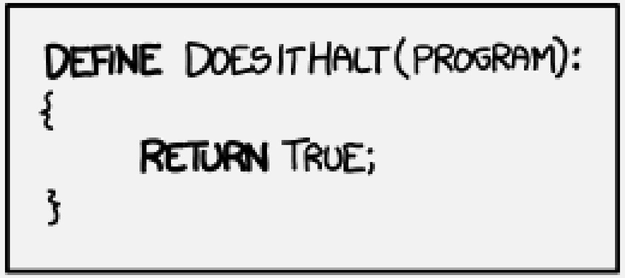
# CS622

# More Undecidability

Monday, October 25, 2021




THE BIG PICTURE SOLUTION
TO THE HALTING PROBLEM

# Announcements

- Hw5 in

- Hw6 out
  - Due Sunday 10/24 11:59pm EST

- Hw4 grades returned

# *Last Time:* The Limits of Algorithms

- $A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ Decidable
- $E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ Decidable
- $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ Decidable

# *Last Time:* 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

- $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ Decidable
- $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ Decidable

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

# *Last Time:* 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
- TBD $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ **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**

# No Algorithms About Language of TMs

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

guage about
emantics" of
undecidable

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

$ANYTHING_{\mathsf{TM}}$ = {<$M$> | $M$ 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 $M_1 \in ANYTHING_{\mathsf{TM}} \Leftrightarrow M_2 \in ANYTHING_{\mathsf{TM}}$

- Also, anything must be "non-trivial":
  - $ANYTHING_{\mathsf{TM}}$ != {}
  - $ANYTHING_{\mathsf{TM}}$ != 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 lang 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 $A_{\text{TM}}$:

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

- 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! (see hw5: complement closed for decidable languages)

# Rice's Theorem Real-World Example

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

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

TRUE

# Rice's Theorem Example

Fermat's Last Theorem

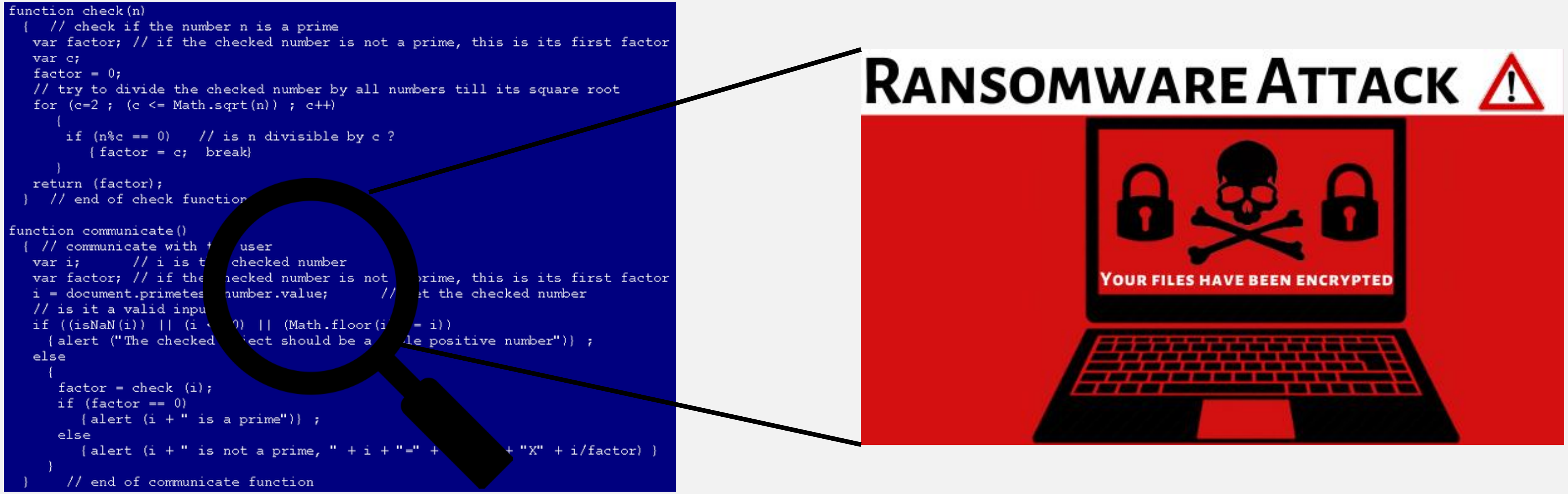
```
main()
{
  If x^n + y^n = z^n, for any integer n > 2
    printf("hello, world\n");
}
```

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

**?????**

$\{<M> \mid M$ 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

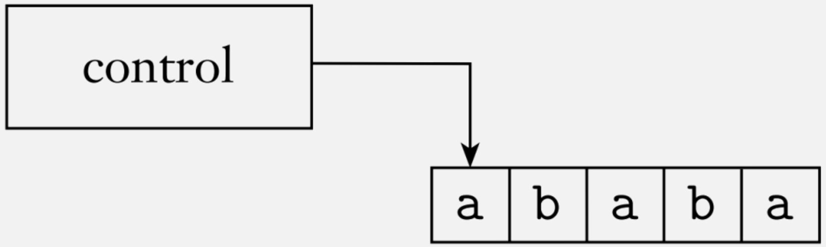
$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ **Undecidable**

- 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?

