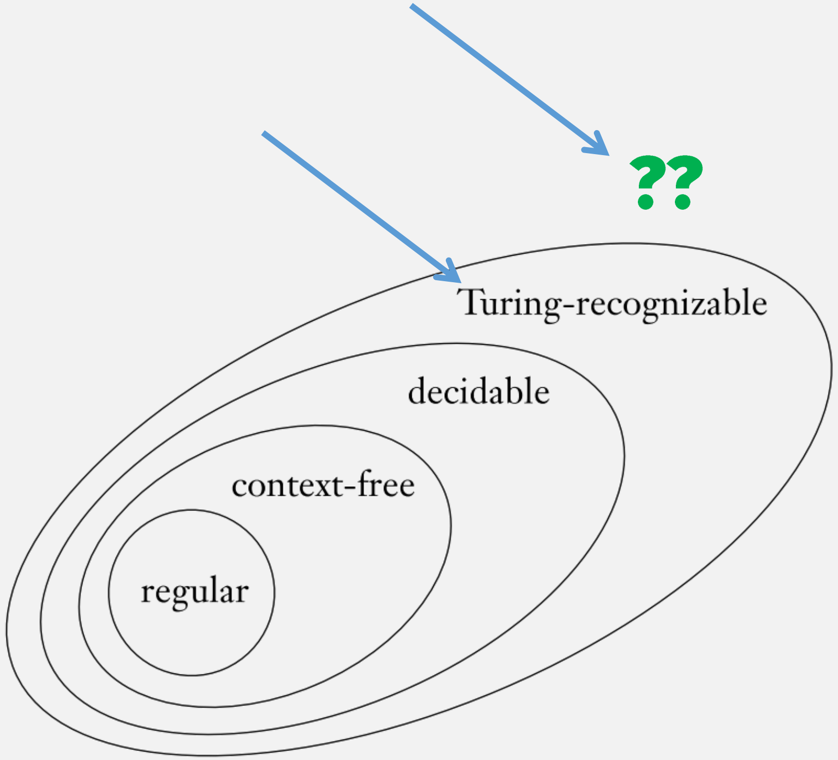
UMB CS 420

Undecidability

Monday, April 10, 2023



Announcements

- HW 8 out
  - due Tuesday 4/11, 11:59pm EST

- No lecture next Monday 4/17
  - Patriot's (Marathon) Day

Quiz Preview

- Is the Universal Turing Machine (A_{TM}) a decider?

Language of DFA description + string pairs, i.e., compute whether a DFA accepts a string

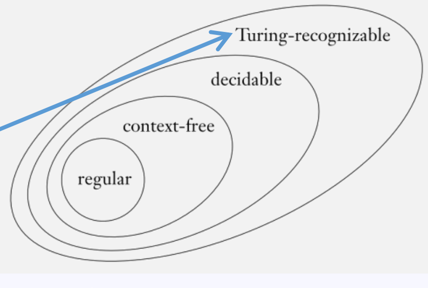
Recap: Decidability of Regular and CFLs

- $A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ Decidable
- $A_{\mathsf{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$ Decidable
- $A_{\mathsf{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$ Decidable
- $E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ Decidable
- $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ Decidable
- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ Decidable
- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ Decidable
- $EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ *Undecidable?*
- $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ *Undecidable?*

Compute something about DFA language from its description

compute whether a TM accepts a string

Thm: A_{TM} is Turing-recognizable



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

U = "On input $\langle M, w\rangle$, where M is a TM and w is a string:

