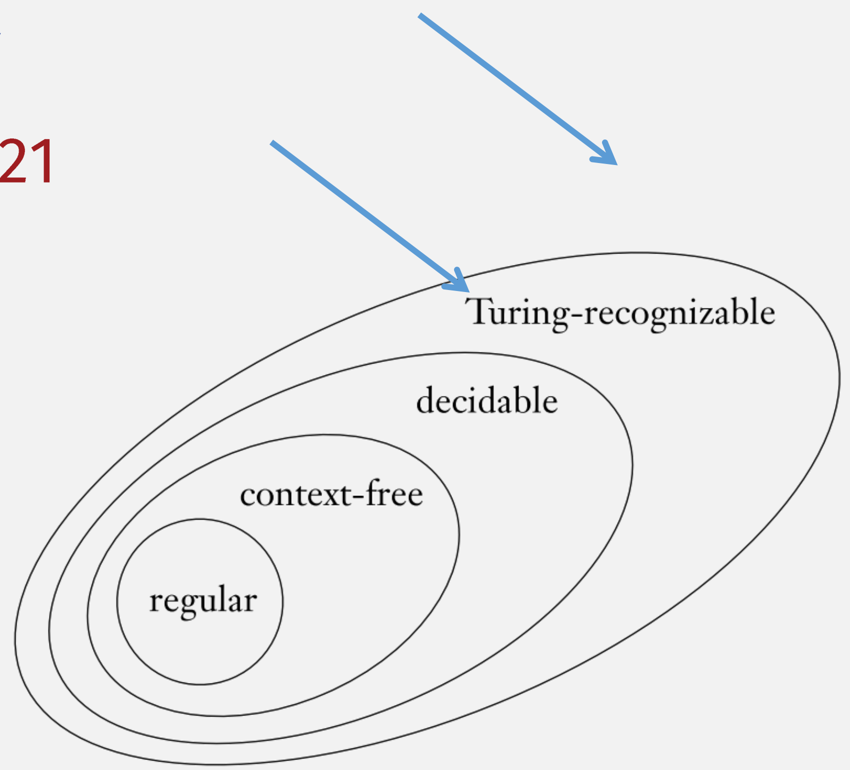
UMBCS622

Undecidability

Wednesday, October 20, 2021



Announcements

- HW 5 due Sun 10/24 11:59pm

Last Time: Decidable Algorithms About Regular Languages

Remember:
TMs = programs
Creating TM = programming
Previous theorems = library

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$

- $A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$

- $A_{\text{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$

- $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$

- $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability." **Bill Gates, April 18, 2002. Keynote address at WinHec**

Last Time: Decidable Algorithms CFLs

- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
  - Convert grammar to Chomsky Normal Form
  - Then check all possible derivations of length 2|*w*| - 1 steps

next

- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$

Thm: E_{CFG} is a decidable language

$E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$

Recall:

$E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$

T = "On input $\langle A\rangle$, where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, *accept*; otherwise, *reject*."

"Reachability" (of accept state from start state) algorithm

Thm: E_{CFG} is a decidable language

$E_{\mathsf{CFG}} = \{\langle G \rangle |\ G \text{ is a CFG and } L(G) = \emptyset\}$

Now create decider that calculates reachability for grammar G

- Except go backwards, start from terminals, to avoid looping

If loop marks at least 1 variable on each iteration, then it eventually terminates because there are finite variables; else loop terminates

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

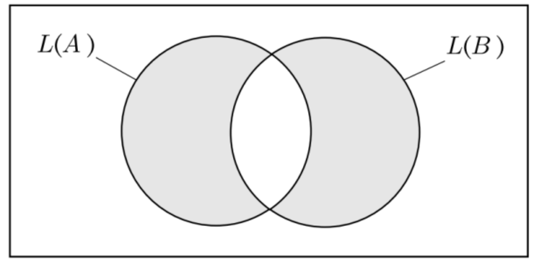
Termination argument?

Thm: EQ_{CFG} is a decidable language?

$EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

Recall: $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

- Used Symmetric Difference



$L(C) = \emptyset \text{ iff } L(A) = L(B)$

  - where C = complement, union, intersection of machines A and B
- Can't do this for CFLs!
  - Intersection and complement are not closed for CFLs!!!

Intersection of CFLs is Not Closed!

Proof (by contradiction), Assume intersection is closed for CFLS

- Then intersection of these CFLs should be a CFL:

$$A = \{\mathtt{a}^m \mathtt{b}^n \mathtt{c}^n \mid m, n \geq 0\}$$

$$B = \{\mathtt{a}^n \mathtt{b}^n \mathtt{c}^m \mid m, n \geq 0\}$$

- But $A \cap B$ = $\{\mathtt{a}^n \mathtt{b}^n \mathtt{c}^n \mid n \geq 0\}$

- … which is not a CFL! (So we have a contradiction)

Complement of a CFL is not Closed!

