Mapping Reducibility

Monday, November 9, 2020

HW7 Questions?

Announcements

- No class next Wednesday Nov 11
- HW8 released early, due next Tues 11:59pm EST
  - (Normal schedule)

Last time: “Reduced” A_{TM} to $HALT_{\text{TM}}$

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

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

- Thm: $HALT_{\text{TM}}$ is undecidable
- Proof, by contradiction:
- Assume $HALT_{\text{TM}}$ 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:

1. Run TM R on input $\langle M, w \rangle$. ← Use R to first check if M will loop on w
2. If R rejects, *reject*.
3. If R accepts, simulate M on w until it halts. ← Then run M on w knowing it won’t loop
4. If M has accepted, *accept*; if M has rejected, *reject*.”

- But A_{TM} has no decider!
- Today: Formalize “reduction” and “reducibilty”

Computable Functions

- Instead of accept/reject, TM "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.

- Example 1: All arithmetic operations

- Example 2: Converting one TM to another
  - E.g., adding states, changing transitions, etc

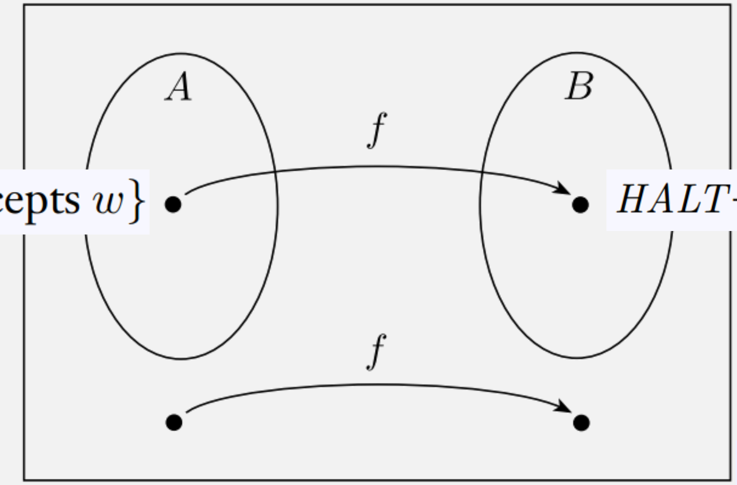
Mapping Reducibility

DEFINITION 5.20

Language A is ***mapping reducible*** to language B, written $A \leq_m B$, 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.



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

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

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.

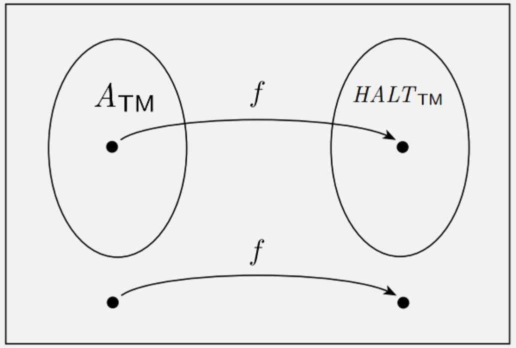
Thm: A_{TM} is mapping reducible to $HALT_{\mathsf{TM}}$

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

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

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

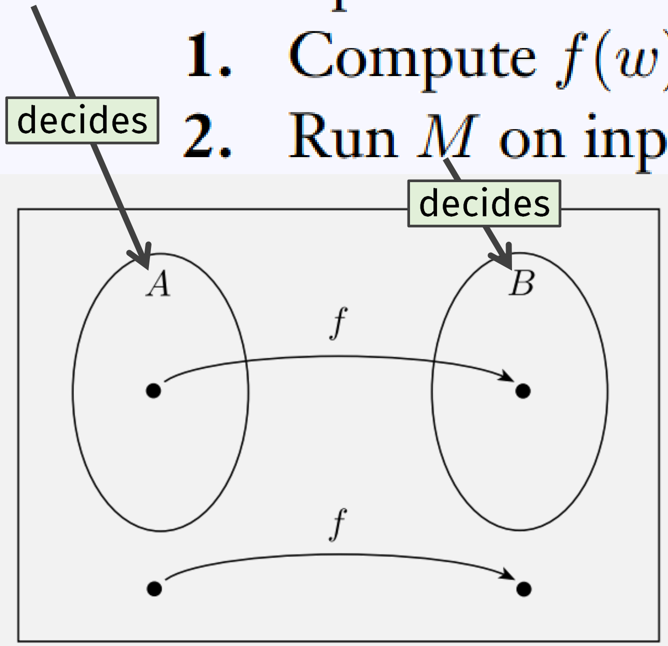
A function $f\colon \Sigma^* \longrightarrow \Sigma^*$ is a ***computable function*** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

Thm: If $A \leq_m B$ 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 $A \leq_m B$, 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 $A \leq_m B$ 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 $A \leq_m B$ and B is decidable, then A is decidable.

