CS420

Chapter 5: Reducibility

Monday, April 5, 2021

Announcements

- HW 7 due date past
- HW 8 due Sun 4/11 11:59pm EST
- HW9 out soon
  - Due Sun 4/18 11:59pm EST
  - Ch5-6 material (starting Wed)

Last time: Diagonalization of TMs

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

All TMs

Opposite

"Opposite" machine

Contradiction: Needs to 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:

"Opposite" machine

$D =$ "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*."

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:

$D =$ "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*."

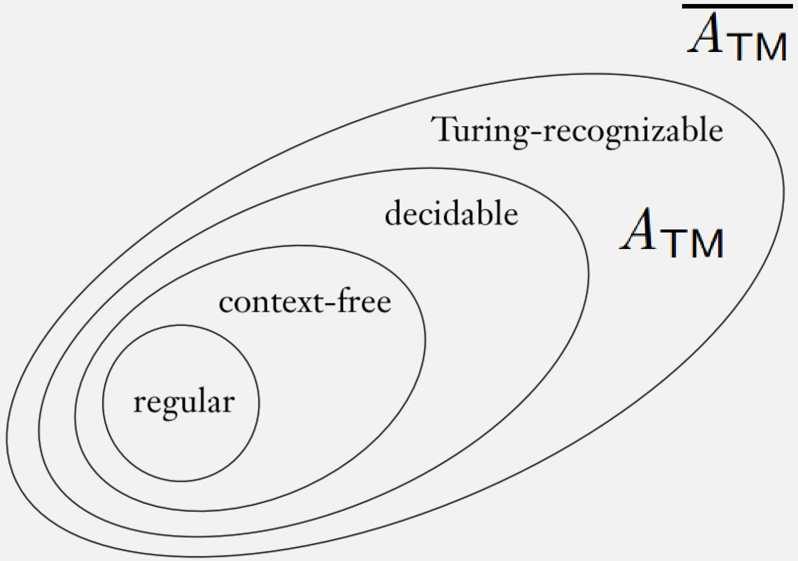
- But D does not exist! Therefore we have a contradiction!

Last time: Unrecognizability

- We've proved:

  A_{TM} is Turing-recognizable

  A_{TM} is undecidable



- And:

  THEOREM 4.22

  A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

- So:

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

Today: 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 …

- … showing that its decider could be used to implement an impossible "D" decider.

- In other words, we **reduced** A_{TM} to the "D" problem.
  - That was hard (needed to invent 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*		?

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

The Halting Problem

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

<u>Thm</u>: $HALT_{\mathsf{TM}}$ is undecidable

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

- Assume $HALT_{\mathsf{TM}}$ has *decider* R; use 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

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

<u>Thm</u>: $HALT_{\mathsf{TM}}$ is undecidable

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

- Assume $HALT_{\mathsf{TM}}$ has *decider* R; use 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: Languages About Machines

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

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, 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:

First, construct M_1

- Run R on input $\langle M_1 \rangle$

Note: M is only an arg; we never actually run M!

- 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."

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)

- <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."

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

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

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 $0^n 1^n$, *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

Want: $L(M_2)$ =

- 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.
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

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

We can't compute <u>anything</u> about Turing Machines, i.e., about programs!

Also Undecidable …

today

- $REGULAR_{\mathrm{TM}}$ = {<M> | M is a TM and $L(M)$ is a regular language}

HW9

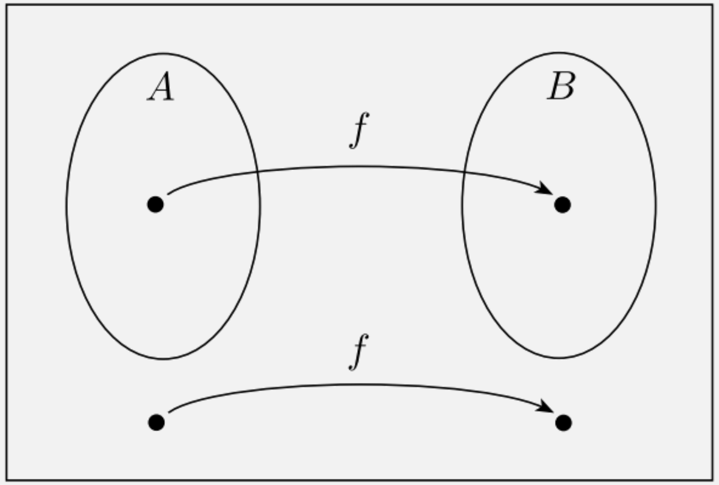
- $CONTEXTFREE_{\mathrm{TM}}$ = {<M> | M is a TM and $L(M)$ is a CFL}
- $DECIDABLE_{\mathrm{TM}}$ = {<M> | M is a TM and $L(M)$ is a decidable language}
- $FINITE_{\mathrm{TM}}$ = {<M> | M is a TM and $L(M)$ is a finite language}
- …

Rice's Theorem

HW9

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

Formalizing Reducibility, i.e., Mapping Reducibility



Flashback: A_{NFA} is a decidable language

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

Decider (i.e., "run" function) for A_{NFA} :

N = "On input $\langle B, w\rangle$, where B is an NFA and w is a string:

1. Convert NFA B to an equivalent DFA C, using the procedure for this conversion given in Theorem 1.39.
2. Run TM M on input $\langle C, w\rangle$.
3. If M accepts, *accept*; otherwise, *reject*."

We said this NFA -> DFA algorithm is a TM, but it doesn't accept/reject?

Computable Functions

- A TM that, instead of accept/reject, “outputs” final tape contents

DEFINITION 5.17

A function $f\colon \Sigma^* \longrightarrow \Sigma^*$ is a ***computable function*** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

- <u>Example 1</u>: All arithmetic operations

- <u>Example 2</u>: Machine conversion algorithms, like DFA -> NFA
  - E.g., adding states, changing transitions, wrapping TM in TM, etc.