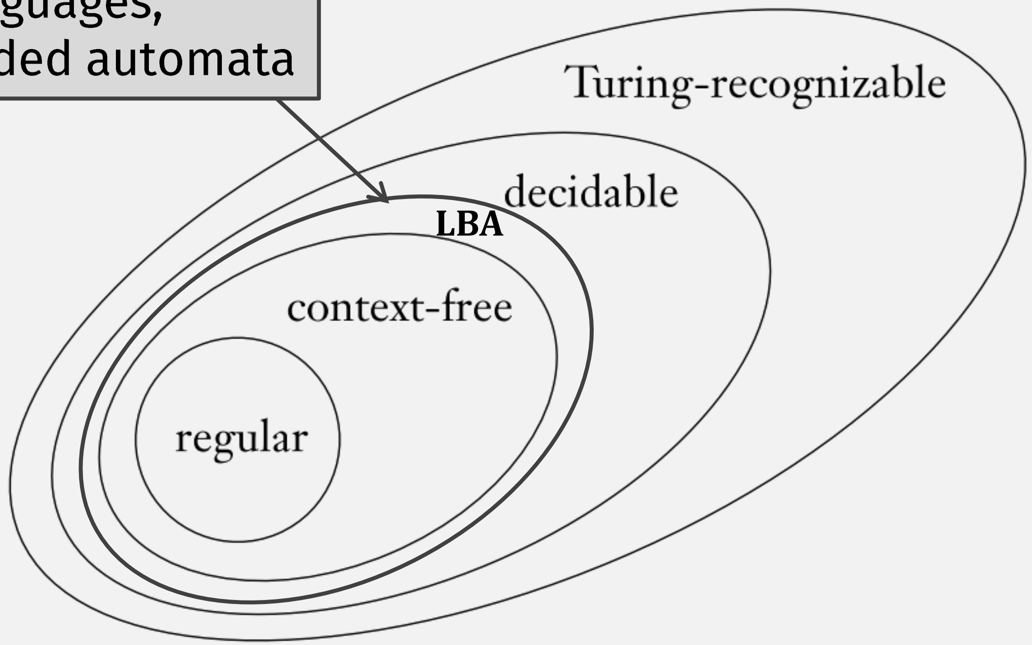
# Linear Bounded Automata

A ***linear bounded automaton*** is a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is—in the same way that the head will not move off the left-hand end of an ordinary Turing machine's tape.



# Context-Sensitive Languages

context-sensitive languages,
recognized by linear bounded automata



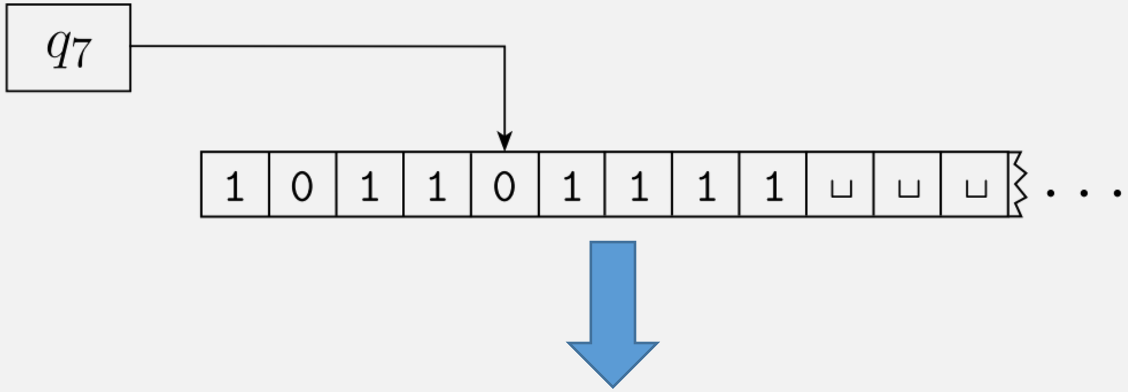

What exactly does it mean to be **context-free vs context-sensitive**?

Chomsky Hierarchy

# Theorem: $A_{\mathsf{LBA}}$ is decidable

$A_{\mathsf{LBA}} = \{\langle M, w\rangle \mid M$ is an LBA that accepts string $w\}$

# *Flashback:* TM Configuration = State + Head + Tape




$$1011q_701111$$

Textual representation of “configuration”

1st char after state is current head position

# How Many Possible Configurations …

- Does an LBA have?
  - $q$ states
  - $g$ tape alphabet chars
  - tape of length $n$

- Possible Configurations = $qng^n$
  - $g^n$ = number of possible tape configurations
  - $qn$ = all the possible head positions

# Theorem: $A_{\mathsf{LBA}}$ is decidable

$A_{\mathsf{LBA}} = \{\langle M, w\rangle |\ M \text{ is an LBA that accepts string } w\}$

Proof: Create decider for $A_{\mathsf{LBA}}$

On input $<M, w>$:

- Simulate $M$ on $w$.
- If $M$ accepts $w$, then accept.
- If $M$ runs $> qng^n$ steps then we are in a loop so halt and reject

Termination argument?

# Theorem: $E_{\mathsf{LBA}}$ is undecidable

$E_{\mathsf{LBA}} = \{\langle M \rangle |\ M \text{ is an } \mathsf{LBA} \text{ where } L(M) = \emptyset\}$

# *Flashback:* TM Configuration Sequences

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

**Single-step**

head

Next config

(Right) $\alpha q_1 \mathbf{a} \beta \vdash \alpha \mathbf{x} q_2 \beta$

write

if $q_1, q_2 \in Q$

$\delta(q_1, \mathbf{a}) = (q_2, \mathbf{x}, \mathrm{R})$

read

$\mathbf{a}, \mathbf{x} \in \Gamma \quad \alpha, \beta \in \Gamma^*$

(Left) $\alpha b q_1 \mathbf{a} \beta \vdash \alpha q_2 b \mathbf{x} \beta$

if $\delta(q_1, \mathbf{a}) = (q_2, \mathbf{x}, \mathrm{L})$

**Extended**

- Base Case

$$\boxed{I \overset{*}{\vdash} I \text{ for any ID } I}$$

- Recursive Case

$\boxed{I \overset{*}{\vdash} J}$ if there exists some ID $K$

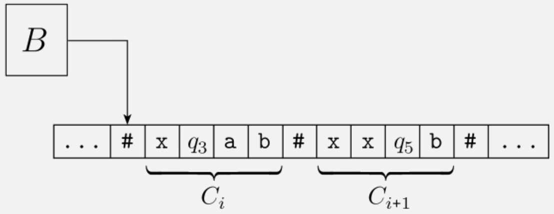
such that $I \vdash K$ and $K \overset{*}{\vdash} J$

# Theorem: $E_{\mathsf{LBA}}$ is undecidable

$E_{\mathsf{LBA}} = \{\langle M\rangle \mid M \text{ is an LBA where } L(M) = \emptyset\}$

Proof, by contradiction:

- Assume $E_{\mathsf{LBA}}$ has *decider* $R$; use to create decider for $A_{\mathsf{TM}}$:
- On input <$M$, $w$>, where $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:
  - Construct LBA $B$:
    - $B$ accepts sequences of $M$ configurations where $M$ accepts $w$, i.e.,
      - First configuration is $q_0 w_1 w_2 \cdots w_n$
      - Last configuration has state $q_{accept}$
      - Each pair of adjacent configs is valid according to $M$'s $\delta$
  - Run $R$ with $B$ as input:
    - If $R$ accepts $B$, then $B$'s language is empty
      - So there's no sequence of $M$ configs that accept $w$, so reject
    - If $R$ rejects $B$, then $B$'s language is not empty
      - So there's a sequence of $M$ configs that accepts $w$, so accept




**Wait!** So any language that can be used to check computation histories must be undecidable

# Theorem: $ALL_{\text{CFG}}$ is undecidable

$ALL_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$

Proof, by contradiction

- Assume $ALL_{\text{CFG}}$ has a decider $R$. Use it to create decider for $A_{\text{TM}}$:

On input <$M$, $w$>:

Can a PDA do this?

- Construct a $\boxed{\text{PDA } P}$ that rejects sequences of $M$ configs that accept $w$
- Convert $P$ to a CFG $G$ (previous class)
- Give $G$ to $R$:
  - If $R$ accepts, then $M$ has no accepting config sequences for $w$, so reject
  - If $R$ rejects, then $M$ has an accepting config sequence for $w$, so accept

# A PDA That Rejects TM $M$ Config Sequences

$$\#\underbrace{\;\longrightarrow\;}_{C_1}\#\underbrace{\;\longleftarrow\;}_{C_2^{\mathcal{R}}}\#\underbrace{\;\longrightarrow\;}_{C_3}\#\underbrace{\;\longleftarrow\;}_{C_4^{\mathcal{R}}}\#\;\cdots\;\#\underbrace{\qquad}_{C_l}\#$$

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

On input $\cancel{\#\underbrace{\qquad}_{C_1}\#\underbrace{\qquad}_{C_2}\#\underbrace{\qquad}_{C_3}\#\cdots\#\underbrace{\qquad}_{C_l}\#}$, nondeterministically:

- Reject if $C_1$ is not $q_0 w_1 w_2 \cdots w_n$
- Reject if $C_l$ does not have $q_{accept}$
- Reject if any $C_i$ and $C_{i+1}$ is invalid according to δ:
  - Push $C_i$ onto the stack
  - Compare $C_i$ with $C_{i+1}$ (reversed):
    - Check that initial chars match
    - On first non-matching char, check that next 3 chars is valid ac
      - Each possible δ can be hard-coded since δ is finite
    - Continue checking remaining chars
    - Reject whenever anything is invalid

Why **reject** accepting configuration sequences?

Could we create a PDA that **accepts** accepting configuration sequences?

But that would mean $E_{\text{CFG}}$ is undecidable??

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

We already proved this is **decidable!**

# Algorithms For CFLs

- $A_{\text{CFG}} = \{\langle G, w\rangle |\ G \text{ is a CFG that generates string } w\}$ Decidable
- $E_{\text{CFG}} = \{\langle G\rangle |\ G \text{ is a CFG and } L(G) = \emptyset\}$ ← Already proved this is **decidable** Decidable
- $ALL_{\text{CFG}} = \{\langle G\rangle |\ G \text{ is a CFG and } L(G) = \Sigma^*\}$ ← Just proved this is **undecidable** **Undecidable**

# Exploring the Limits of CFLs

- This is a CFL: $\{w_1\#w_2 \mid w_1 \neq w_2\}$
  - PDA nondeterministically checks matching positions in $1^{st}/2^{nd}$ parts
  - And rejects if **<u>any</u>** are not the same
  - I.e., Each branch is "context free"

This is similar to the <u>config-rejecting PDA</u>

- This is <u>not</u> a CFL: $\{w_1\#w_2 \mid w_1 = w_2\}$
  - Can nondeterministically check matching positions
  - But needs to accept only if **<u>all</u>** branches match
  - I.e., each branch is <u>not</u> "context free"

This is similar to the *ww* language (not pumpable)

An <u>config-accepting PDA</u> would be like this language … i.e., not a CFL!

**Summary**: CFLs cannot do (stack-based) nondet. computation where a branch <u>depends</u> on other branch results

(This is also why **union is closed for CFLs** but **intersection is not**)

# Algorithms For CFLs

- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ ← Already proved this is **decidable** — Decidable
- $ALL_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$ ← Just proved this is **undecidable** — **Undecidable**
- $EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ — **Undecidable?**

(Still need to prove this is undecidable)

# Theorem: $EQ_{\text{CFG}}$ is undecidable

$EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

- Proof by contradiction: Assume $EQ_{\text{CFG}}$ has a decider $R$
- Use $R$ to create a decider for $ALL_{\text{CFG}}$:

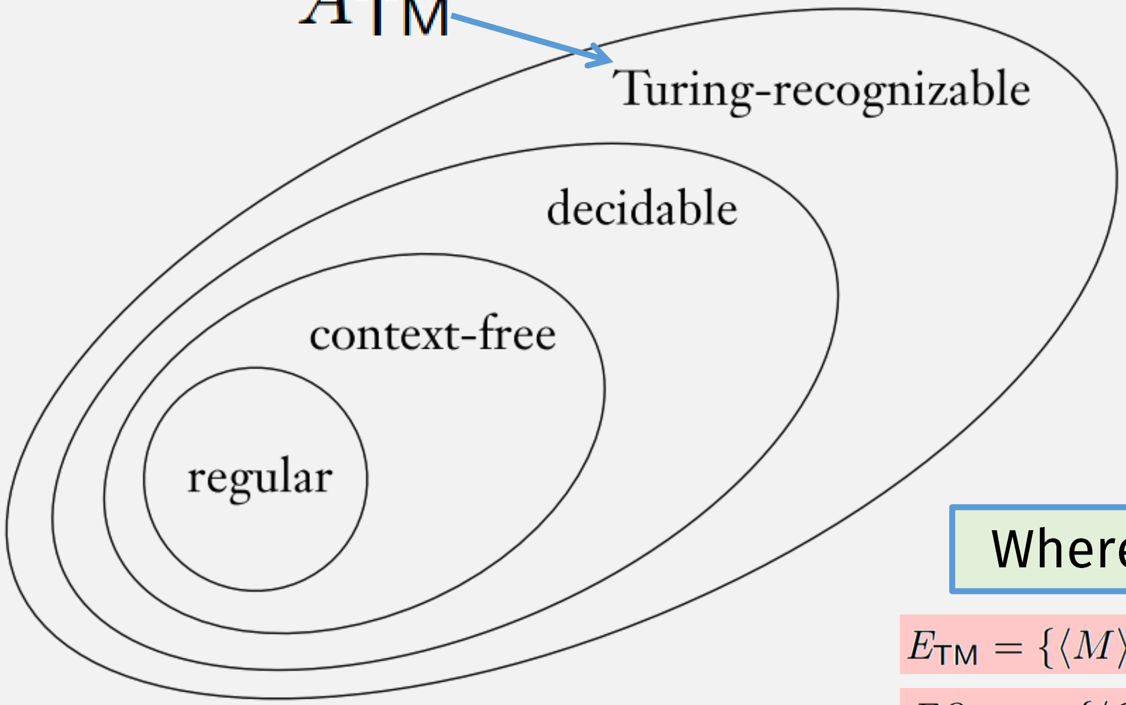
On input <$G$>:

- Construct a CFG $G_{ALL}$ which generates all possible strings
- Run $R$ with $G$ and $G_{ALL}$
- Accept $G$ if $R$ accepts, else reject

# Turing Unrecognizable?

Is there anything out here?

$A_{\mathsf{TM}}$




Where do these go?

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

$EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

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

# Thm: Some langs are not Turing-recognizable

Proof: requires 2 lemmas

- Lemma 1: The **set of all languages** is *uncountable*
  - Proof: Show there is a bijection with another uncountable set …
    - … The set of all infinite binary sequences

- Lemma 2: The **set of all TMs** is *countable*

- Therefore, some language is not recognized by a TM
  (pigeonhole principle)

# Mapping a Language to a Binary Sequence

All Possible Strings

Some Language
(subset of above)

Its (unique)
Binary Sequence

$$
\begin{array}{rcccccccccccl}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots & \} \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots & \} \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots &
\end{array}
$$

Each digit represents one possible string:
- 1 if lang has that string,
- 0 otherwise

# Thm: Some langs are not Turing-recognizable

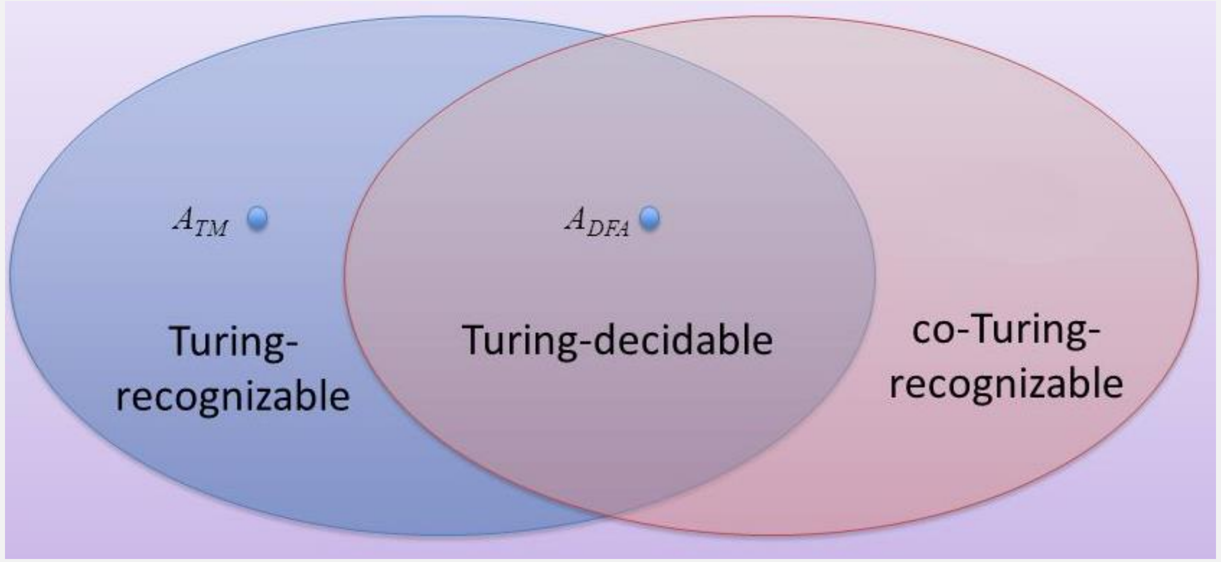
Proof: requires 2 lemmas

- Lemma 1: The **set of all languages** is *uncountable*
  - Proof: Show there is a bijection with another uncountable set …
    - … The set of all infinite binary sequences
    - ➢ Now just prove set of infinite binary sequences is uncountable (diagonalization)
- 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
- Therefore, some language is not recognized by a TM $\blacksquare$

# Co-Turing-Recognizability

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

# Thm: Decidable ⇔ Recognizable & co-Recognizable



# Thm: Decidable ⇔ Recognizable & co-Recognizable

⇒ If a language is **decidable,** then it is **recognizable** and **co-recognizable**

- Decidable => Recognizable (hw5):
  - A decider is a recognizer, bc decidable langs are a subset of recognizable langs
- Decidable => Co-Recognizable:
  - To create co-decider from a decider … switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

⇐ If a language is **recognizable** and **co-recognizable,** then it is **decidable**

# <u>Thm</u>: Decidable ⇔ Recognizable & co-Recognizable

⇒ <u>If</u> a language is **decidable**, <u>then</u> it is **recognizable** and **co-recognizable**

- Decidable => Recognizable:
  - A decider is a recognizer, bc decidable langs are a subset of recognizable langs
- Decidable => Co-Recognizable:
  - To create co-decider from a decider … switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