- If CFLs closed under complement:

$$\text{if } G_1 \text{ and } G_2 \text{ context-free}$$

$$\overline{L(G_1)} \text{ and } \overline{L(G_2)} \text{ context-free}$$

From the assumption

$$\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free}$$

Union of CFLs is closed

$$\overline{\overline{L(G_1)} \cup \overline{L(G_1)}} \text{ context-free}$$

From the assumption

$$L(G_1) \cap L(G_2) \text{ context-free}$$

DeMorgan's Law!

But intersection is not closed for CFLS (prev slide)

Thm: EQ_{CFG} is a decidable language?

$EQ_{\mathsf{CFG}} = \{\langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

- No! 
  - There's no algorithm to decide whether two grammars are equivalent!

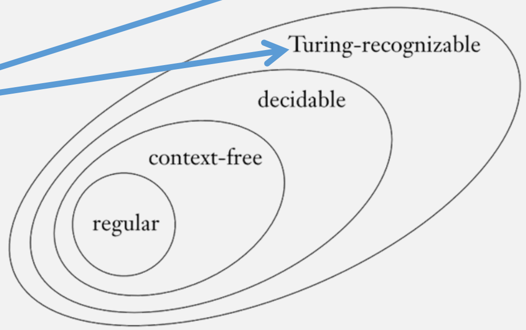
- It's not recognizable either!

Summary of Decidable Algorithms for CFLs

- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
  - Convert grammar to Chomsky Normal Form
  - Then check all possible derivations of length 2|*w*| - 1 steps
- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$
  - Compute "reachability" of start variable from terminals
- $EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
  - We couldn't prove that this is decidable!
  - (So you cant use this theorem when creating another decider)

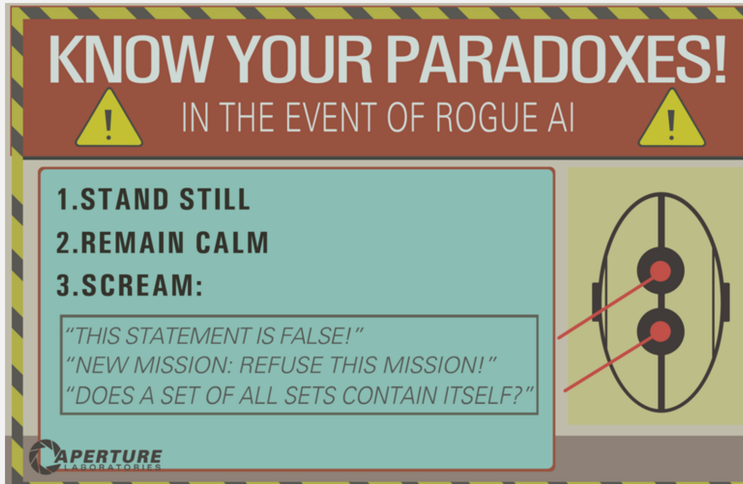
The Limits of Turing Machines?

- TMs represent all possible "computations"
  - I.e., any (Python, Java, …) program you write is a TM
- So what is **not** computable? I.e., what's here?



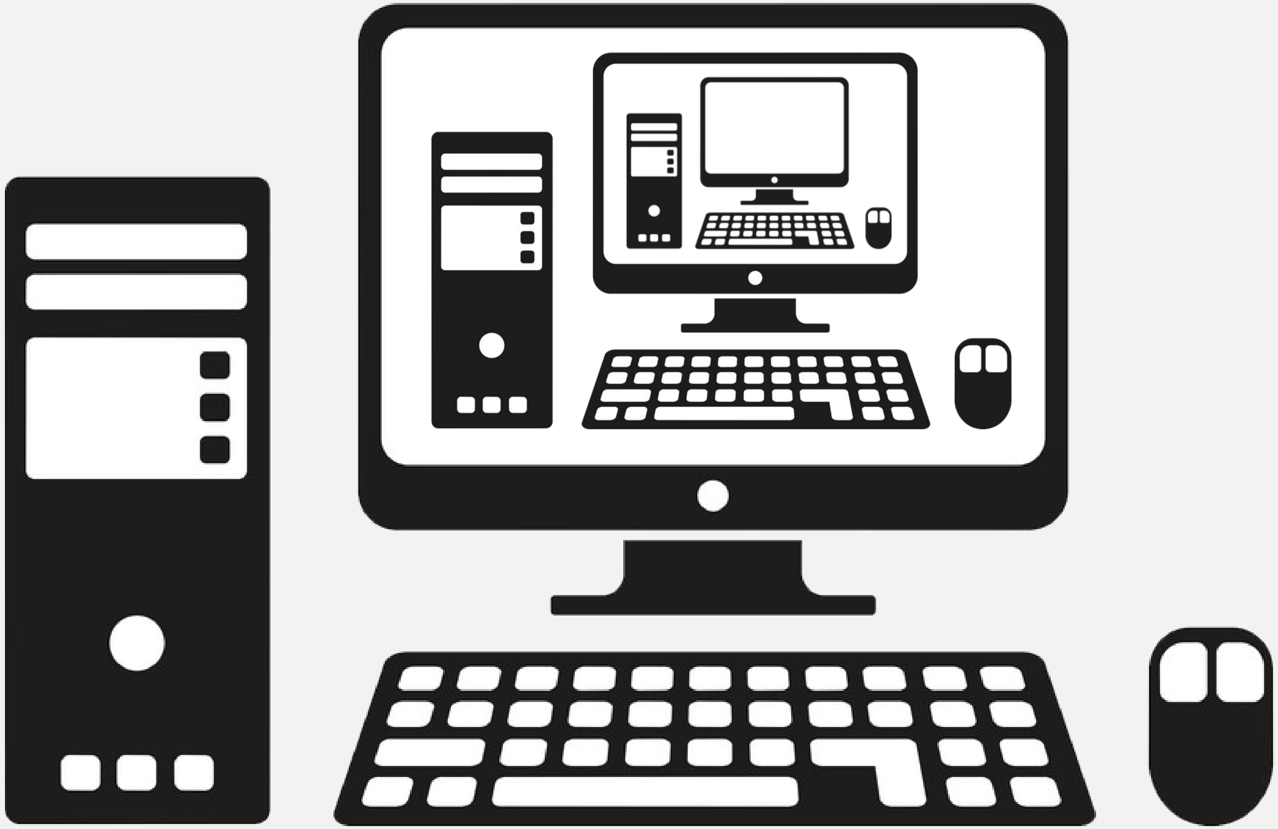
- A way to test the limit of a computational model is to see what it can compute about computational models …
  - **Thought**: Is there an algorithm to determine whether a TM is an algorithm?

