CS420

Reducibility

Monday, March 28, 2022

Announcements

- HW 7 in
  - ~~Due Sun 3/27 11:59pm~~
- HW 8 out
  - Due Sun 4/3 11:59pm

Last Time: 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:
  - **Use hypothetical** A_{TM} **decider to** create an <u>impossible</u> **decider "*D*"!**

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

- Step # 1: **coming up with "*D*" --- <u>hard!</u>**
  - Need to invent **diagonalization**

- Step # 2: **"reduce"** A_{TM} **to the "*D*" problem ---** <u>easier</u>**!**

- From now on: **undecidability proofs <u>only need to do</u>** <u>step # 2</u>**!**
  - And we now **have two "impossible" problems to choose from**

Last Time: 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:

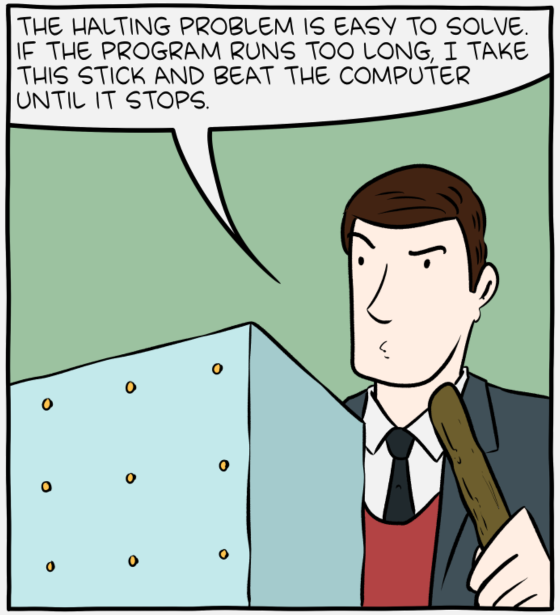
- **Assume:** $HALT_{\mathsf{TM}}$ has *decider* R; use it to create decider for A_{TM}:

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

- ...

contradiction

- But A_{TM} is undecidable and has no decider!



What if Alan Turing had been an engineer?

Last Time: 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:

Using our hypothetical $HALT_{\mathsf{TM}}$ decider R

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

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

$S =$ "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*."

Termination argument:
Step 1: R is a decider so always halts
Step 3: M always halts because R said so

Undecidability Proof Technique #1: Reduce from A_{TM}

Last Time: The Halting Problem

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

Thm: $HALT_{TM}$ is undecidable

Proof, by contradiction:

- **Assume:** $HALT_{TM}$ has *decider* R; use it to create decider for A_{TM}:

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

  ~~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 does not exist!
  - So $HALT_{TM}$ is also undecidable!

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**
- $HALT_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ — **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
- today: $E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — **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 it to create *decider* for A_{TM}:

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_1 is only used as arg to R; we never run it!
- 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*. ← Input not w, always reject
2. If $x = w$, run M on input w and *accept* if M does."

Input is w, maybe accept

M_1 accepts w 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 for A_{TM} cannot exist!

- Assume E_{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:

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."

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**
- $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ — Decidable
- $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ — **Undecidable**
- $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ — **Undecidable**

needs

next

Reduce to something else: EQ_{TM} is undecidable

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

Proof, by contradiction:

- Assume: EQ_{TM} has *decider* R; use it to create *decider* for ~~A_{TM}~~:

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

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*."

Reduce to something else: EQ_{TM} is undecidable

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

Proof, by contradiction:

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

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

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*."

- But E_{TM} is undecidable!

Summary: Undecidability Proof Techniques

- **Proof Technique #1:** $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
  - Use hypothetical decider to implement impossible A_{TM} decider
  - Example **Proof:** $HALT_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

Reduce

- **Proof Technique #2:**
  - Use hypothetical decider to implement impossible A_{TM} decider
  - But first modify the input M
  - Example **Proof:** $E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$

Reduce

- **Proof Technique #3:**
  - Use hypothetical decider to implement non-A_{TM} impossible decider
  - Example **Proof:** $EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Summary: Decidability and Undecidability

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

Also Undecidable …

next

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

Undecidability Proof Technique #2: Modify input TM M

Thm: $REGULAR_{\mathsf{TM}}$ 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 it to create *decider* for A_{TM}:

$S =$ "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: $REGULAR_{\text{TM}}$ is undecidable (continued)

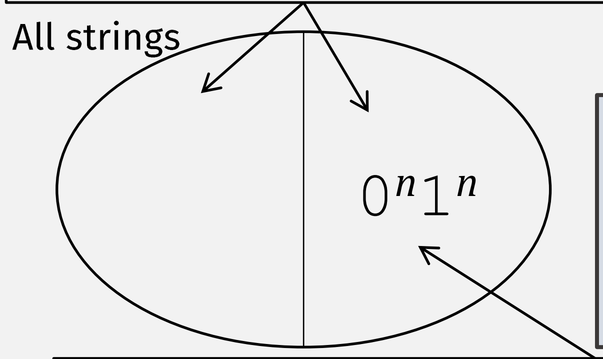
$REGULAR_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

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 does not accept w, M_2 accepts all strings (regular lang)



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

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

Also Undecidable …

Seems like no algorithm can compute **anything** about language of TMs, i.e., about programs!

