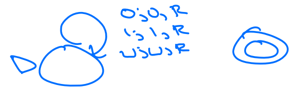
From Friday $A_{TM} \leq_m HALT_{TM}$ is witnessed by computable function given by Turing machine with high level description

"On input x

1. Check if $x = \langle M, w\rangle$ for some TM M, string w. If not, output $\langle$ 

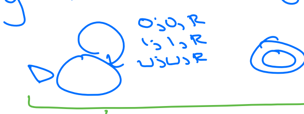
, $\varepsilon \rangle$
2. Build $M' =$ "On input y,
    1. Simulate M on y
    2. If accepts, accept. If rejects, loop."
3. Output $\langle M', w\rangle$"

Why does this function witness mapping reduction?

- Computable b/c computed by Turing machine above (that halts and output for each input)
- Consider arbitrary x.

Case 1: $x \in A_{TM}$. So $x = \langle M, w\rangle$ for some TM M, string w. Output of function is $\langle M', w\rangle$ with M' accepting (hence halting on) w. Thus output of function is in $HALT_{TM}$ ✓

Case 2. $x \notin A_{TM}$

Subcase 2a. $x \neq \langle M, w\rangle$ for any TM M, string w. So output is $\langle$ 

, $\varepsilon \rangle$ which is not in $HALT_{TM}$ b/c this TM loops on ε. ✓

Subcase 2b. $x = \langle M, w\rangle$ for some TM M, string w and M loops on w. Output of function is $\langle M', w\rangle$ where M''s computation on w loops in step 2 so M' does not halt on w ✓

Subcase 2c. $x = \langle M, w\rangle$ for some TM M, string w and M rejects w. Output of function is $\langle M', w\rangle$ where M''s computation on w loops in step 3 so M' does not halt on w ✓

Monday - Memorial Day

No class today.

Wednesday

Recall: The class of Turing recognizable languages is not closed under complementation.

Recall: A is **mapping reducible to** B, written $A \leq_m B$, means there is a computable function $f : \Sigma^* \to \Sigma^*$ such that *for all* strings x in Σ^*,

$$\boxed{x \in A \qquad \text{if and only if} \qquad f(x) \in B.}$$

True or False: $\overline{A_{TM}} \leq_m \overline{HALT_{TM}}$

want a function that is ① computable ② for each $x \in \Sigma^*$, $x \in \overline{A_{TM}}$ iff $f(x) \in \overline{HALT_{TM}}$

$x \notin A_{TM}$ iff $f(x) \notin HALT_{TM}$

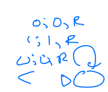
witness function for $A_{TM} \leq_m HALT_{TM}$ also witnesses that $\overline{A_{TM}} \leq_m \overline{HALT_{TM}}$

True or False: $HALT_{TM} \leq_m A_{TM}$.

BUT can't use the same function! Why?

Goal $\langle M, w \rangle$ with M halts on w $\rightsquigarrow \langle M', w \rangle$ M' accepts w

Consider function computed by TM with high level description: "On input x

1. If $x \neq \langle M, w \rangle$ M TM, w string, output $\langle$  $, \varepsilon \rangle$
2. Otherwise, $x = \langle M, w \rangle$ for some M TM, w string and define M' = "On input y
   1. Run M on y
   2. If M accepts, accept. If M rejects, accept."
3. Output $\langle M', w \rangle$ "

To do: prove this works ...

Theorem (Sipser 5.28): If $A \leq_m B$ and B is recognizable, then A is recognizable.

Proof:

extra practice

Corollary: If $A \leq_m B$ and A is unrecognizable, then B is unrecognizable.

To witness $HALT_{TM} \leq_m A_{TM}$

a function on Σ^* needs to be computable and for each x

Case ① if $x \neq \langle M, w \rangle$ for any TM M, string w

output of function must not be in A_{TM}.

Case ② If $x = \langle M, w \rangle$ for some TM M, string w

and M halts on w then output

of function must be $\langle \underbrace{\quad\quad}_{TM}, \underbrace{\quad\quad}_{string} \rangle$

where this TM accepts this string.