⇐ <u>If</u> a language is **recognizable** and **co-recognizable**, <u>then</u> it is **decidable**

- Let $M_1$ = recognizer for the language,
- and $M_2$ = recognizer for its complement
- Decider $M$:
  - Run 1 step on $M_1$,
  - Run 1 step on $M_2$,
  - Repeat, until one machine accepts. If it's $M_1$, accept. If it's $M_2$, reject

Termination Arg: Either $M_1$ or $M_2$ must accept and halt, so $M$ halts and is a decider

# A Turing-unrecognizable language

- We've proved:

  $A_{\mathsf{TM}}$ is Turing-recognizable

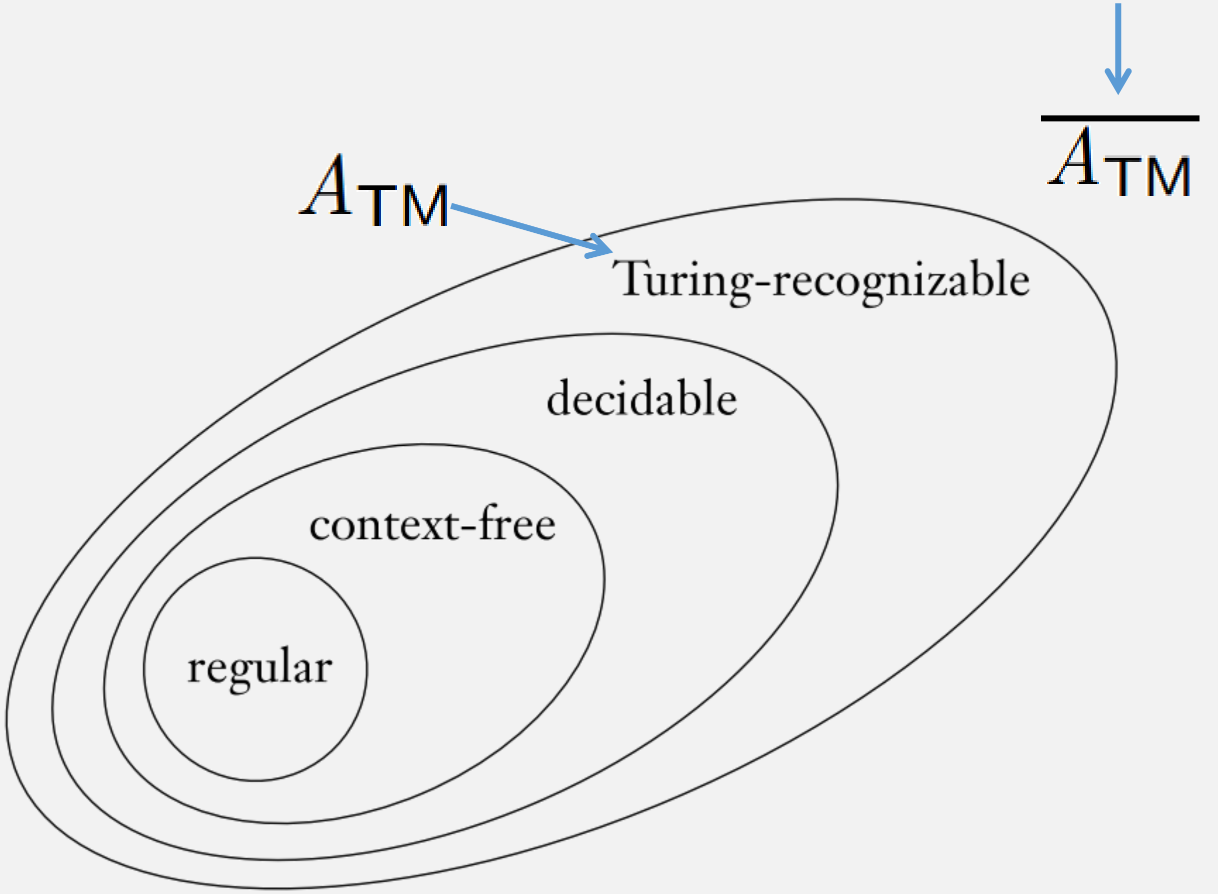
  $A_{\mathsf{TM}}$ is undecidable

- So:

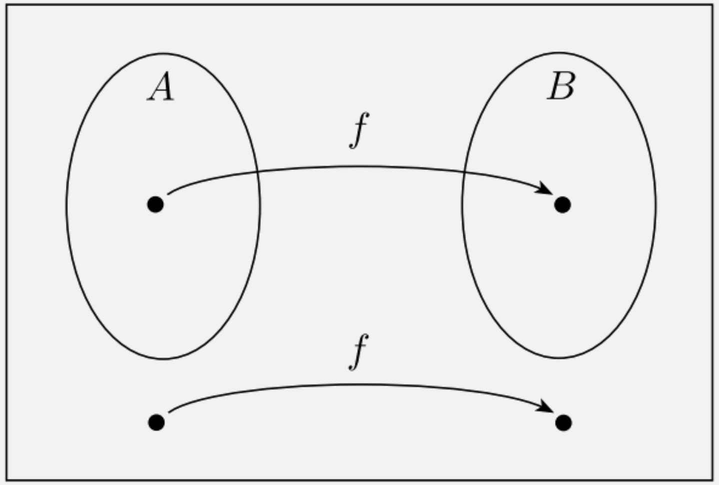
  $\overline{A_{\mathsf{TM}}}$ is not Turing-recognizable

- Because: recognizable & co-recognizable implies decidable

Is there anything out here?



# Mapping Reducibility



## *Last time:* "Reduced"

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

$\Downarrow$

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

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

Proof, by contradiction:

**PROBLEM**: What if it takes forever to create this decider?

- Assume $HALT_{\mathsf{TM}}$ has *decider* $R$; use to create $A_{\mathsf{TM}}$ *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*."

- Contradiction: $A_{\mathsf{TM}}$ is undecidable and has no decider!

**We need a formal definition of "reducibility"**

# *Flashback:* $A_{\mathsf{NFA}}$ is a decidable language

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

Decider for $A_{\mathsf{NFA}}$ :

$N$ = "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 **NFA→DFA**
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?