Hmmm



Is A_{TM} is undecidable ???

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



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.
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

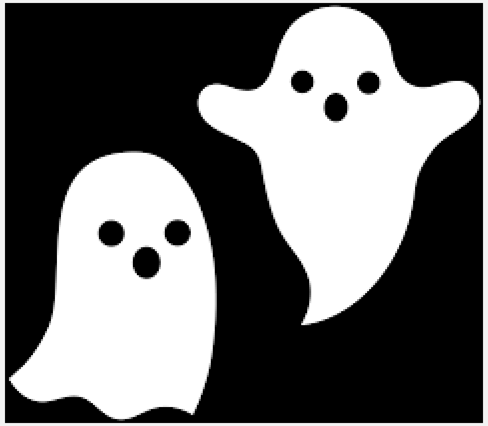
U = Extended delta "run" function for TMs

- Computer that can simulate other computers
- i.e., "The Universal Turing Machine"
- Problem: *U* loops when *M* loops

Thm: A_{TM} is undecidable

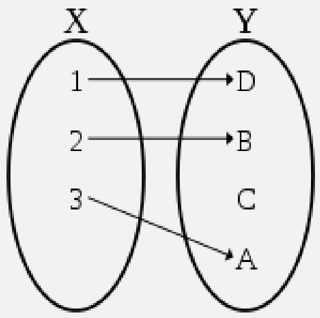
$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$

- ???

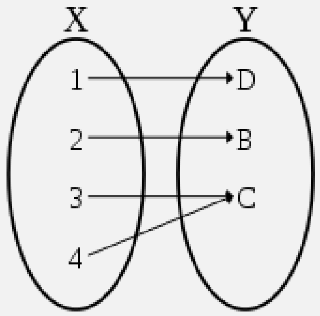


Kinds of Functions (a fn maps DOMAIN → RANGE)

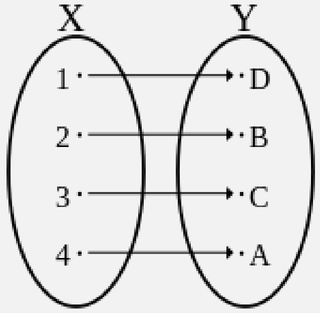
- **Injective,** a.k.a., “one-to-one”
  - Every element in DOMAIN has a unique mapping
  - <u>How to remember</u>:
    - Entire DOMAIN is mapped “in” to the RANGE



- **Surjective,** a.k.a., “onto”
  - Every element in RANGE is mapped to
  - <u>How to remember</u>:
    - “Sur” = “over” (eg, <u>sur</u>vey); 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
    - This set has the <u>same size</u> as the set of natural numbers
    - This is called "countably infinite"