Case ③ If $x = \langle M, w \rangle$ for some TM M, string w

and M loops on w then output

of function must <u>not</u> be in A_{TM}.

if $X \leq_m Y$ and X is undecidable then so is Y.

Strategy:

(i) To prove that a recognizable language R is undecidable, prove that $A_{TM} \leq_m R$.

(ii) To prove that a co-recognizable language U is undecidable, prove that $\overline{A_{TM}} \leq_m U$, i.e. that $A_{TM} \leq_m \overline{U}$.

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

Example string in E_{TM} is $\langle$ $\rangle$. Example string not in E_{TM} is $\langle M_1, M_2 \rangle$. or $\langle$ $\rangle$

E_{TM} is decidable / undecidable and recognizable / unrecognizable.

$\overline{E_{TM}}$ is decidable / undecidable and recognizable / unrecognizable. informally, need to prove.

Claim: A_{TM} $\leq_m \overline{E_{TM}}$. i.e. $\overline{A_{TM}} \leq_m E_{TM}$

Proof: Need computable function $F : \Sigma^* \to \Sigma^*$ such that $x \in A_{TM}$ iff $F(x) \notin E_{TM}$. Define

$F =$ " On input x,

1. Type-check whether $x = \langle M, w \rangle$ for some TM M and string w. If so, move to step 2; if not, output $\langle$ $\rangle$
2. Construct the following machine M'_x:

   $M'_x =$ " On input y
   1. Ignore input.
   2. Run M on w
   3. If M accepts w, accept y.
   4. If M rejects w, reject y. "
3. Output $\langle M'_x \rangle$."

Verifying correctness:

Input string	Output string
$x \in A_{TM}$ $\langle M, w \rangle$ where $w \in L(M)$	$F(x) \notin E_{TM}$? $L(M'_x) = \Sigma^*$ ✓
$x \notin A_{TM}$ $\langle M, w \rangle$ where $w \notin L(M)$	$F(x) \in E_{TM}$? $L(M'_x) = \emptyset$ ✓
x not encoding any pair of TM and string	$F(x) \in E_{TM}$? Yes b/c $L(\langle \quad \rangle) = \emptyset$ ✓

Review: Week 9 Wednesday

Please complete the review quiz questions on Gradescope about mapping reductions.

Pre class reading for next time: Introduction to Chapter 7.

Reference undecidable sets:
A_{TM}, $\overline{A_{TM}}$, $HALT_{TM}$, $\overline{HALT_{TM}}$, E_{TM}, $\overline{E_{TM}}$

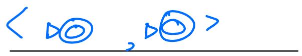
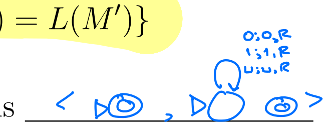
Reference unrecognizable sets:
$\overline{A_{TM}}$, $\overline{HALT_{TM}}$, E_{TM}.

Friday

Recall: A is **mapping reducible to** B, written $A \leq_m B$, means there is a computable function $f : \Sigma^* \to \Sigma^*$ such that *for all* strings x in Σ^*,

$$x \in A \qquad \text{if and only if} \qquad f(x) \in B.$$

$$EQ_{TM} = \{\langle M, M' \rangle \mid M \text{ and } M' \text{ are both Turing machines and } L(M) = L(M')\}$$

Example string in EQ_{TM} is  . Example string not in EQ_{TM} is .

Best guess

EQ_{TM} is decidable / undecidable and recognizable / unrecognizable .

$\overline{EQ_{TM}}$ is decidable / undecidable and recognizable / unrecognizable .

To prove, show that $\overline{HALT_{TM}}$ $\leq_m EQ_{TM}$ and that $\overline{HALT_{TM}}$ $\leq_m \overline{EQ_{TM}}$.

since $\overline{HALT_{TM}}$ is unrecognizable, EQ_{TM} will be too

since $\overline{HALT_{TM}}$ is unrecognizable, $\overline{EQ_{TM}}$ will be too.