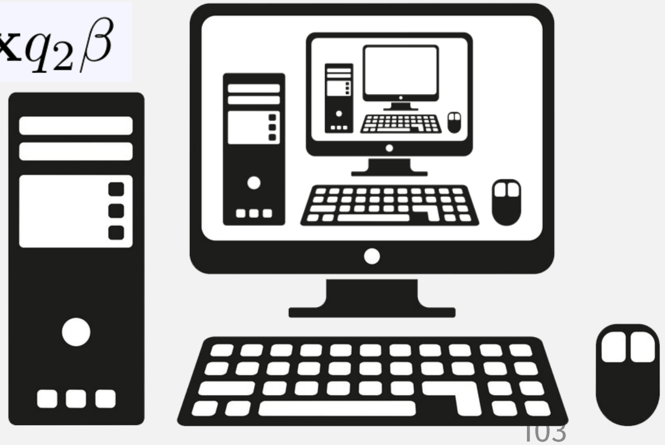
1. Simulate M on input w. ← Can go into infinite loop, causing U to loop
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

U = Implements TM computation steps $\alpha q_1 \mathbf{a}\beta \vdash \alpha \mathbf{x} q_2 \beta$

- i.e., "The Universal Turing Machine"
- "Program" simulating other programs (**interpreter**)
- Problem: U loops when M loops

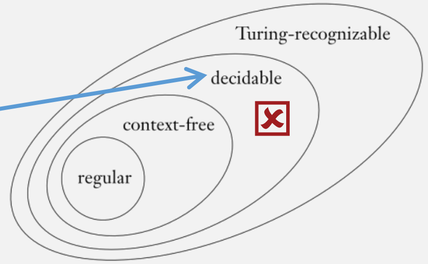
Termination argument?

So it's a **recognizer**, not a decider



Thm: A_{TM} is undecidable

How to prove … not in here?



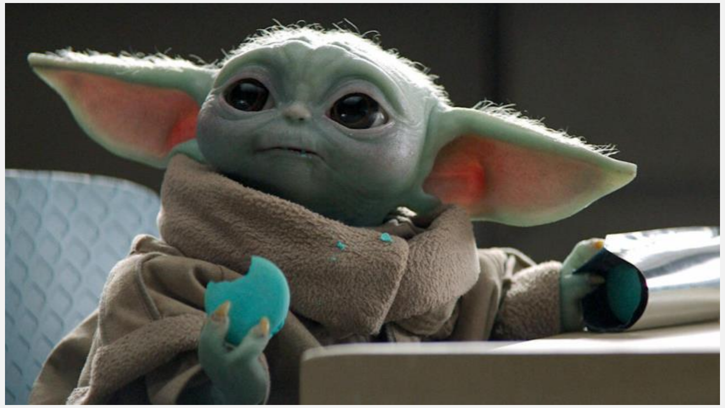
$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

- ???

Flashback: Prove Aliens Do Not Exist

In general, **proving something <u>not true</u> is different** (and often harder) **than proving it <u>true</u>**

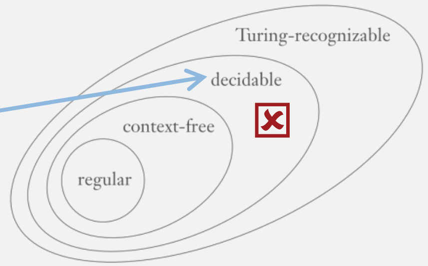
In some cases, **it's possible,** but **typically requires new proof techniques!**



<u>Example</u> (**Regular** Languages)
Prove **a language is regular:**
- Create a DFA

Prove **a language is not regular:**
- Proof by contradiction using **Pumping Lemma**