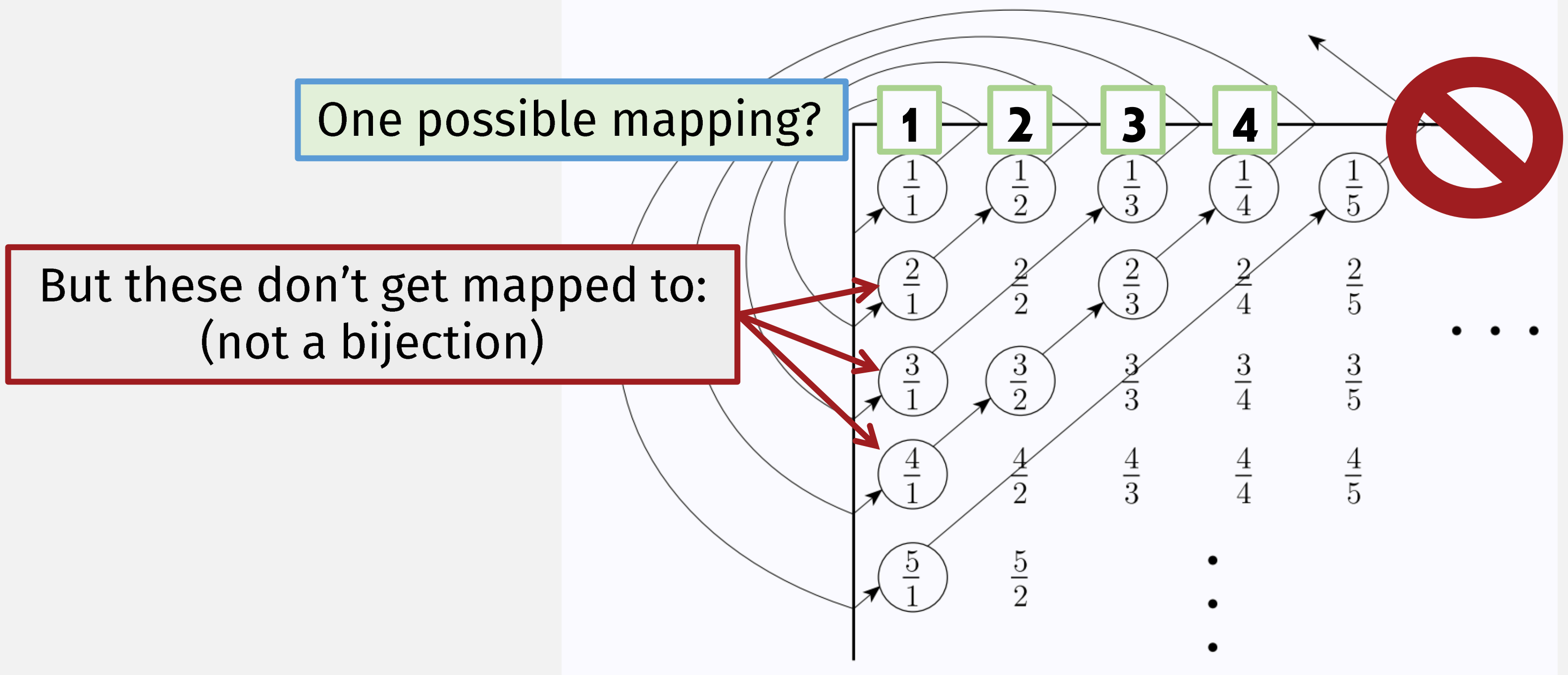
<u>Exercise</u>: Which set is larger?

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

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

Exercise: Which set is larger?

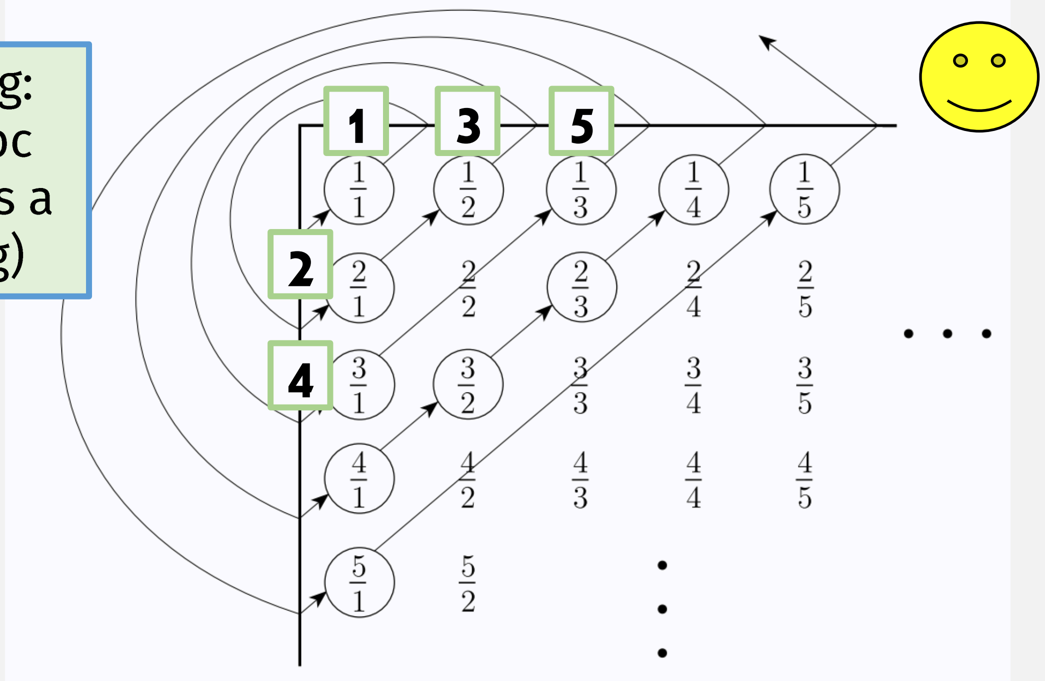
- The set of:
  - Natural numbers $\mathcal{N}$, or
  - Positive rational numbers? $\mathcal{Q} = \{\frac{m}{n} | \, m, n \in \mathcal{N}\}$
- They are the **same** size! Both are countably infinite



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:
(it's a bijection bc every fraction has a unique mapping)



Exercise: Which set is larger?