Using the complementation strategy from before, it's equivalent to prove that

1 $\boxed{HALT_{TM} \leq_m \overline{EQ_{TM}}}$ and 2 $HALT_{TM} \leq_m EQ_{TM}$

When we define the witnessing function for each reduction

Verifying correctness will mean:

1

Input string	Output string	
$x \in HALT_{TM}$ $\langle M, w \rangle$ where M halts on w	$f(x) \notin EQ_{TM}$!	✓
$x \notin HALT_{TM}$ $\langle M, w \rangle$ where M loops on w	$f(x) \in EQ_{TM}$!	✓
x not encoding any pair of TM and string	$f(x) \in EQ_{TM}$!	✓

CAN'T DISTINGUISH ALGORITHMICALLY

CAN CHECK!

Now we need to define the function that will achieve these goals.

Define function as that computed by the following Turing machine

"On input x:

1. If $x \neq \langle M, w \rangle$ for any Turing machine M, string w,

   output $\langle$, $\rangle$ ← note that this output is in EQ_{TM}.

2. If $x = \langle M, w \rangle$ for some Turing machine M, string w,

   Build new Turing machine

   M'_x = "On input y: (no type check!)

   $x = \langle M, w \rangle$

   1. If $y \neq w$, reject.
   2. Else, run M on w
   3. If M accepts w, accept.
   4. If M rejects w, accept"

   (M halts on w)

3. Output $\langle M'_x$,  $\rangle$ "

What is $L(M'_x)$?

Is $w \in L(M'_x)$ iff M halts on w ?

Notice that

If $\langle M, w \rangle \in HALT_{TM}$ then $L(M'_x) = \{w\}$ which is not equal to $L($ $)$

If $\langle M, w \rangle \notin HALT_{TM}$ then $L(M'_x) = \emptyset$ which is equal to $L($ $)$

i.e. $X \in HALT_{TM}$ gives $f(x) \notin EQ_{TM}$

$X \notin HALT_{TM}$ gives $f(x) \in EQ_{TM}$.

② To prove $HALT_{TM} \leq_m EQ_{TM}$

Need a computable witnessing function, let's call it $g: \Sigma^* \to \Sigma^*$, with

IF $x \in HALT_{TM}$ THEN $g(x) \in EQ_{TM}$
$x = \langle M, w\rangle$
M halts on w

IF $x \notin HALT_{TM}$, THEN $g(x) \notin EQ_{TM}$
$x = \langle M, w\rangle$
M loops on w

IF $x \neq \langle M, w\rangle$ for any M, w THEN $g(x) \notin EQ_{TM}$

Define function as that computed by the following Turing machine

"On input x:

1. If $x \neq \langle M, w\rangle$ for any Turing machine M, string w,

   output ⟨ ▷◎, 

   ⟩

note that this output is EQTM.

2. If $x = \langle M, w\rangle$ for some Turing machine M, string w,

   Build new Turing machine

   M'_x = "On input y. (no type check!)
   1. If $y \neq \underline{w}$, accept
   2. Else, run M on w
   3. If M accepts w, accept.
   4. If M rejects w, accept"

   $x = \langle M, w\rangle$

   (M halts on w)

3. Output ⟨ M'_x, ▷◎ ⟩"

Verifying:

IF $x \in HALT_{TM}$ THEN $g(x) \in EQ_{TM}$?

$x = \langle M, w \rangle$

M halts on w

In this case, $g(x) = \langle M'_x, \triangleright\circledcirc \rangle$ with M'_x accepting all strings : $L(M'_x) = \Sigma^* = L(\triangleright\circledcirc)$

so $g(x) \in EQ_{TM}$ ✓

IF $x \notin HALT_{TM}$, THEN $g(x) \notin EQ_{TM}$?

$x = \langle M, w \rangle$

M loops on w

In this case $g(x) = \langle M'_x, \triangleright\circledcirc \rangle$ with M'_x accepting all strings except w: $L(M'_x) \neq \Sigma^* = L(\triangleright\circledcirc)$

so $g(x) \notin EQ_{TM}$. ✓

IF $x \neq \langle M, w \rangle$ for any M, w THEN $g(x) \notin EQ_{TM}$?

