

Repeat until termination:

- 1) Run  $M_1$  for next step on input  $x$
- 2) Run  $M_2$  for next step on input  $x$
- 3) If  $M_1$  accepts, accept. If  $M_2$  accepts, reject.

# Classes of Languages



# Reductions

# Scientists vs. Engineers

A computer scientist and an engineer are stranded on a desert island. They find two palm trees with one coconut on each. The engineer climbs a tree, picks a coconut and eats.



The computer scientist climbs the second tree, picks a coconut, climbs down, climbs up the first tree and places it there, declaring success.

“Now we’ve reduced the problem to one we’ve already solved.”  
(Please laugh)

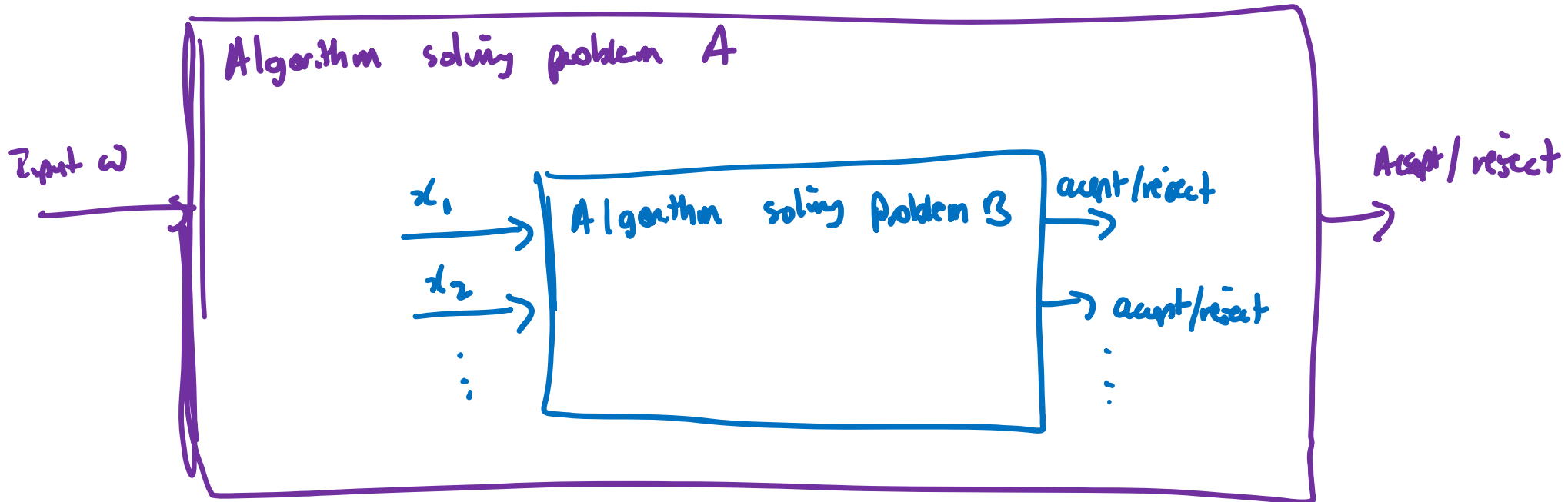
# Reductions

Eating a coconut from Tree 2

Eating a coconut from Tree 1

A **reduction** from problem  $A$  to problem  $B$  is an algorithm solving problem  $A$  which uses an algorithm solving problem  $B$  as a subroutine

If such a reduction exists, we say “ $A$  reduces to  $B$ ”



# Two uses of reductions

**Positive uses:** If  $A$  reduces to  $B$  and  $B$  is decidable, then  $A$  is also decidable

$$E_{DFA} = \{ \langle D \rangle \mid D \text{ is a DFA s.t. } L(D) = \emptyset \}$$

$$EQ_{DFA} = \{ \langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2) \}$$

**Theorem:**  $EQ_{DFA}$  is decidable

**Proof:** The following TM decides  $EQ_{DFA}$

On input  $\langle D_1, D_2 \rangle$ , where  $\{D_1, D_2\}$  are DFAs:

1. Construct a DFA  $D$  that recognizes the symmetric difference  $L(D_1) \Delta L(D_2)$
2. Run the decider for  $E_{DFA}$  on  $\langle D \rangle$  and return its output

# Two uses of reductions

**Negative uses:** If  $A$  reduces to  $B$  and  $A$  is undecidable, then  $B$  is also undecidable

$UD = \{ \langle M \rangle \mid \text{TM } M \text{ does not accept input } \langle M \rangle \}$

$A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \}$

Suppose  $H$  decides  $A_{\text{TM}}$

Consider the following TM  $D$ .

On input  $\langle M \rangle$  where  $M$  is a TM:

1. Run  $H$  on input  $\langle M, \langle M \rangle \rangle$
2. If  $H$  accepts, **reject**. If  $H$  rejects, **accept**.

**Claim:** If  $H$  decides  $A_{\text{TM}}$  then  $D$  decides

$UD = \{ \langle M \rangle \mid M \text{ is a TM that does not accept input } \langle M \rangle \}$

# Two uses of reductions

**Negative uses:** If  $A$  reduces to  $B$  and  $A$  is undecidable, then  $B$  is also undecidable

## Template for undecidability proof by reduction:

1. Suppose to the contrary that  $B$  is decidable
2. Using a decider for  $B$  as a subroutine, construct an algorithm deciding  $A$
3. But  $A$  is undecidable. Contradiction!