More generally, we've been saying "**programs = TMs**", but programs do more than accept/reject?

# Computable Functions

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

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 between machines, like **DFA→NFA**
  - E.g., adding states, changing transitions, wrapping TM in TM, etc.

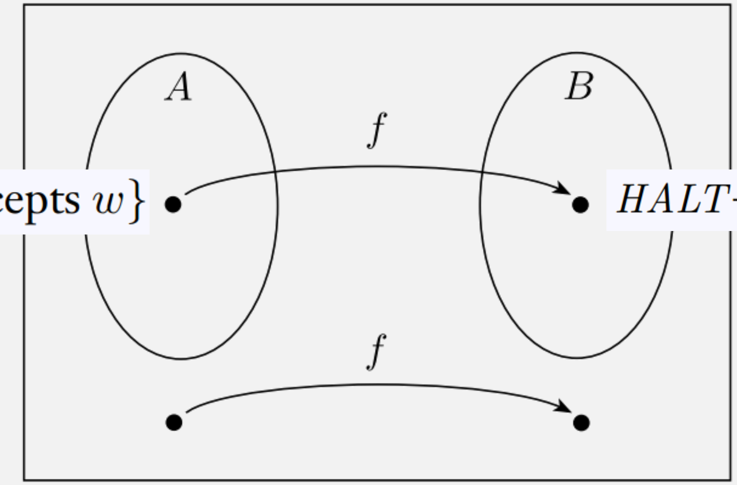
# Mapping Reducibility

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

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




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

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.

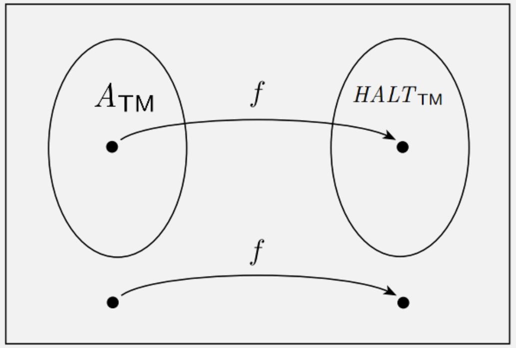
# Thm: $A_{\mathsf{TM}}$ is mapping reducible to $HALT_{\mathsf{TM}}$

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

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

- 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 $M'$

Still need to show:
$M$ accepts $w$
if and only if
$M'$ halts on $w$

Output new $M'$

$M'$ is like $M$, except it always loops when it doesn't accept

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 $B$.

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.

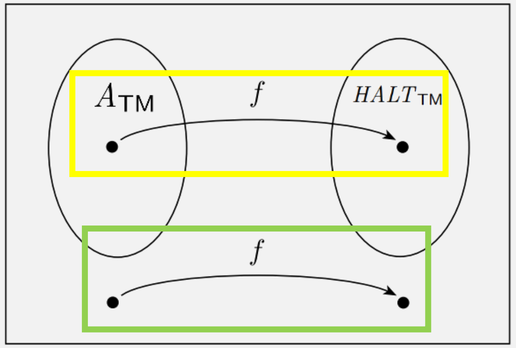
$\Rightarrow$ If $M$ accepts $w$, then $M$' halts on $w$

- $M$' accepts (and thus halts) if $M$ accepts

$\Leftarrow$ If $M$' halts on $w$, then $M$ accepts $w$

$\Leftarrow$ (Alternatively) If $M$ doesn't accept $w$, then $M$' doesn't halt on $w$ (contrapositive)

- Two possibilities
  1. $M$ loops: $M$' loops and doesn't halt
  2. $M$ rejects: $M$' loops and doesn't halt




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 $\langle M', w\rangle$."

# Use Mapping Reducibility to Prove …

- Decidability

- Undecidability

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

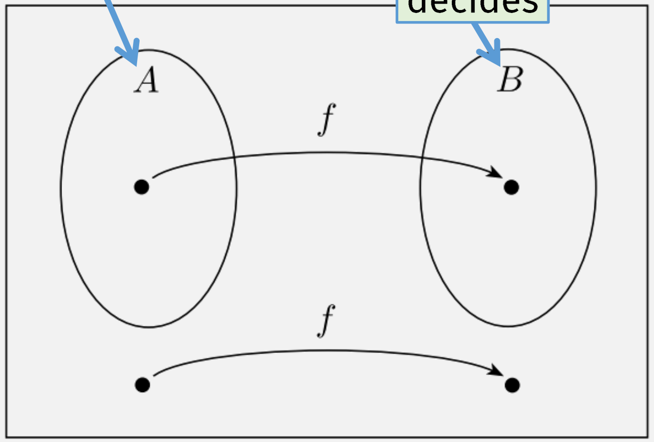
Has a decider

**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."

decides

decides




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 $w$,

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

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).
- Contradiction: we already said $A$ is undecidable

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.
- $A_{\mathsf{TM}} \leq_{\mathrm{m}} HALT_{\mathsf{TM}}$
- Since $A_{\mathsf{TM}}$ is undecidable, then $HALT_{\mathsf{TM}}$ is undecidable

# *Flashback:* $EQ_{\mathsf{TM}}$ 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 $EQ_{\mathsf{TM}}$ has *decider* $R$; use to create $E_{\mathsf{TM}}$ *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*."

<u>Alternate proof</u>: Show: $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

- <u>Computable fn</u> $f$ : $\langle M\rangle$ → $\langle M, M_1\rangle$

<u>Last step</u>: show iff requirements of mapping reducibility (exercise)

# Reducing to complement: $E_{\mathsf{TM}}$ is undecidable

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

Proof, by contradiction:

- Assume $E_{\mathsf{TM}}$ has *decider* $R$; use to create $A_{\mathsf{TM}}$ *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 $M_1$

$M_1 =$ "On input $x$:

**1.** If $x \neq w$, *reject*.

**2.** If $x = w$, run $M$ on input $w$ and *accept* if $M$ does."

**2.** Run $R$ on input $\langle M_1 \rangle$.

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

If $M$ accepts $w$, $M_1$ not in $E_{\mathrm{TM}}$!

Alternate proof: computable fn: $\langle M, w \rangle \rightarrow \langle M_1 \rangle$ ???