In this case, $g(x) = \langle \triangleright\circledcirc , \triangleright\bigcirc\circlearrowleft^{0;0,R}_{1;1,R \; \sqcup;\sqcup,R} \; \circledcirc \rangle$

and $L(\triangleright\circledcirc) = \Sigma^* \neq \emptyset = L(\triangleright\bigcirc\circlearrowleft^{0;0,R}_{1;1,R \; \sqcup;\sqcup,R} \; \circledcirc)$

So $g(x) \notin EQ_{TM}$. ✓

Monday

In practice, computers (and Turing machines) don't have infinite tape, and we can't afford to wait unboundedly long for an answer. "Decidable" isn't good enough - we want "Efficiently decidable".

For a given algorithm working on a given input, how long do we need to wait for an answer? How does the running time depend on the input in the worst-case? average-case? We expect to have to spend more time on computations with larger inputs.

A language is **recognizable** if ____

A language is **decidable** if ____

A language is **efficiently decidable** if _______________________________________

A function is **computable** if ____

A function is **efficiently computable** if _______________________________________

Definition (Sipser 7.1): For M a deterministic decider, its **running time** is the function $f : \mathbb{N} \to \mathbb{N}$ given by

$$f(n) = \text{max number of steps } M \text{ takes before halting, over all inputs of length } n$$

Definition (Sipser 7.7): For each function $t(n)$, the **time complexity class** $TIME(t(n))$, is defined by

$$TIME(t(n)) = \{L \mid L \text{ is decidable by a Turing machine with running time in } O(t(n))\}$$

An example of an element of $TIME(1)$ is

An example of an element of $TIME(n)$ is

Note: $TIME(1) \subseteq TIME(n) \subseteq TIME(n^2)$

Definition (Sipser 7.12) : P is the class of languages that are decidable in polynomial time on a deterministic 1-tape Turing machine

$$P = \bigcup_k TIME(n^k)$$

Compare to exponential time: brute-force search.

Theorem (Sipser 7.8): Let $t(n)$ be a function with $t(n) \geq n$. Then every $t(n)$ time deterministic multitape Turing machine has an equivalent $O(t^2(n))$ time deterministic 1-tape Turing machine.

Review: Week 9 Friday

Please complete the review quiz questions on Gradescope about complexity.

Pre class reading for next time: Skim Chapter 7.

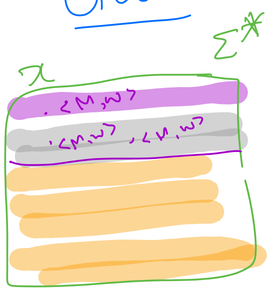
BONUS EXAMPLE

Claim: $A_{TM} \leq_m \mathcal{E}_{TM} = \{ \langle M \rangle \mid M \text{ is TM, } M \text{ accepts } \varepsilon \}$

Pf: Need witness function $f: \Sigma^* \to \Sigma^*$, computable

and $x \in A_{TM}$ iff $f(x) \in \mathcal{E}_{TM}$, for all x

Goal



① $x \in A_{TM}$ ie $x = \langle M, w \rangle$ M accepts w
WANT $f(x) = \langle M'_x \rangle$ M'_x accepts ε. ✓

② $x \notin A_{TM}$ where $x = \langle M, w \rangle$ M doesn't accept w
$f(x) = \langle M'_x \rangle$ $\langle M'_x \rangle \notin \mathcal{E}_{TM}$ ✓
WANT $f(x) \notin \mathcal{E}_{TM}$

③ $x \notin A_{TM}$ b/c $x \neq \langle M, w \rangle$ for any TM M, string w
WANT $f(x) \notin \mathcal{E}_{TM}$

Define f to be computed by TM

M_f = "On input x — local variable representing input
1. If $x \neq \langle M, w \rangle$ for any TM M, string w
output $\langle$ ▷ 0;0,R / 1;1,R / ⊔;⊔,R ◎ $\rangle$
2. Otherwise $x = \langle M, w \rangle$
3. Build M'_x = "On input y — local var representing input
1. Run M on w.
2. If M accepts, accept
3. If M rejects, reject"
4. Output $\langle M'_x \rangle$"

Does M'_x accept ε? Yes if M accepts w
(b/c then $L(M'_x) = \Sigma^*$)

No if M doesn't accept w
(b/c then $L(M'_x) = \emptyset$)