CS420
Chapter 5: Reducibility

Wed, November 4, 2020

HW 6/7 Questions?

HW announcements

- HW4 grades released
- HW7 released
- New partner required starting from hw7

Last time: Diagonalization of TMs

Result of giving a TM itself 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	*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*		?	
$\vdots$			$\vdots$				$\ddots$

Opposite

Contradiction: Needs to be both reject and accept

TM *D* can't exist!

Last time: 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.
- Assume A_{TM} is decidable. Then there exists a decider:

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

- If H exists, then we can create:

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

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

- But D does not exist! Contradiction!

Reducibility

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

- We proved A_{TM} undecidable by showing that its decider …
- … could be used to implement an impossible "D" decider.
  - Was hard to prove (diagonalization)

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$
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*		?

- In other words, we **reduced** A_{TM} to the "D" problem.

- But now we can just reduce things to A_{TM}: much easier!

The Halting Problem

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

- <u>Thm</u>: $HALT_{\mathsf{TM}}$ is undecidable
- <u>Proof</u>, by contradiction:
- Assume $HALT_{\mathsf{TM}}$ 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:

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} has no decider!

<u>Recall</u> A_{TM}'s *recognizer* (which might loop):

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>Might need to change *M*</u>: E_{TM} is undecidable

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

- <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 anything)
- if R rejects, then *accept* ($\langle M \rangle$ accepts w)

- <u>Idea</u>: Wrap $\langle M \rangle$ in a TM that *only* accepts *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: $REGULAR_{\text{TM}}$ is undecidable

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

- Proof, by contradiction:
- Assume $REGULAR_{\text{TM}}$ 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

<u>Thm</u>: $REGULAR_{\mathsf{TM}}$ is undecidable (continued)

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

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

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."

Can use A_{CFG} decider here

$A_{\mathsf{CFG}} = \{\langle G, w \rangle |\ G \text{ is a CFG that generates string } w\}$

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

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

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

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:

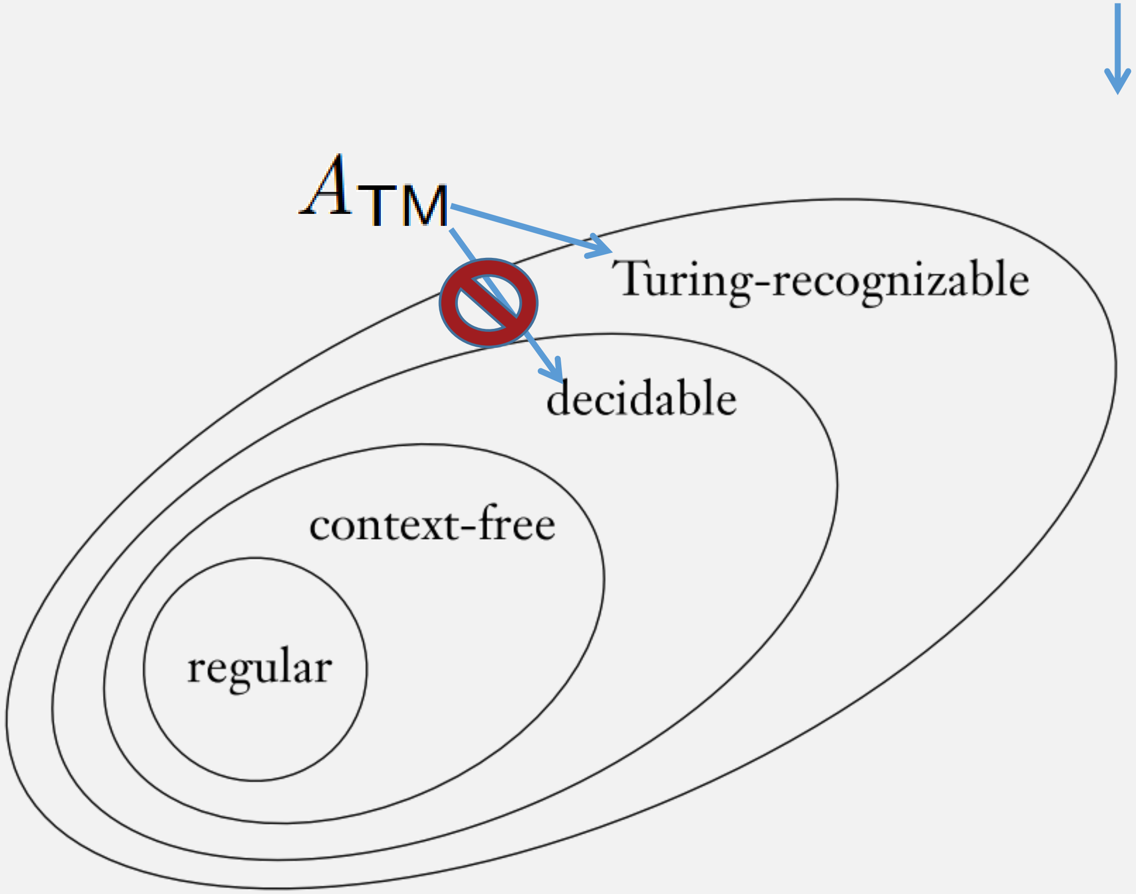
1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.