New Theorems: Summary

- 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.

- $A_{\mathsf{TM}} \leq_{\mathrm{m}} HALT_{\mathsf{TM}}$

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

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

$S =$ "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*."

- Alternate proof: Show: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$
- Computable fn f : $\langle M\rangle$ → $\langle M, M_1\rangle$

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.

Reducing to complement: E_{TM} is undecidable

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

- 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:
1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

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."

But *M1* does opposite

- Alternate: computable fn: $\langle M, w\rangle$ → $\langle M_1 \rangle$???
  - So this only reduces A_{TM} to $\overline{E_{\mathsf{TM}}}$
  - Still proves E_{TM} is undecidable
    - (HW8: undecidable langs closed under complement)

More Theorems

If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.

If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

- Same proofs as:

If $A \leq_m B$ and B is decidable, then A is decidable.

If $A \leq_m B$ and A is undecidable, then B is undecidable.

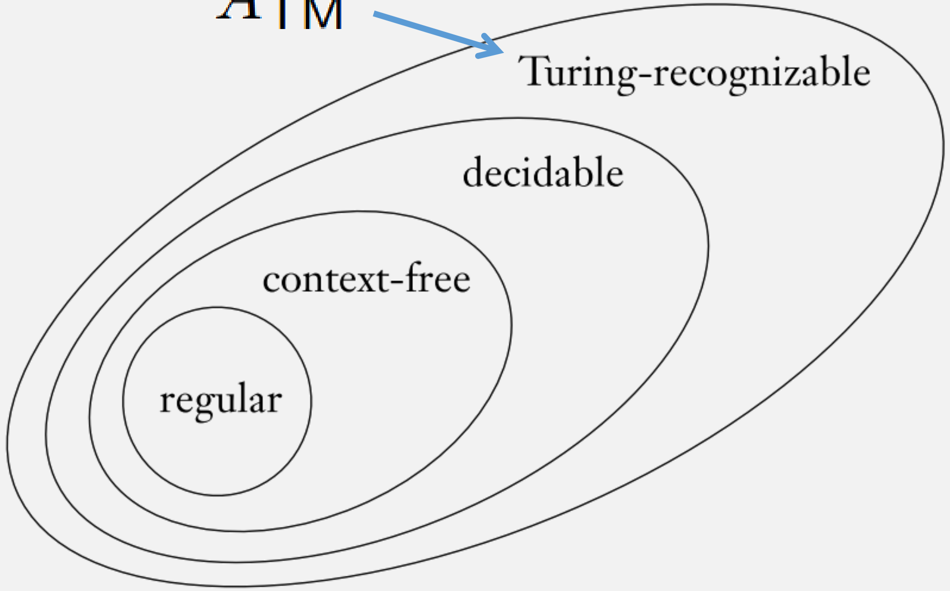
Thm: EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

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

1. EQ_{TM} is not Turing-recognizable

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

A_{TM}



$\overline{A_{\mathsf{TM}}} \leq_m EQ_{\mathsf{TM}}$ A is not Turing-recognizable, th EQ_{TM} not Turing-recognizable.

Review: Mapping Reducibility

DEFINITION 5.20

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

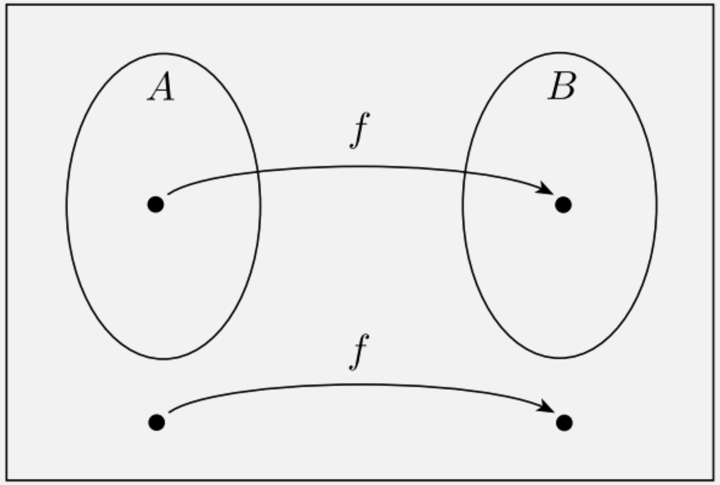
$$w \in A \iff f(w) \in B.$$

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

$A \leq_m B$

implies

$\overline{A} \leq_m \overline{B}$



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.