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

An Algorithm About Program Behavior?

```
main()
{
    printf("hello, world\n");
}
```

Write a program that,
given another program as its argument,
returns TRUE if that argument prints
“**`Hello, World!`**”

TRUE

Fermat's Last Theorem
(unknown for ~350 years, solved in 1990s)

```
main()
{
  If x^n + y^n = z^n, for any integer n > 2
    printf("hello, world\n");
}
```

Write a program that,
given another program as its argument,
returns TRUE if that argument prints
"`Hello, World!`"

?????

Also Undecidable …

Seems like no algorithm can compute **anything** about Turing Machines, i.e., about programs!

- $REGULAR_{TM}$ = {<M> | M is a TM and $L(M)$ is a regular language}
- $CONTEXTFREE_{TM}$ = {<M> | M is a TM and $L(M)$ is a CFL}
- $DECIDABLE_{TM}$ = {<M> | M is a TM and $L(M)$ is a decidable language}
- $FINITE_{TM}$ = {<M> | M is a TM and $L(M)$ is a finite language}
- …
- $ANYTHING_{TM}$ = {<M> | M is a TM and "… **anything** …" about $L(M)$}

Rice's Theorem

Rice's Theorem: $ANYTHING_{\text{TM}}$ is Undecidable

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

- "… **Anything** …", more precisely:
  - For any M_1, M_2, if $L(M_1) = L(M_2)$ …
  - … then $M_1 \in ANYTHING_{\text{TM}} \Leftrightarrow M_2 \in ANYTHING_{\text{TM}}$

- Also, "… **Anything** …"must be "non-trivial":
  - $ANYTHING_{\text{TM}}$!= {}
  - $ANYTHING_{\text{TM}}$!= set of all TMs

Rice's Theorem: $ANYTHING_{\text{TM}}$ is Undecidable

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

Proof by contradiction

- Assume some language satisfying $ANYTHING_{\text{TM}}$ has a decider R.
  - Since $ANYTHING_{\text{TM}}$ is non-trivial, then there exists $M_{ANY} \in ANYTHING_{\text{TM}}$
  - Where R accepts M_{ANY}
- Use R to create decider for A_{TM}:

On input <M, w>:

- Create M_w:

  M_w = on input x:
  - Run M on w
    - If M rejects w: reject x
    - If M accepts w:
      Run M_{ANY} on x and accept if it accepts, else reject

  If M accepts w: $M_w = M_{\text{ANY}}$
  If M doesn't accept w: M_w accepts nothing

  These two cases must be different, (so R can distinguish when M accepts w)

- Run R on M_w
  - If it accepts, then $M_w = M_{ANY}$, so M accepts w, so accept
  - Else reject

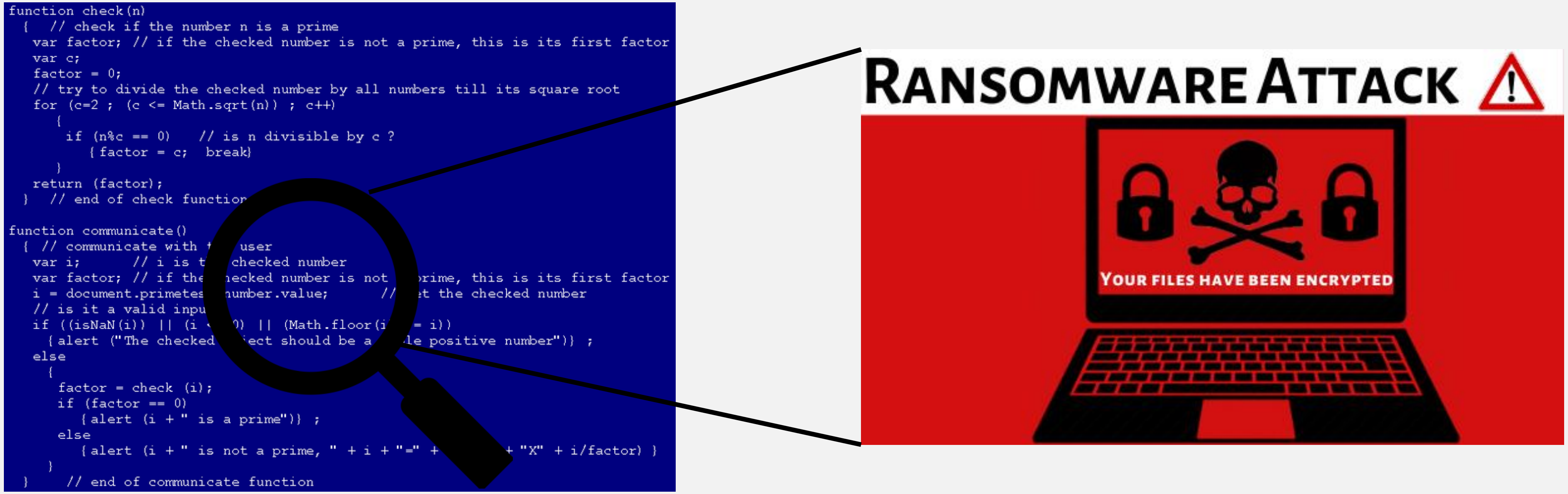
Wait! What if the TM that accepts nothing is in $ANYTHING_{\text{TM}}$!

Proof still works! Just use the complement of $ANYTHING_{\text{TM}}$ instead!

Rice’s Theorem Implication

{<M> | M is a TM that installs malware}

Undecidable!
(by Rice’s Theorem)



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

- In hindsight, of course a restricted TM (a **decider**) shouldn't be able to simulate unrestricted TM (a **recognizer**)
- But could a restricted TM simulate an even more restricted TM?
  - Next time

Check-in Quiz 3/28

On gradescope