$L(M) = \emptyset$

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

Turing Unrecognizable?

Is there anything out here?



Thm: Some langs are not Turing-recognizable

- Lemma 1: The **set of all strings** in Σ^* is *countable*
  - Count strings of length 0, then
  - Count strings of length 1, …
- 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 (Lemma 1)
- Lemma 3: The **set of all infinite binary sequences** $\mathcal{B}$ is *uncountable*
  - Diagonalization proof (HW7)
- Lemma 4: The **set of all languages** is *uncountable*
  - There is a mapping to $\mathcal{B}$

Mapping a Lang to a Binary Sequence

All Possible Strings (countable)	$\Sigma^* = \{$	$\varepsilon,$	$0,$	$1,$	$00,$	$01,$	$10,$	$11,$	$000,$	$001,$	$\cdots$	$\}$
Some Language	$A = \{$		$0,$		$00,$	$01,$			$000,$	$001,$	$\cdots$	$\}$
Its Binary Sequence	$\chi_A =$	0	1	0	1	1	0	0	1	1	$\cdots$	

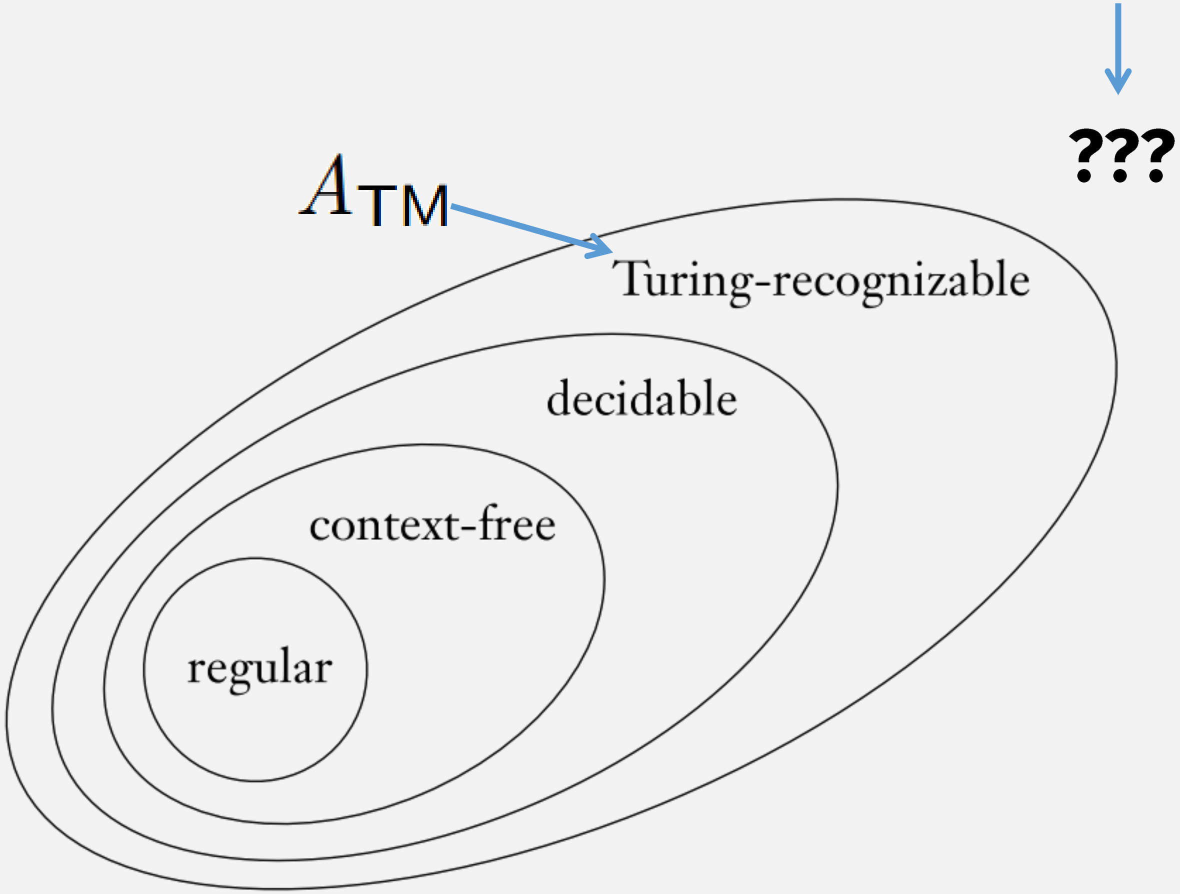
1 if lang has this string, 0 otherwise

Thm: Some langs are not Turing-recognizable

- Lemma 1: The **set of all strings** in Σ^* is *countable*
  - Count strings of length 0, then
  - Count strings of length 1, …
- 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 (Lemma 1)
- Lemma 3: The **set of all infinite binary sequences** $\mathcal{B}$ is *uncountable*
  - Diagonalization proof (HW7)
- Lemma 4: The **set of all languages** is *uncountable*
  - There is a mapping to $\mathcal{B}$
- Corollary 5:
  - TMs countable, langs uncountable => some langs are not Turing-recognizable

Turing Unrecognizable?