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

This is called **"diagonalization"**

Proof, by contradiction:

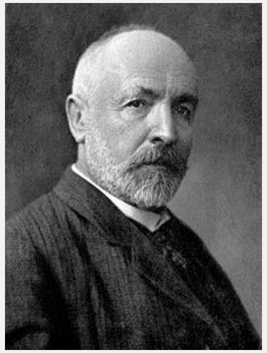
- Assume a bijection between natural and real numbers exists.
  - This means that every real number should get mapped to.
- But we show that in any given mapping, … e.g.:
  - Some real number is not mapped to …
  - E.g., a number that has different digits at each position:

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

$$x = 0.4641\ldots$$

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

Georg Cantor



- Invented set theory
- Came up with countable infinity in 1873
- And uncountability:
  - And how to show uncountability with “diagonalization” technique

A formative day for Georg Cantor.

Diagonalization with Turing Machines

Diagonal: Result of Giving a TM its own Encoding as Input

All TM Encodings

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

opposites

All TMs

Try to construct "opposite" TM

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. Then there exists a decider:

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

2. If H exists, then we can create an "opposite" machine:

From the previous slide

$D =$ "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*."

Result of giving a TM itself as input

3 Easy Steps!

<u>Thm</u>: A_{TM} is undecidable

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

<u>Proof</u> by contradiction:

1. <u>Assume</u> A_{TM} is decidable. Then there exists a decider:

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

2. <u>If</u> H exists, <u>then</u> we can create an "opposite" machine:

From the previous slide

~~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! <u>Contradiction</u>! So 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 …
- … specifically, showing that its decider could be used to implement an impossible decider “D”!

- Coming up with “D” was hard (needed to invent diagonalization)

- But then we more easily **reduced** A_{TM} to the “D”

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$
M_1	$\underline{accept}$	$reject$	$accept$	$reject$		$accept$
M_2	$accept$	$\underline{accept}$	$accept$	$accept$	$\cdots$	$accept$
M_3	$reject$	$reject$	$\underline{reject}$	$reject$		$reject$
M_4	$accept$	$accept$	$reject$	$\underline{reject}$		$accept$
$\vdots$			$\vdots$		$\ddots$	
D	$reject$	$reject$	$accept$	$accept$		$\underline{?}$

- Now we can also reduce problems to A_{TM}!

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

The Halting Problem

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

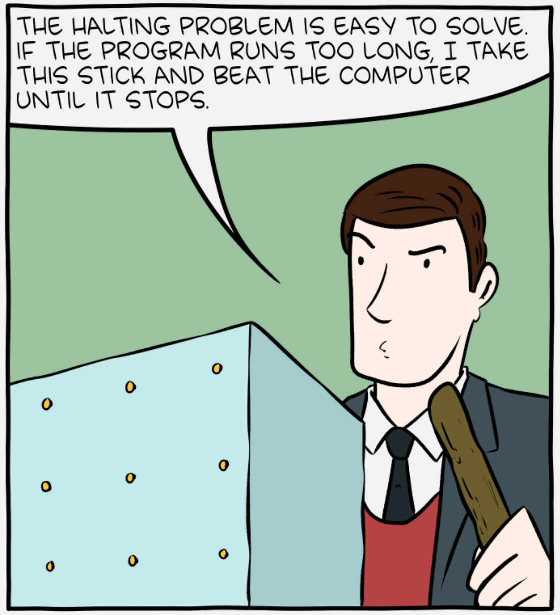
Thm: $HALT_{\mathsf{TM}}$ is undecidable

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!



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:

- 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*."

The Halting Problem

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

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., this decider that we just created cannot exist! So $HALT_{\mathsf{TM}}$ is undecidable

Easier Undecidability Proofs

In general, to prove the undecidability of a language:

- Use proof by contradiction:

- Assume the language is decidable,

- Show that its decider can be used to create a decider for …

- … a known undecidable language …

- … which doesn’t have a decider!

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

Reducibility: Modifying the TM

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

<u>Thm:</u> E_{TM} is undecidable

<u>Proof</u>, by contradiction:

- Assume E_{TM} has *decider* R; use to create A_{TM} *decider*:

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

<u>First</u>, construct M_1

- Run R on input $\langle M_1 \rangle$
- If R accepts, *reject* (because it means $\langle M \rangle$ doesn't accept w)
- if R rejects, then *accept* ($\langle M \rangle$ accepts w)

Note: M_1 is <u>only</u> used as arg to R; we never run it!