# Halting Problem

**Computational problem:** Given a program (TM) and input  $w$ , does that program halt (either accept or reject) on input  $w$ ?

**Formulation as a language:**

$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$$

**Ex.**  $M =$  “On input  $x$  (a natural number written in binary):

For each  $y = 1, 2, 3, \dots$ :

*binary encoding of* If  $y^2 = x$ , **accept**. Else, continue.”

Is  $\langle M, 101 \rangle \in HALT_{TM}$ ?

- a) Yes, because  $M$  accepts on input 101
- b) Yes, because  $M$  rejects on input 101
- c) No, because  $M$  rejects on input 101
- d) No, because  $M$  loops on input 101



# Halting Problem

**Computational problem:** Given a program (TM) and input  $w$ , does that program halt (either accept or reject) on input  $w$ ?

**Formulation as a language:**

$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$$

**Ex.**  $M =$  “On input  $x$  (a natural number in binary):

For each  $y = 1, 2, 3, \dots$ :

If  $y^2 = x$ , **accept**. Else, continue.”

$\langle M, 1017 \rangle \notin HALT_{TM}$

$M' =$  “On input  $x$  (a natural number in binary):

For each  $y = 1, 2, 3, \dots, x$ :

If  $y^2 = x$ , **accept**. Else, continue.

**Reject.**”

$\langle M', 1017 \rangle \in HALT_{TM}$

# Halting Problem

$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$

**Theorem:**  $HALT_{TM}$  is undecidable

**Proof:** Suppose for contradiction that there exists a decider  $H$  for  $HALT_{TM}$ . We construct a decider  $V$  for  $A_{TM}$  as follows:

On input  $\langle M, w \rangle$ : # Instance of  $A_{TM}$  problem

1. Run  $H$  on input  $\langle M, w \rangle$
2. If  $H$  rejects, **reject**
3. If  $H$  accepts, run  $M$  on  $w$
4. If  $M$  accepts, **accept**  
Otherwise, **reject**.

Proof that  $V$  decides  $A_{TM}$ :

1)  $\langle M, w \rangle \in A_{TM} \Rightarrow M$  accepts  $w$   
 $\Rightarrow H$  accepts input  $\langle M, w \rangle$   
 $\Rightarrow$  Go to step 3, where we run  $M$  on  $w$  and it accepts  $\Rightarrow V$  accepts ✓

2)  $\langle M, w \rangle \notin A_{TM}$

a)  $M$  rejects  $w$   
 $\Rightarrow H$  accepts input  $\langle M, w \rangle$   
 $\Rightarrow$  Go to step 3, where we run  $M$  on  $w$  and it rejects  $\Rightarrow V$  rejects ✓

b)  $M$  loops forever on  $w$   
 $\Rightarrow H$  rejects input  $\langle M, w \rangle$   
 $\Rightarrow V$  rejects in step 2 ✓

Use  $H$  to test whether  $M$  will halt on  $w$

Once we know  $M$  will halt on  $w$ ,  
run  $M$  on  $w$

This is a reduction from  $A_{TM}$  to  $HALT_{TM}$

# Halting Problem

$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$

**Theorem:**  $HALT_{TM}$  is undecidable

**Proof:** Suppose for contradiction that there exists a decider  $H$  for  $HALT_{TM}$ . We construct a decider for  $V$  for  $A_{TM}$  as follows:

On input  $\langle M, w \rangle$ :

1. Run  $H$  on input  $\langle M, w \rangle$
2. If  $H$  rejects, **reject**
3. If  $H$  accepts, run  $M$  on  $w$
4. If  $M$  accepts, **accept**  
Otherwise, **reject**.

↓

This is a reduction from  $A_{TM}$  to  $HALT_{TM}$

# Halting Problem

**Computational problem:** Given a program (TM) and input  $w$ , does that program halt on input  $w$ ?

- A central problem in formal verification
- Dealing with undecidability in practice:
  - Use heuristics that are correct on most real instances, but may be wrong or loop forever on others
  - Restrict to a “non-Turing-complete” subclass of programs for which halting is decidable
  - Use a programming language that lets a programmer specify hints (e.g., loop invariants) that can be compiled into a formal proof of halting

# Emptiness testing for TMs

Given a TM  $M$ , does  $M$  recognize the empty language?

$$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

**Theorem:**  $E_{TM}$  is undecidable

First attempt  
(Doesn't work -- will repair it Thursday)

**Proof:** Suppose for contradiction that there exists a decider  $R$  for  $E_{TM}$ . We construct a decider  $V$  for  $A_{TM}$  as follows:

On input  $\langle M, w \rangle$ : # Instance of  $A_{TM}$

1. Run  $R$  on input ???  $\langle M \rangle$

:

2. If  $R$  accepts, reject  
If  $R$  rejects, accept.

Analysis:

- $\langle M, w \rangle \in A_{TM}$   
 $\Rightarrow M$  accepts  $w$   
 $\Rightarrow L(M) \neq \emptyset$   
 $\Rightarrow \langle M \rangle \notin E_{TM}$   
 $\Rightarrow R$  rejects  $\Rightarrow V$  accepts
- $\langle M, w \rangle \notin A_{TM}$   
 $\Rightarrow M$  does not accept  $w$   
 $\Rightarrow L(M)$  may or may not be empty

This is a reduction from  $A_{TM}$  to  $E_{TM}$   
so  $R$ 's answer is useless