- So this only reduces $A_{\mathsf{TM}}$ to $\overline{E_{\mathsf{TM}}}$
- It's good enough! Still proves $E_{\mathsf{TM}}$ is undecidable
  - Because undecidable langs are closed under complement

Last step: show iff requirements of mapping reducibility (exercise)

# Undecidable Langs Closed under Complement

- E.g., if $L$ is undecidable and $\overline{L}$ is decidable …
- … then we can create decider for $L$ from decider for $\overline{L}$ …
- … which is a contradiction!

- Because decidable languages are closed under complement!

# Use Mapping Reducibility to Prove …

- Decidability
- Undecidability
- Recognizability
- Unrecognizability

# More Helpful Theorems

If $A \leq_m B$ and $B$ is $\boxed{\text{Turing-recognizable,}}$ 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.

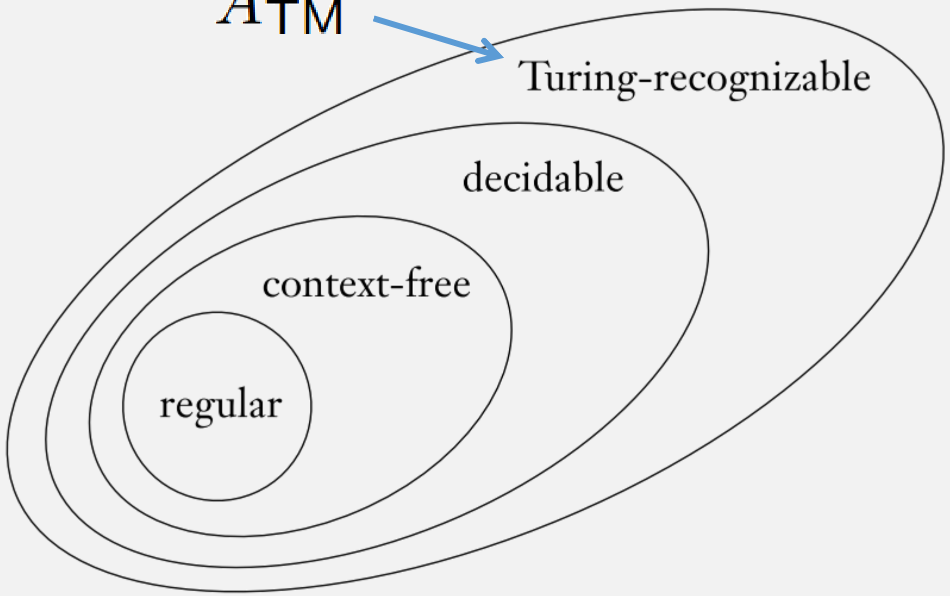
Thm: $EQ_{\mathsf{TM}}$ is neither Turing-recognizable nor co-Turing-recognizable.

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

1. $EQ_{\mathsf{TM}}$ is not Turing-recognizable

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

$A_{\mathsf{TM}}$




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

# Mapping Reducibility implies Mapping Red. of Complements

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 $w$,

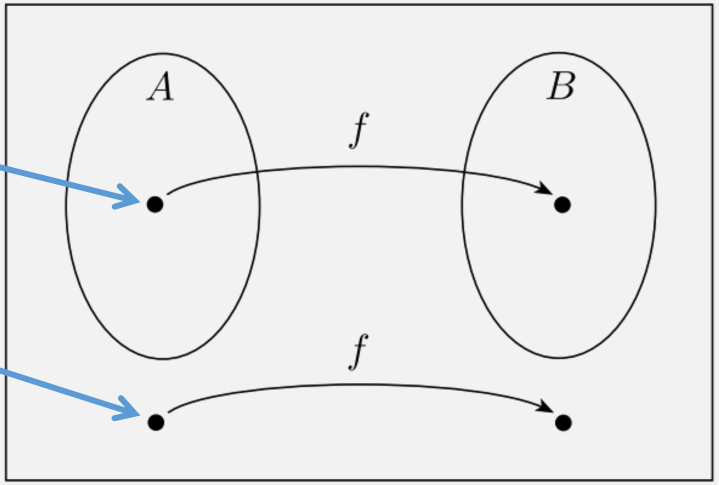
$$\boxed{w \in A \iff f(w) \in B.}$$

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

$\boxed{A \leq_{\mathrm{m}} B}$

implies

$\boxed{\overline{A} \leq_{\mathrm{m}} \overline{B}}$



**Thm:** $EQ_{\mathsf{TM}}$ 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. $EQ_{\mathsf{TM}}$ is not Turing-recognizable

   Two Choices:

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

# Thm: $EQ_{\mathsf{TM}}$ 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

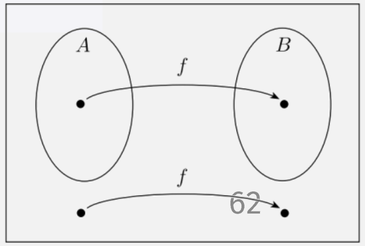
   1. *Reject*."

   $M_2 =$ "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 $M_2$



Thm: $EQ_{\mathsf{TM}}$ 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. $EQ_{\mathsf{TM}}$ is not Turing-recognizable

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

# **Prev**: $EQ_{\mathsf{TM}}$ is not Turing-recognizable

$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle |\ M_1$ and $M_2$ 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 $\langle M_1, M_2\rangle$."

**DONE!**

# Now: $\overline{EQ}_{\mathsf{TM}}$ 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~~ everything

   1. *Accept*."