Mapping Reducibility

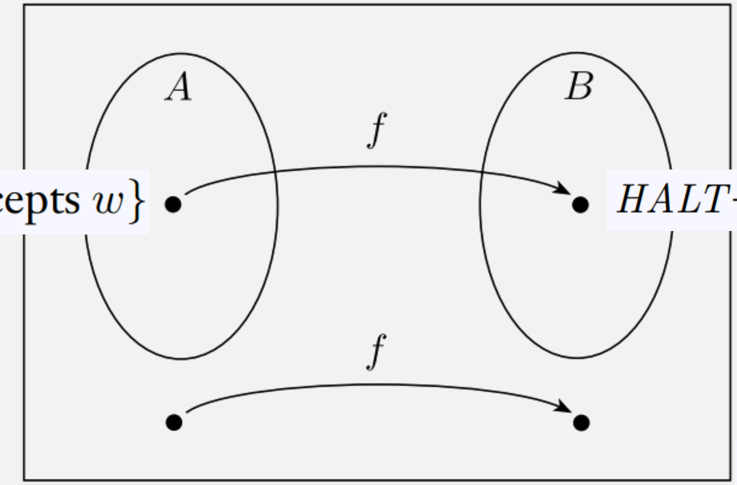
DEFINITION 5.20

Language A is ***mapping reducible*** to language B, written $\boxed{A \leq_{\mathrm{m}} B}$, if there is a $\boxed{\text{computable function } f\colon \Sigma^* \longrightarrow \Sigma^*}$, where for every w,

$$\boxed{w \in A \iff f(w) \in B.}$$

The function f is called the ***reduction*** from A to B.

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



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

DEFINITION 5.17

A function $f\colon \Sigma^* \longrightarrow \Sigma^*$ is a ***computable function*** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

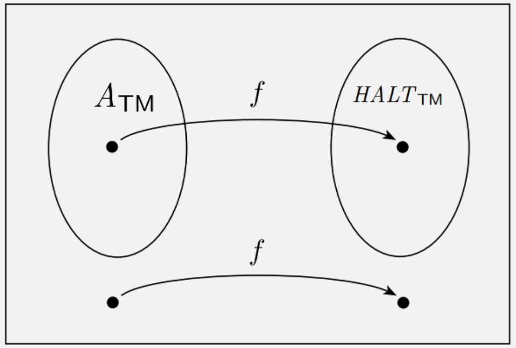
Thm: A_{TM} is mapping reducible to $HALT_{\mathsf{TM}}$

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

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

- To show: $A_{\mathsf{TM}} \leq_{\mathrm{m}} HALT_{\mathsf{TM}}$
- Want: computable fn f : $\langle M, w\rangle \rightarrow \langle M', w'\rangle$ where:

$$\langle M, w\rangle \in A_{\mathsf{TM}} \text{ if and only if } \langle M', w'\rangle \in HALT_{\mathsf{TM}}$$



The following machine F computes a reduction f.

$F =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on x.
   2. If M accepts, *accept*.
   3. If M rejects, enter a loop."
2. Output $\langle M', w\rangle$."

Converts M to M'

M accepts w

$\Leftrightarrow$

M' halts on w

Output new M'

DEFINITION 5.20

Language A is ***mapping reducible*** to language B, written $A \leq_{\mathrm{m}} B$, if there is a computable function $f\colon \Sigma^* \longrightarrow \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B.$$

The function f is called the ***reduction*** from A to B.

DEFINITION 5.17

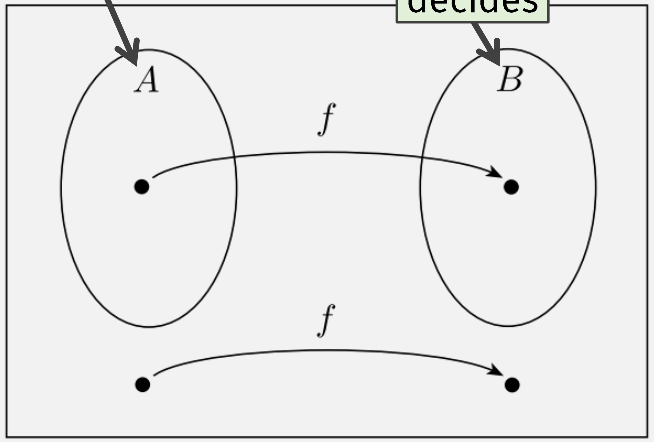
A function $f\colon \Sigma^* \longrightarrow \Sigma^*$ is a ***computable function*** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

How is mapping reducibility useful?

Thm: If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable.

PROOF We let M be the decider for B and f be the reduction from A to B. We describe a decider N for A as follows.

$N =$ "On input w:
1. Compute $f(w)$.
2. Run M on input $f(w)$ and output whatever M outputs."



DEFINITION 5.20

Language A is ***mapping reducible*** to language B, written $A \leq_{\mathrm{m}} B$, if there is a computable function $f : \Sigma^* \longrightarrow \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B.$$

The function f is called the ***reduction*** from A to B.

Coro: If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable.

- Proof by contradiction.
- Assume B is decidable.
- Then A is decidable (by the previous thm).
- So we have a contradiction.

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable.

Summary: Mapping Reducibility Theorems

- If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable.

Known   Unknown

- If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable.

Alternate Proof: The Halting Problem

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

- If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable.
- $A_{\mathsf{TM}} \leq_{\mathrm{m}} HALT_{\mathsf{TM}}$

Check-in Quiz 4/5

On gradescope