

# Proving Undecidability (1)

Recall the language

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

**Proof that  $A_{\text{TM}}$  is not TM-decidable (Thm. 4.11)**

(Contradiction) Assume that TM  $G$  decides  $A_{\text{TM}}$ :

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

From  $G$  we construct a new TM  $D$  that will get us into trouble...

# Proving Undecidability (2)

The TM D works as follows on input  $\langle M \rangle$  (a TM):

1) Run G on  $\langle M, \langle M \rangle \rangle$

2) Disagree with the answer of G

(The TM D always halts because G always halts.)

In short:  $D\langle M \rangle = \begin{cases} \text{"accept"} & \text{if G rejects } \langle M, \langle M \rangle \rangle \\ \text{"reject"} & \text{if G accepts } \langle M, \langle M \rangle \rangle \end{cases}$

Hence:  $D\langle M \rangle = \begin{cases} \text{"accept"} & \text{if M does not accept } \langle M \rangle \\ \text{"reject"} & \text{if M does accept } \langle M \rangle \end{cases}$

Now run D on  $\langle D \rangle$  (“on itself”)...

# Proving Undecidability (3)




Result:  $D\langle D \rangle = \begin{cases} \text{"accept"} & \text{if } D \text{ does not accept } \langle D \rangle \\ \text{"reject"} & \text{if } D \text{ does accept } \langle D \rangle \end{cases}$

This does not make sense:  $D$  only accepts if it rejects, and vice versa.  
(Note again that  $D$  always halts.)

Contradiction:  **$A_{TM}$  is not TM-decidable.**

This proof used diagonalization implicitly...




# Review of Proof (1)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	☹
$M_1$	accept		accept		
$M_2$	accept	accept	accept	accept	
$M_3$					☹
$M_4$	accept	accept			
					

‘Acceptance behavior’ of  $M_i$  on  $\langle M_j \rangle$

# Review of Proof (2)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	☹
$M_1$	accept	reject	accept	reject	
$M_2$	accept	accept	accept	accept	
$M_3$	reject	reject	reject	reject	☹
$M_4$	accept	accept	reject	reject	









‘Deciding behavior’ of  $G$  on  $\langle M_i, \langle M_j \rangle \rangle$

# Review of Proof (3)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	☹	$\langle D \rangle$	☹
$M_1$	accept	reject	accept	reject			
$M_2$	accept	accept	accept	accept			
$M_3$	reject	reject	reject	reject	☹		
$M_4$	accept	accept	reject	reject			

# Review of Proof (4)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	☹	$\langle D \rangle$	☹
$M_1$	accept	reject	accept	reject			
$M_2$	accept	accept	accept	accept			
$M_3$	reject	reject	reject	reject	☹		
$M_4$	accept	accept	reject	reject			
							
$D$	reject	reject	accept	accept	☹	?	
							

Contradiction for  $D$  on input  $\langle D \rangle$ .

# Another View of the Problem

The “**Self-referential paradox**” occurs when we force the TM  $D$  to disagree with itself.

On the one hand,  $D$  knows what it is going to do on input  $\langle D \rangle$ , but then it decides to do something else instead.

*“You cannot know for sure what you will do in the future, because then you could decide to change your actions and create a paradox.”*



# Self-Reference in Math

The diagonalization method implements the self-reference paradox in a mathematical way.

In logic this approach is often used to prove that certain things are impossible.

Kurt Gödel gave a mathematical equivalent of “This sentence is not true.”

Old puzzle: In a town, there is a barber who shaves all those who do not shave themselves.

Who shaves the barber ?

# Self-Reference in CSE

What happens if a computer program  $M$  tries to answer questions about itself  $\langle M \rangle$ ?

Sometimes this is perfectly okay:

- How big is  $\langle M \rangle$ ?
- Is  $\langle M \rangle$  a proper TM?

Other questions lead to paradoxes:

- Does  $\langle M \rangle$  halt or not?
- Is there a smaller program  $M'$  that is equivalent?

# TM-Unrecognizable

$A_{TM}$  is not TM-decidable, but it is TM-recognizable.  
What about a language that is not recognizable?

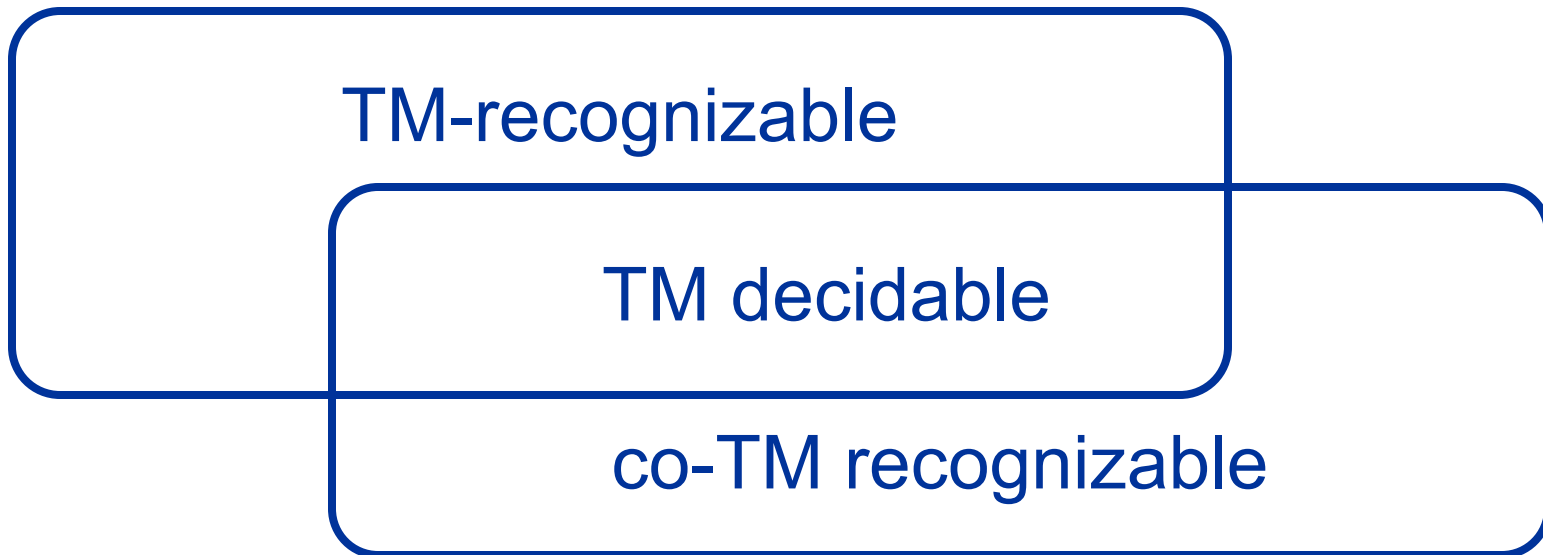
**Theorem 4.22:** If a language  $A$  is recognizable and its complement  $\bar{A}$  is recognizable, then  $A$  is Turing machine decidable.