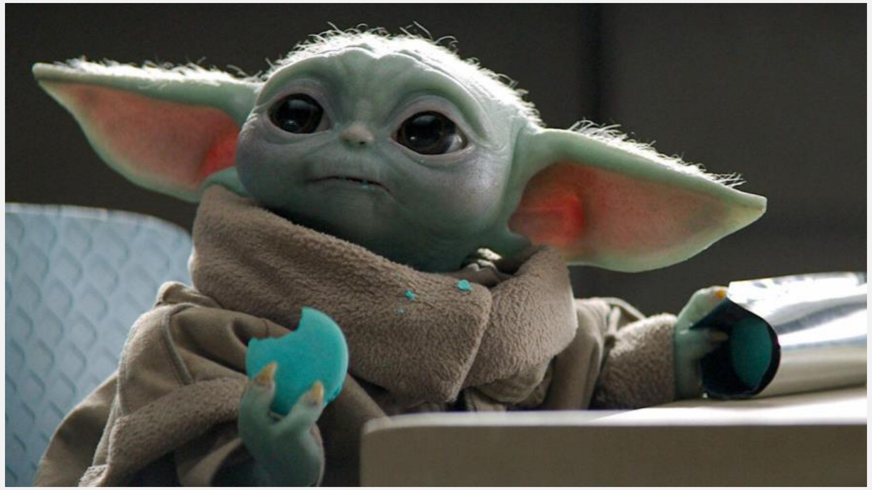
Not in here?



Thm: A_{TM} is undecidable

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

- ???



today

Example (**decidable** languages)

Prove a language is **decidable:**

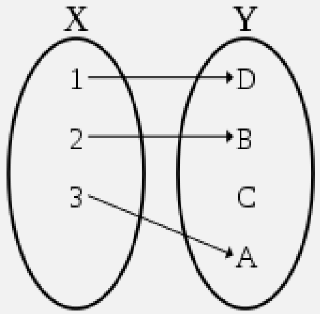
- Create a **decider** TM (with termination argument)

Prove a language is **not decidable:**

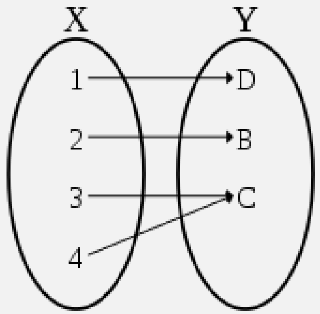
- ????

Kinds of Functions (a fn maps DOMAIN → RANGE)

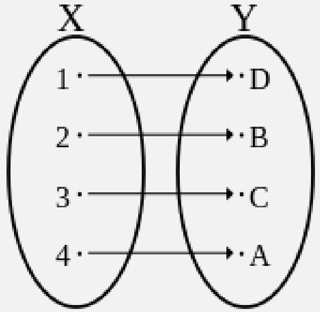
- **Injective,** a.k.a., "one-to-one"
  - Every element in DOMAIN has a unique mapping
  - How to remember:
    - Entire DOMAIN is mapped "**in**" to the RANGE



- **Surjective,** a.k.a., "onto"
  - Every element in RANGE is mapped to
  - How to remember:
    - "**Sur**" = "over" (eg, survey); DOMAIN is mapped "over" the RANGE



- **Bijective,** a.k.a., "correspondence" or "one-to-one correspondence"
  - Is both injective and surjective
  - Unique pairing of every element in DOMAIN and RANGE



Countability

- A set is "**countable**" if it is:
  - Finite
  - Or, there exists a **bijection** between the set and the natural numbers
    - In this case, the set has the same size as the set of natural numbers
    - This is called "**countably infinite**"

Exercise: Which set is larger?

- The set of:
  - Natural numbers, or
  - Even numbers?
- They are the same size! Both are **countably infinite**
  - Proof: Bijection:

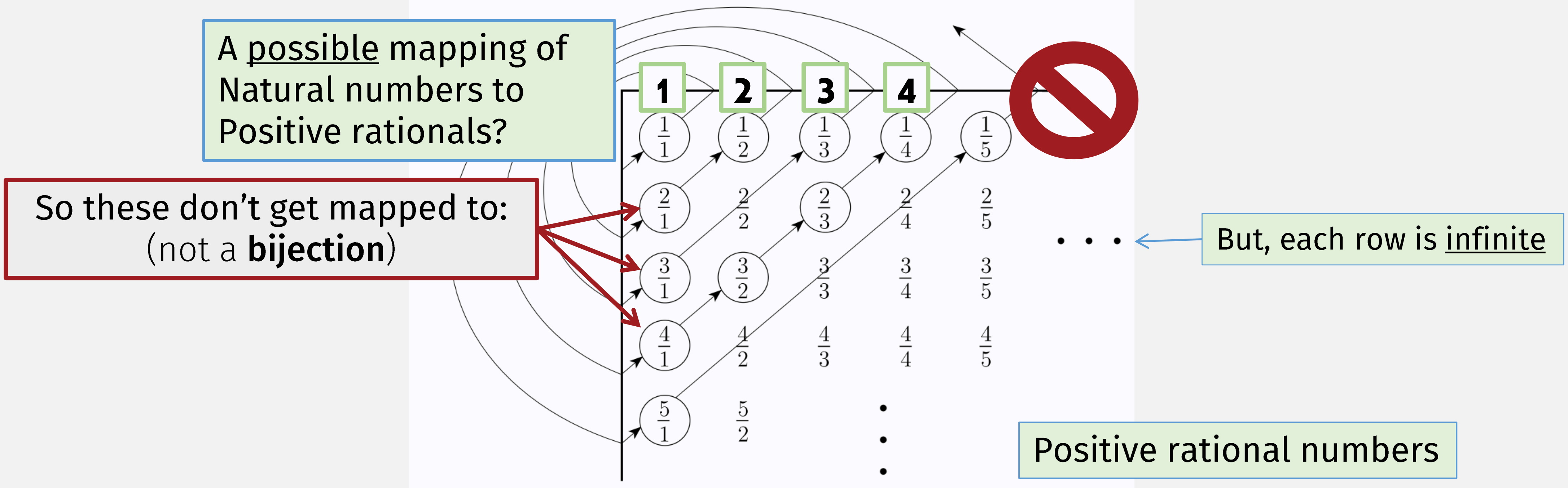
n	$f(n) = 2n$
1	2
2	4
3	6
$\vdots$	$\vdots$

Natural numbers | Even numbers

Every natural number maps to a unique even number, and vice versa

Exercise: Which set is larger?

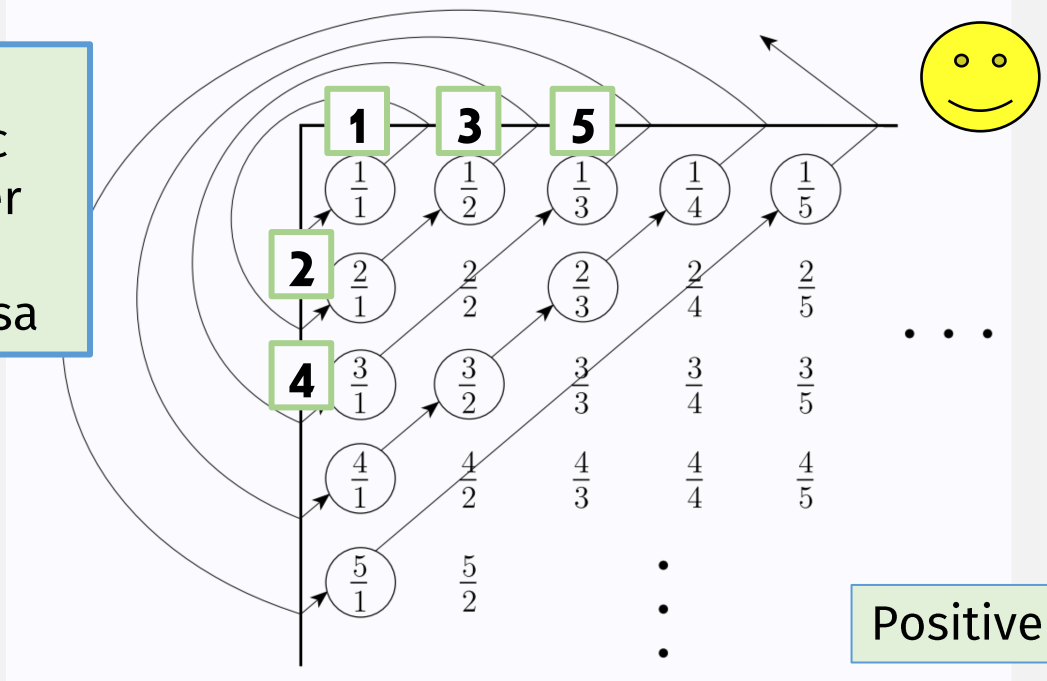
- The set of:
  - Natural numbers $\mathcal{N}$, or
  - Positive rational numbers? $\mathcal{Q} = \{\frac{m}{n} | m, n \in \mathcal{N}\}$
- They are the same size! Both are **countably infinite**