- <u>Idea</u>: Wrap $\langle M \rangle$ in a new TM that <u>can *only* accept *w*</u>:

$M_1 =$ "On input x:
1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by contradiction:

This decider R cannot exist!

- Assume E_{TM} has *decider* R; use to create A_{TM} *decider*:

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

First, construct M_1

- Run R on input $\langle M_1\rangle$
- If R accepts, *reject* (because it means $\langle M\rangle$ doesn't accept w)
- if R rejects, then *accept* ($\langle M\rangle$ accepts w)

- Idea: Wrap $\langle M\rangle$ in a new TM that can only accept w:

M_1 = "On input x:
1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

One more, modify M: $REGULAR_{\mathsf{TM}}$ is undecidable

$REGULAR_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

Proof, by contradiction:

- Assume $REGULAR_{\mathsf{TM}}$ has *decider* R; use to create A_{TM} *decider*:

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

- First, construct M_2 (??)
- Run R on input $\langle M_2 \rangle$
- If R accepts, *accept*; if R rejects, *reject*

Want: $L(M_2)$ =
- regular, if M accepts w
- nonregular, if M does not accept w

Thm: $REGULAR_{\mathsf{TM}}$ is undecidable (continued)

$REGULAR_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

$M_2 =$ "On input x:

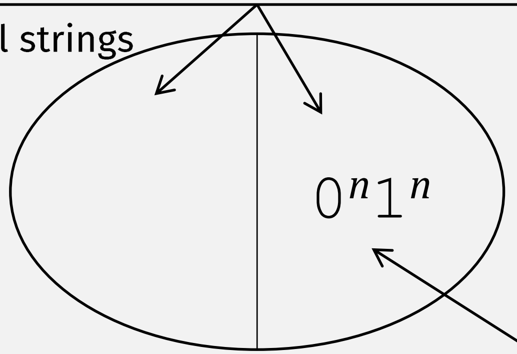
1. If x has the form 0^n1^n, *accept*.
2. If x does not have this form, run M on input w and *accept* if M accepts w."

Always accept strings 0^n1^n
$L(M_2)$ = nonregular, so far

If M accepts w,
accept everything else,
so $L(M_2) = \Sigma^* =$ regular

if M <u>does not</u> accept w, M_2 accepts all strings (regular lang)

All strings



if M accepts w, M_2 accepts this non-regular lang

<u>Want</u>: $L(M_2)$ =

- regular, if M accepts w
- nonregular, if M does not accept w

Summary: The Limits of Algorithms

- $A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ — Decidable
- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ — **Undecidable**
- $E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ — Decidable
- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ — Decidable
- $E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — **Undecidable**
- $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ — Decidable
- $EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ — **Undecidable**
- next $EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ — **Undecidable**

needs

Reduce to something else: EQ_{TM} is undecidable

$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Proof, by contradiction:

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

- Assume EQ_{TM} has *decider* R; use to create $\cancel{A_{\mathsf{TM}}}$ *decider*:

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

1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.

2. If R accepts, *accept*; if R rejects, *reject*."

Reduce to something else: EQ_{TM} is undecidable

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Proof, by contradiction:

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

- Assume EQ_{TM} has *decider* R; use to create $\cancel{A_{\mathsf{TM}}}$ *decider*:

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

1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.
2. If R accepts, *accept*; if R rejects, *reject*."

- But E_{TM} is undecidable!

Summary

- $A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ Decidable
- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ Decidable
- $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ **Undecidable**
- $E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ Decidable
- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ Decidable
- $E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ **Undecidable**
- $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ Decidable
- $EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ **Undecidable**
- $EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ **Undecidable**

There's no algorithm to compute <u>anything</u> about Turing Machines, i.e., **about general programs!**