Thm: EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

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

1. EQ_{TM} is not Turing-recognizable

- Create Computable fn: $\overline{A_{\mathsf{TM}}} \rightarrow EQ_{\mathsf{TM}}$
- Or Computable fn: $A_{\mathsf{TM}} \rightarrow \overline{EQ_{\mathsf{TM}}}$

Thm: EQ_{TM} is not Turing-recognizable

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

- Create Computable fn: $A_{\mathsf{TM}} \rightarrow \overline{EQ_{\mathsf{TM}}}$
- $\langle M, w\rangle \rightarrow \langle M_1, M_2\rangle$ M_1 and M_2 are TMs and $L(M_1) \neq L(M_2)$

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

1. Construct the following two machines, M_1 and M_2.

   $M_1 =$ "On any input: ← Accepts nothing
   1. *Reject*."

   $M_2 =$ "On any input: ← Accepts nothing or everything
   1. Run M on w. If it accepts, *accept*."

2. Output $\langle M_1, M_2\rangle$."

- If M accepts w, M_1 not equal to M_2
- If M does not accept w, M_1 equal to M_2

<u>Thm</u>: EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

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

1. EQ_{TM} is not Turing-recognizable
   - Create <u>Computable fn</u>: $\overline{A_{\mathsf{TM}}} \rightarrow EQ_{\mathsf{TM}}$
   - Or <u>Computable fn</u>: $A_{\mathsf{TM}} \rightarrow \overline{EQ_{\mathsf{TM}}}$
   - **DONE!**

2. $\overline{EQ_{\mathsf{TM}}}$ is not ~~co~~-Turing-recognizable
   - (A lang is co-Turing-recog. if it is complement of Turing-recog. lang)

Thm: $\overline{EQ}_{\text{TM}}$ is not Turing-recognizable

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

- Create Computable fn: $A_{\text{TM}} \rightarrow \overline{EQ}_{\text{TM}}$
- $\langle M, w\rangle \rightarrow \langle M_1, M_2\rangle$ M_1 and M_2 are TMs and $L(M_1) \neq L(M_2)$

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

1. Construct the following two machines, M_1 and M_2.

   $M_1 =$ "On any input: ← Accepts ~~nothing~~ everything

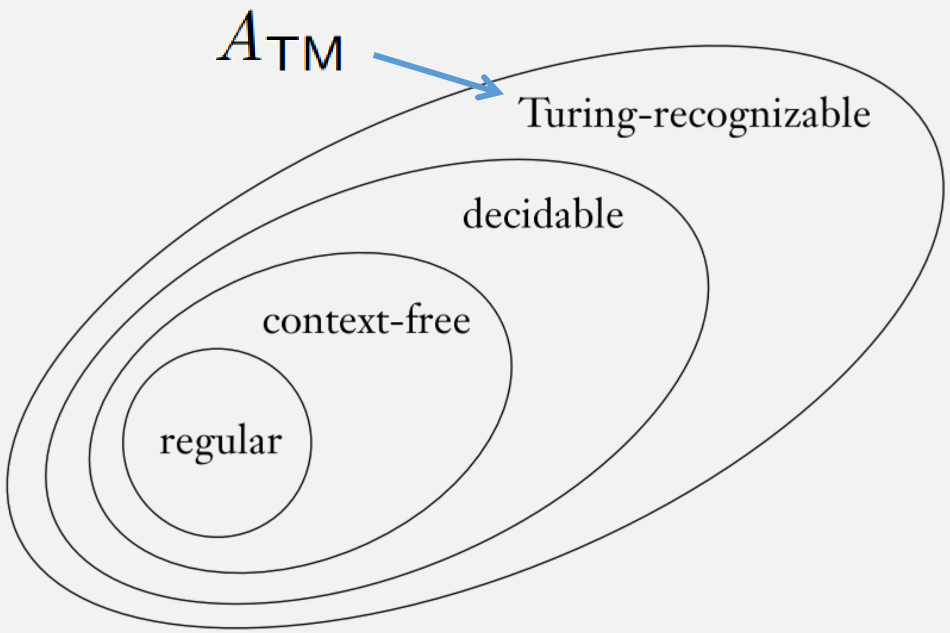
   1. *Accept*."

   $M_2 =$ "On any input: ← Accepts nothing or everything

   1. Run M on w. If it accepts, *accept*."

2. Output $\langle M_1, M_2\rangle$."

DONE!



EQ_{TM}

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

$\overline{EQ_{\mathsf{TM}}}$

Check-in Quiz 11/9

On gradescope

End of Class Survey 11/9

See course website