Exercise: Which set is larger?

- The set of:
  - Natural numbers $\mathcal{N}$, or
  - Positive rational numbers? $\mathcal{Q} = \{\frac{m}{n} | m, n \in \mathcal{N}\}$
- They are the same size! Both are **countably infinite**

Another mapping:
This is a **bijection** bc every natural number maps to a unique fraction, and vice versa



Positive rational numbers

Exercise: Which set is larger?

- The set of:
  - Natural numbers $\mathcal{N}$, or
  - Real numbers? $\mathcal{R}$
- There are more real numbers. It is **uncountably infinite.**

This proof technique is called **diagonalization**

Proof, by contradiction:

- Assume a bijection between natural and real numbers exists.
  - So: every natural num maps to a unique real, and vice versa
- But we show that in any given mapping, … e.g.:
  - Some real number is not mapped to …
  - E.g., a number that has different digits at each position:

$$x = 0.4641\ \ldots$$

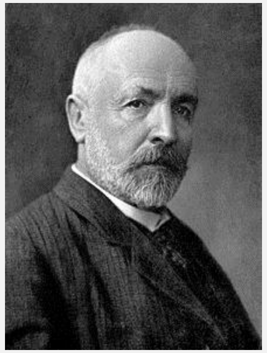
different

n	$f(n)$
1	$3.14159\ldots$
2	$55.55555\ldots$
3	$0.12345\ldots$
4	$0.50000\ldots$
$\vdots$	$\vdots$

A hypothetical mapping

  - This number cannot be in the mapping …
  - … So we have a **contradiction**!

Georg Cantor



- Invented set theory
- Came up with **countable infinity** (1873)
- And **uncountability**:
  - Also: how to show uncountability with "**diagonalization**" technique

A formative day for Georg Cantor.

Diagonalization with Turing Machines

<u>Diagonal</u>: Result of Giving a TM its own Encoding as Input

All TM Encodings

All TMs	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	<u>*accept*</u>	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	<u>*accept*</u>	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	<u>*reject*</u>	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	<u>*reject*</u>		*accept*	
$\vdots$		$\vdots$			$\ddots$		
~~D~~	~~*reject*~~	~~*reject*~~	~~*accept*~~	~~*accept*~~		<u>?</u>	
$\vdots$		$\vdots$					

opposites

Try to construct "opposite" TM D

TM D can't exist!

What should happen here?

It must both accept and reject!

3 Easy Steps!

Thm: A_{TM} is undecidable

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

Proof by contradiction:

1. **Assume** A_{TM} is decidable. So there exists a decider H for it:

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

2. **Use** H in another TM … the impossible "opposite" machine:

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

H computes M's result with itself as input

Do the opposite

From previous slide (does opposite of what input TM would do if given itself)

3 Easy Steps!

Thm: A_{TM} is undecidable

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

Proof by contradiction:

This cannot be true

1. **Assume** A_{TM} is decidable. So there exists a decider H for it:

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

2. **Use** H in another TM … the impossible "opposite" machine:

$D =$ "On input $\langle M\rangle$, where M is a TM:
1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

3. But D does not exist! **Contradiction**! So the assumption is false.

Easier Undecidability Proofs

- We proved $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ undecidable …

  … by contradiction:
- By showing its decider can help create impossible decider "D"!