Also Undecidable …

today

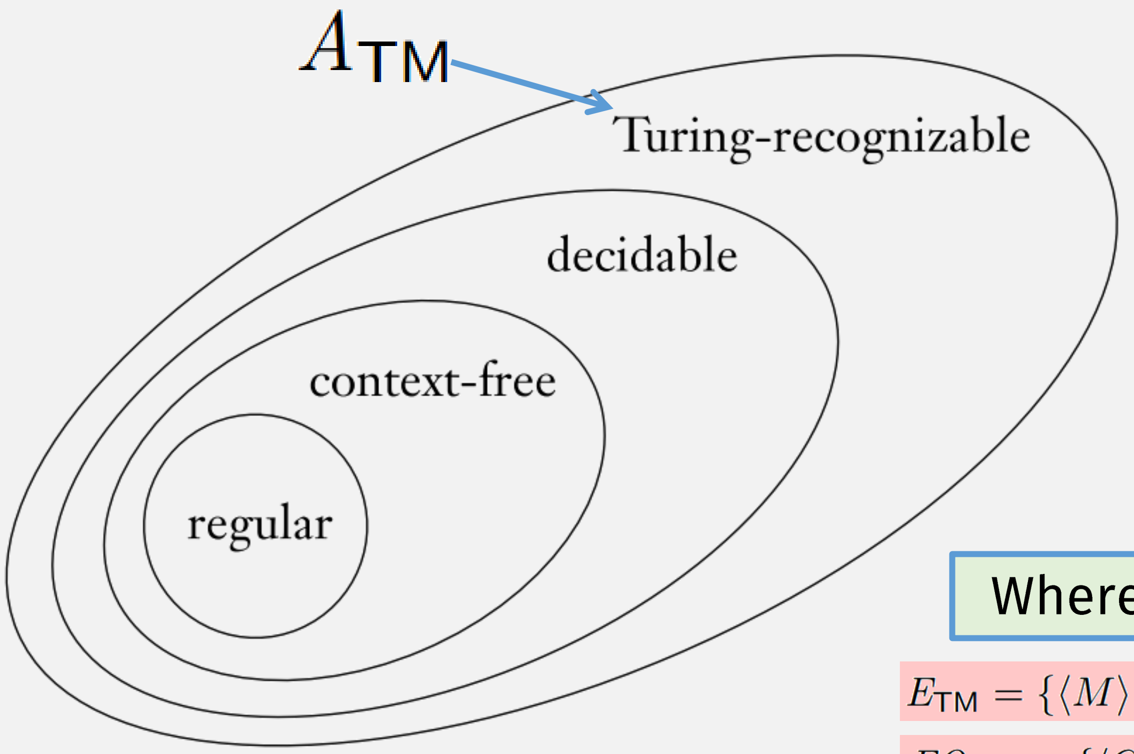
- $REGULAR_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a regular language}
- $CONTEXTFREE_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a CFL}
- $DECIDABLE_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a decidable language}
- $FINITE_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a finite language}
- …

Rice's Theorem

- $ANYTHING_{\text{TM}}$ = {<M> | M is a TM and "**anything**" about $L(M)$}

Turing Unrecognizable?

Is there anything out here?



Where do these go?

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

$EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Thm: Some langs are not Turing-recognizable

Proof: requires 2 lemmas

- Lemma 1: The **set of all languages** is *uncountable*
  - Proof: Show there is a bijection with another uncountable set …
    - … The set of all infinite binary sequences
- Lemma 2: The **set of all TMs** is *countable*
- Therefore, some language is not recognized by a TM

Mapping a Language to a Binary Sequence

All Possible Strings

Some Language (subset of above)

Its (unique) Binary Sequence

$$\begin{array}{lcccccccccccc} \Sigma^* & = & \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots \} \\ A & = & \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots \} \\ \chi_A & = & & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots \end{array}$$

Each digit represents one possible string:
- 1 if lang has that string,
- 0 otherwise

Thm: Some langs are not Turing-recognizable

Proof: requires 2 lemmas

- Lemma 1: The **set of all languages** is *uncountable*
  - Proof: Show there is a bijection with another uncountable set …
    - … The set of all infinite binary sequences
    - ➢ Now just prove set of infinite binary sequences is uncountable (diagonalization)

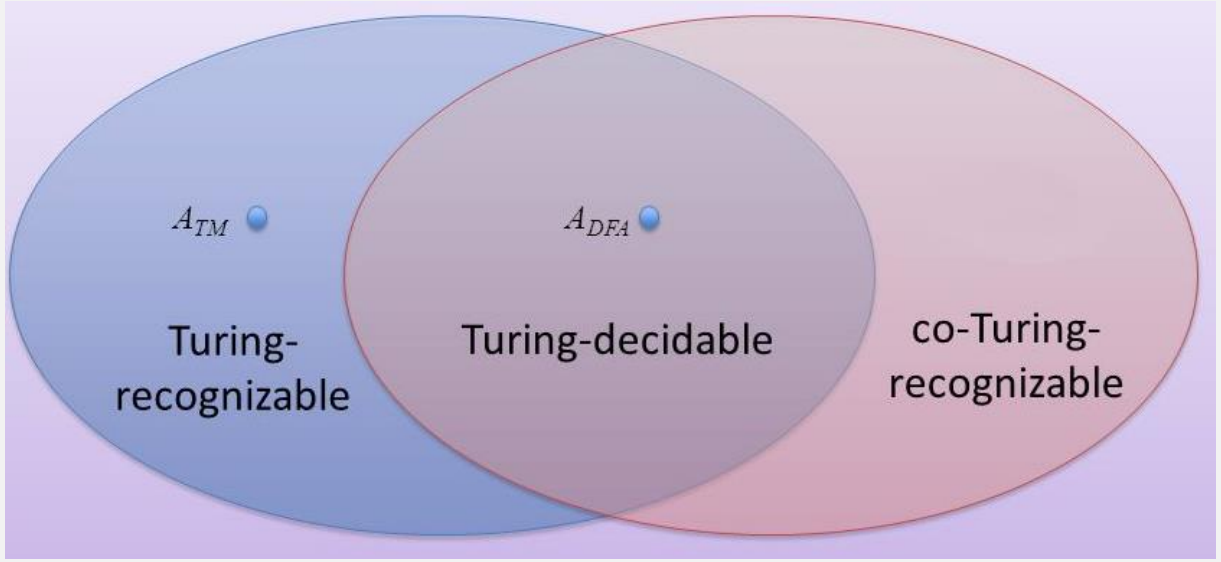
- Lemma 2: The **set of all TMs** is *countable*
  - Because every TM M can be encoded as a string $<M>$
  - And set of all strings is countable

- Therefore, some language is not recognized by a TM $\blacksquare$

Co-Turing-Recognizability

- A language is **co-Turing-recognizable** if …
- … it is the <u>complement</u> of a Turing-recognizable language.