**Proof:** Run the recognizing TMs for  $A$  and  $\bar{A}$  in parallel on input  $x$ . Wait for one of the TMs to accept. If the TM for  $A$  accepted: “accept  $x$ ”; if the TM for  $\bar{A}$  accepted: “reject  $x$ ”.

# $\bar{A}_{TM}$ is not TM-Recognizable

By the previous theorem it follows that  $\bar{A}_{TM}$  cannot be TM-recognizable, because this would imply that  $A_{TM}$  is TM decidable (Corollary 4.23).

We call languages like  $\bar{A}_{TM}$  co-TM recognizable.



# Things that TMs Cannot Do:

The following languages are also unrecognizable:

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

$$EQ_{\text{TM}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are TMs} \\ \text{with } L(G) = L(H) \}$$

To be precise:

- $E_{\text{TM}}$  is co-TM recognizable
- $EQ_{\text{TM}}$  is not even co-Turing recognizable

How can we prove these facts?

# Next: reducibility

- We still need to ***prove*** that the Halting problem is undecidable.
- Do more examples of undecidable problems.
- Try to get a general technique for proving undecidability.

# Halting problem

- Assume that it is decidable. So there is a TM  $S$  that decides

$\text{HALT} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$

- Use  $S$  as a **subroutine** to get a TM  $S$  to decide

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

- Therefore  $A_{\text{TM}}$  is decidable. CONTRADICTION!
- Details follow ....

# Halting problem - 2

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

- Run TM  $R$  on input  $\langle M, w \rangle$ .
- If  $R$  rejects, REJECT.
- If  $R$  accepts, simulate  $M$  on  $w$  until it halts.
- If  $M$  has accepted, ACCEPT, else REJECT"



# More undecidability

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

We mentioned that  $E_{TM}$  is co-TM recognizable.

We will prove next that  $E_{TM}$  is undecidable.

Intuition: You cannot solve this problem UNLESS you solve the halting problem!!

But this is hard to formalize, so we use  $A_{TM}$ .  
Instead.

# $E_{TM}$ is undecidable

Assume  $R$  decides  $E_{TM}$ . Use  $R$  to design TM  $S$  to decide  $A_{TM}$ .

- Given a TM  $M$  and input  $w$ , define a new TM  $M'$ :
  - If  $x \neq w$ , reject
  - If  $x = w$ , accept iff  $M$  accepts  $w$

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

- Construct  $M'$  as above.
- Run TM  $R$  on input  $\langle M' \rangle$ .
- If  $R$  accepts, REJECT; If  $R$  rejects, ACCEPT.”

# $EQ_{TM}$ is undecidable

If this is decidable, then we can solve  $E_{TM}$ !! (You need to check equality with TM  $M_1$  that rejects all inputs)

Assume  $R$  decides  $EQ_{TM}$ . Use  $R$  to design TM  $S$  to decide  $E_{TM}$ .

$S =$  “On input  $\langle M \rangle$

- Run TM  $R$  on input  $\langle M, M_1 \rangle$ .
- If  $R$  accepts, ACCEPT; If  $R$  rejects, REJECT.”

# The running idea

All our proofs had a common structure

- The first undecidable proof was hard – used diagonalization/self-reference.
- For the rest, we assumed decidable and used it as a subroutine to design TM's that decide known undecidable problems.
- Can we make this technique more structured?

# Mapping Reducibility

Thus far, we used reductions informally:

If “knowing how to solve A” implied “knowing how to solve B”, then we had a **reduction** from B to A.

Sometimes we had to negate the answer to the “ $\in A$ ?” question, sometimes not. In general, it was unspecified which transformations were allowed around the “ $\in A$ ?”-part of the reduction.

Let us make this formal...

# Computable Functions

A function  $f:\Sigma^*\rightarrow\Sigma^*$  is a TM-computable function if there is a Turing machine that on every input  $w\in\Sigma^*$  halts with just  $f(w)$  on the tape.

All the usual computations (addition, multiplication, sorting, minimization, etc.) are all TM-computable.

Note: alterations to TMs, like “given a TM  $M$ , we can make an  $M'$  such that...” can also be described by computable functions that satisfy  $f(\langle M \rangle) = \langle M' \rangle$ .