- Hard: **Coming up with "D"** (needed to invent diagonalization)

- But **then we** more easily **reduced** A_{TM} **to "D"**

- Easier: **reduce** problems to A_{TM}!

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*
M_3	*reject*	*reject*	*reject*	*reject*		*reject*
M_4	*accept*	*accept*	*reject*	*reject*		*accept*
$\vdots$			$\vdots$		$\ddots$	
D	*reject*	*reject*	*accept*	*accept*		?

I.e., “Algorithm to determine if a TM is an decider”?

The Halting Problem

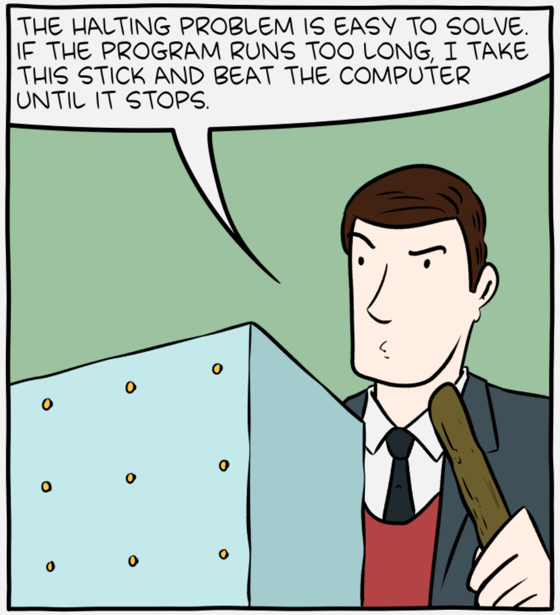
$HALT_{\mathsf{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

Thm: $HALT_{\mathsf{TM}}$ is undecidable

Proof, by **contradiction**:

- **Assume** $HALT_{\mathsf{TM}}$ has decider R; use it to create decider for A_{TM} :
- …
- But A_{TM} is undecidable and has no decider!

contradiction



What if Alan Turing had been an engineer?

The Halting Problem

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

Thm: $HALT_{\mathsf{TM}}$ is undecidable

Proof, by **contradiction**:

Using our hypothetical decider R

- Assume $HALT_{\mathsf{TM}}$ has decider R; use it to create decider for A_{TM} :

S = "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Run TM R on input $\langle M, w\rangle$.
2. If R rejects, *reject*. ← This means M loops on input w
3. If R accepts, simulate M on w until it halts. ← This step always halts
4. If M has accepted, *accept*; if M has rejected, *reject*."

Termination argument:
Step 1: R is a decider so always halts
Step 3: M always halts bc R said so

The Halting Problem

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

Thm: $HALT_{\mathsf{TM}}$ is undecidable

Proof, by **contradiction**:

- Assume $HALT_{\mathsf{TM}}$ has decider R; use it to create decider for A_{TM} :

~~S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:~~
~~1. Run TM R on input $\langle M, w \rangle$.~~
~~2. If R rejects, *reject*.~~
~~3. If R accepts, simulate M on w until it halts.~~
~~4. If M has accepted, *accept*; if M has rejected, *reject*."~~

- But A_{TM} is undecidable!
  - I.e., the decider we just created does not exist! So $HALT_{\mathsf{TM}}$ is undecidable

Easier Undecidability Proofs

In general, to prove the undecidability of a language,
use **proof by contradiction**:

1. Assume the language is decidable (and thus has a **decider**)

2. Show that its decider can be used to create another decider …

   … for a known undecidable language …

3. … which cannot have a decider! That's a **Contradiction**!

Summary: The Limits of Algorithms

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ — Decidable
- $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ — **Undecidable**
- $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ — Decidable
- $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ — Decidable
- $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — **Undecidable**

next

Check-in Quiz 4/10

On gradescope