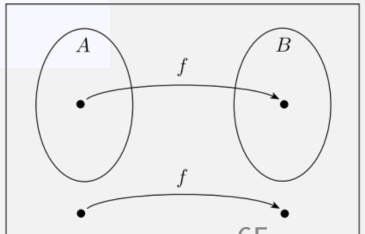
   $M_2 =$ "On any input: ← Accepts nothing or everything

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

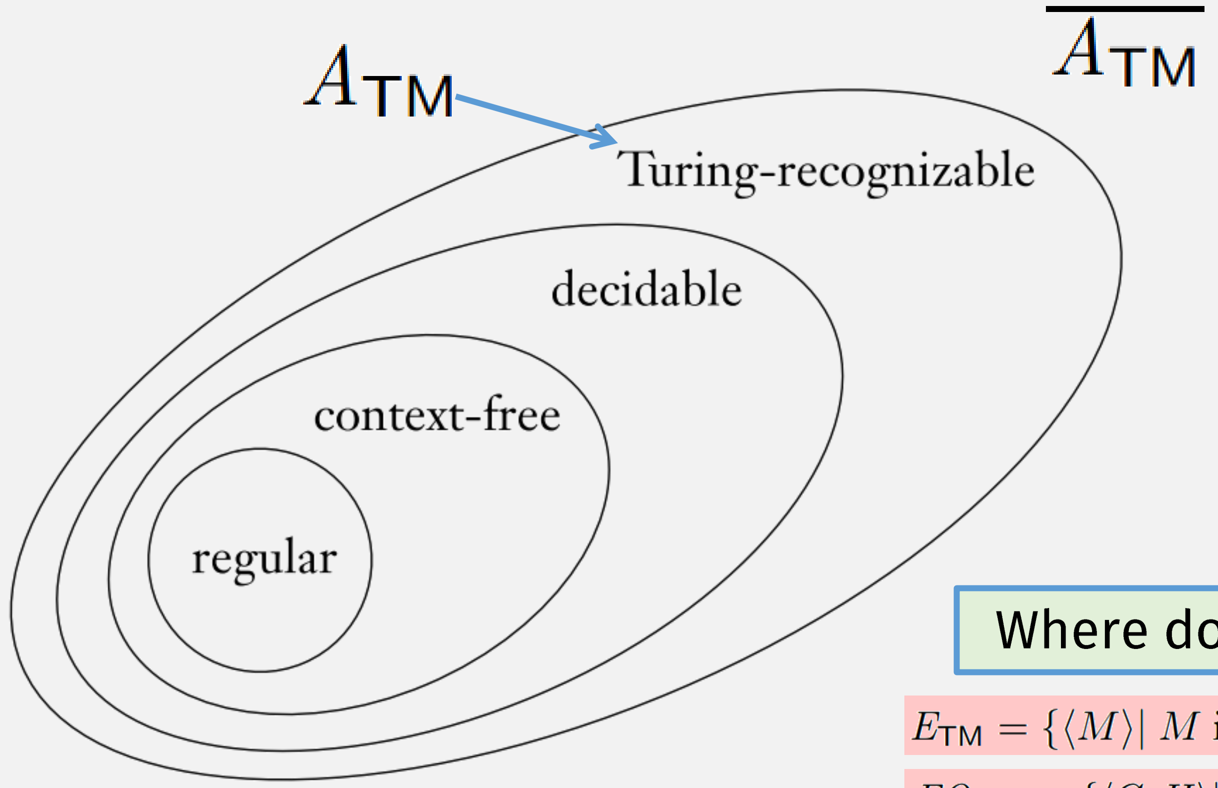
2. Output $\langle M_1, M_2\rangle$."

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

**DONE!**



# Unrecognizable Languages?



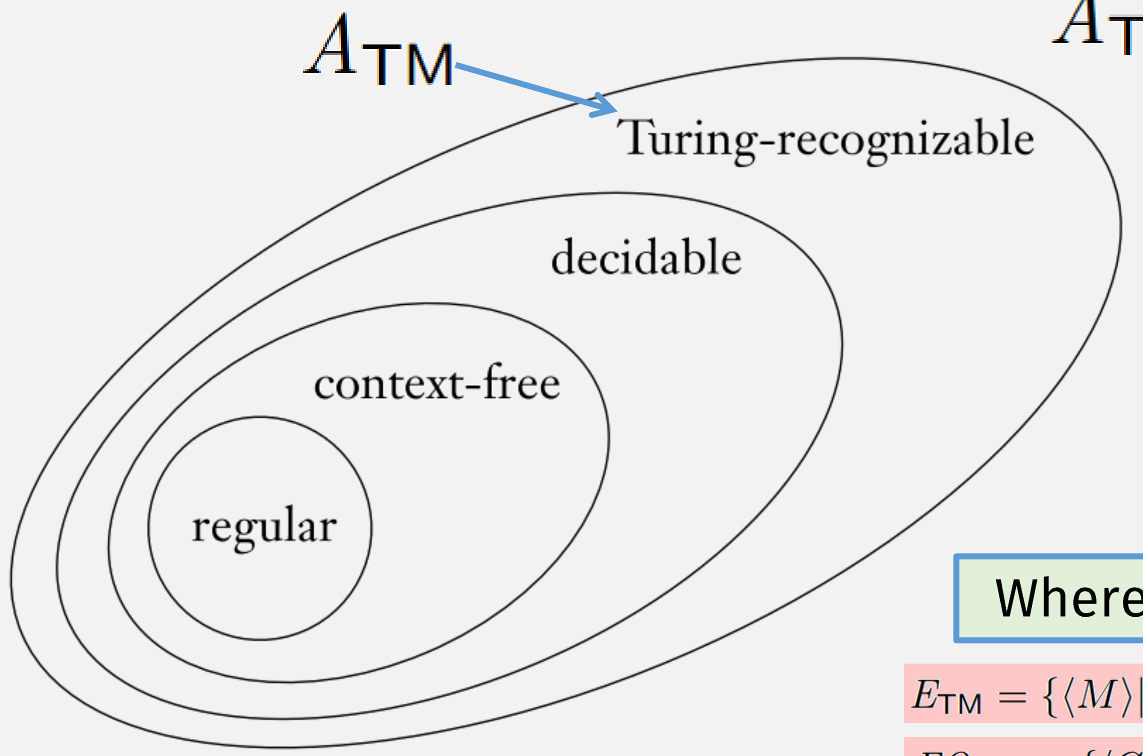

Where do these go?

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

$EQ_{\mathsf{CFG}} = \{\langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

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

# Unrecognizable Languages




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

$EQ_{\mathsf{TM}}$

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

Where do these go?

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

$EQ_{\mathsf{CFG}} = \{\langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

# Check-in Quiz 10/25

On gradescope