Thm: Decidable ⇔ Recognizable & co-Recognizable



Thm: Decidable ⇔ Recognizable & co-Recognizable

⇒ If a language is **decidable,** then it is **recognizable** and **co-recognizable**

- Decidable => Recognizable:
  - A decider is a recognizer, bc decidable langs are a subset of recognizable langs
- Decidable => Co-Recognizable:
  - To create co-decider from a decider … switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

⇐ If a language is **recognizable** and **co-recognizable,** then it is **decidable**

<u>Thm</u>: Decidable ⇔ Recognizable & co-Recognizable

⇒ <u>If</u> a language is **decidable**, <u>then</u> it is **recognizable** and **co-recognizable**

- Decidable => Recognizable:
  - A decider is a recognizer, bc decidable langs are a subset of recognizable langs
- Decidable => Co-Recognizable:
  - To create co-decider from a decider … switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

⇐ <u>If</u> a language is **recognizable** and **co-recognizable**, <u>then</u> it is **decidable**

- Let M_1 = recognizer for the language,
- and M_2 = recognizer for its complement
- Decider M:
  - Run 1 step on M_1,
  - Run 1 step on M_2,
  - Repeat, until one machine accepts. If it's M_1, accept. If it's M_2, reject

Termination Arg: Either M_1 or M_2 must accept and halt, so M halts and is a decider

A Turing-unrecognizable language

- We've proved:

  A_{TM} is Turing-recognizable

  A_{TM} is undecidable

- So:

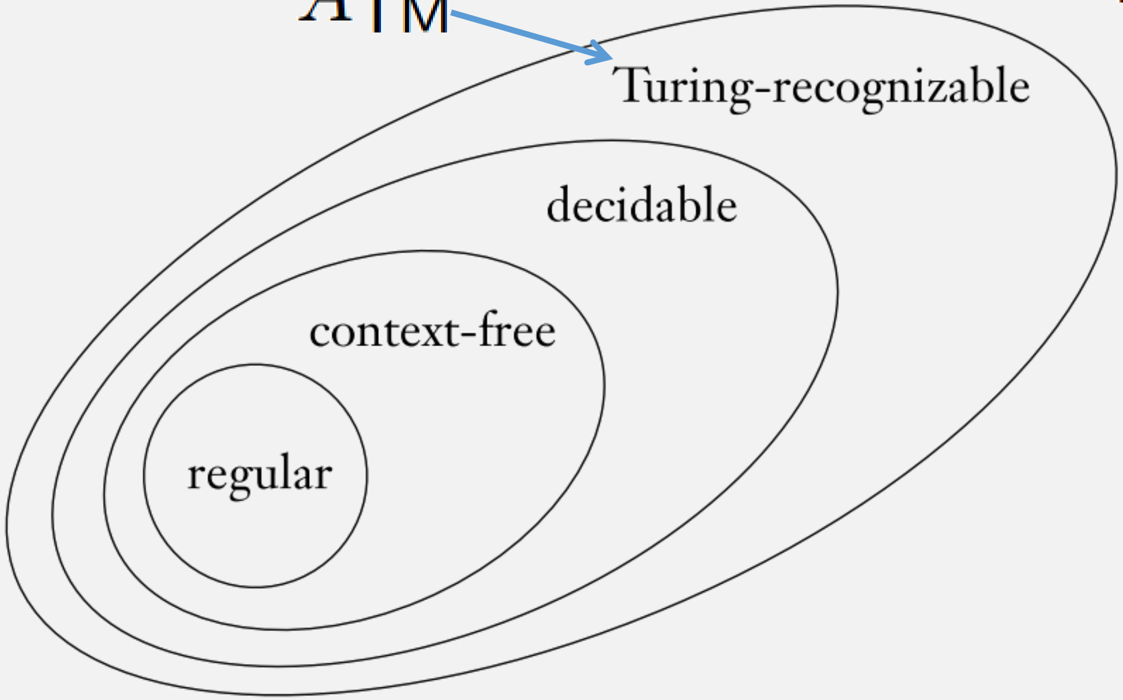
  $\overline{A_{\mathsf{TM}}}$ is not Turing-recognizable

- Because: recognizable & co-recognizable implies decidable

Is there anything out here?

$\overline{A_{\mathsf{TM}}}$

A_{TM}



Check-in Quiz 10/20

On gradescope