Is there anything out here?



Co-Turing-Recognizability

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

Thm: Decidable ⇔ Turing & co-Turing-recognizable

- => If a language is decidable,
  then it is Turing-recognizable and co-Turing-recognizable.
  - Decidable langs ⊂ recognizable langs
    - decidable ➔ Turing-recognizable
  - Complement closed for decidable langs
    - decidable ➔ co-Turing-recognizable

Thm: Decidable ⇔ Turing & co-Turing-recognizable

- => If a language is decidable,
  then it is Turing-recognizable and co-Turing-recognizable.
  - Decidable langs ⊂ recognizable langs
    - decidable ➔ Turing-recognizable
  - Complement closed for decidable langs
    - decidable ➔ co-Turing-recognizable
- <= If a language is Turing- and co-Turing recognizable,
  then it is decidable.
  - Let $M1$ = recognizer for the lang, $M2$ = recognizer for complement
  - Decider M:
    - Run 1 step on $M1$, and 1 step on $M2$,
    - Repeat until one machine accepts. If it's $M1$, accept. If it's $M2$, reject
  - $M1$ or $M2$ must accept and halt, so M halts and is a decider

A Turing-unrecognizable language

- We've proved:

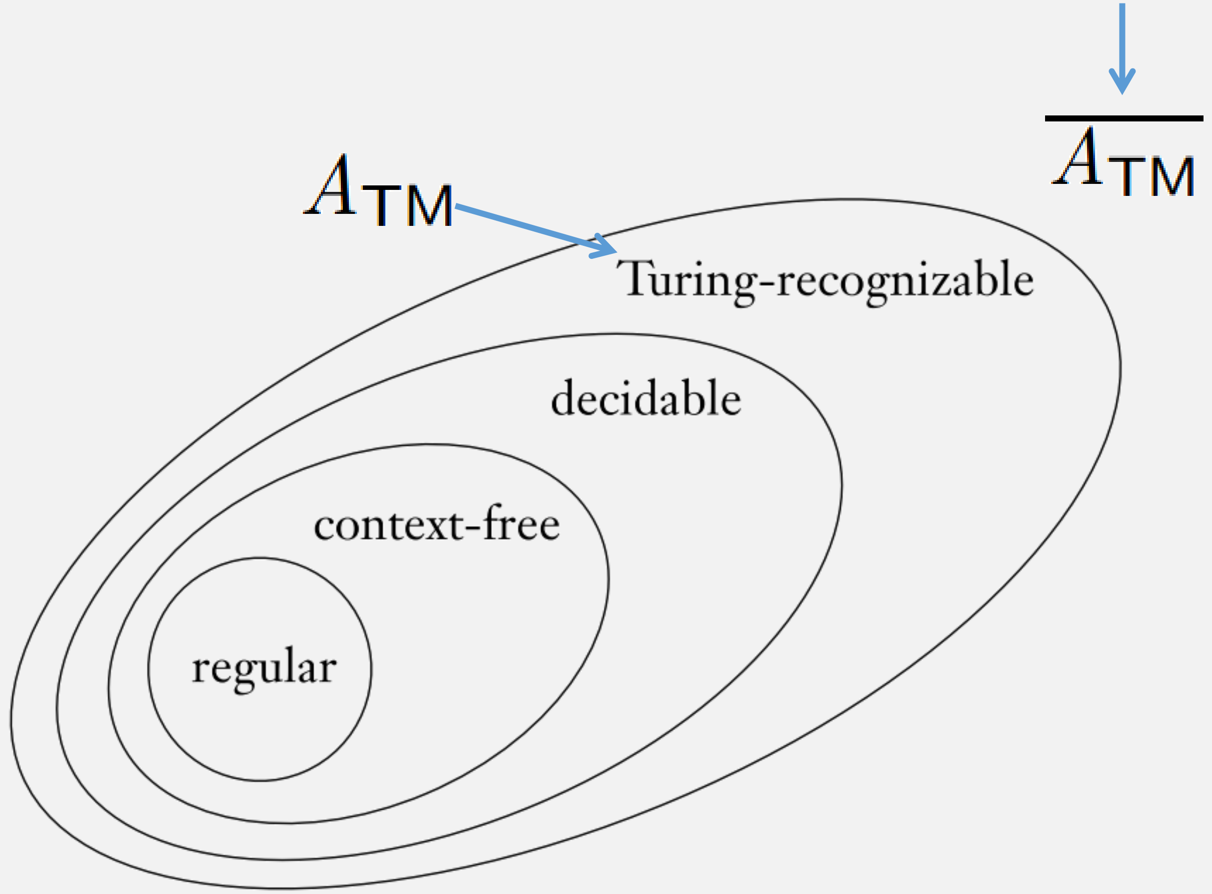
  A_{TM} is Turing-recognizable

  A_{TM} is undecidable

- So:

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

Is there anything out here?



Check-in Quiz 11/4

On gradescope

End of Class Survey 11/4

See course website