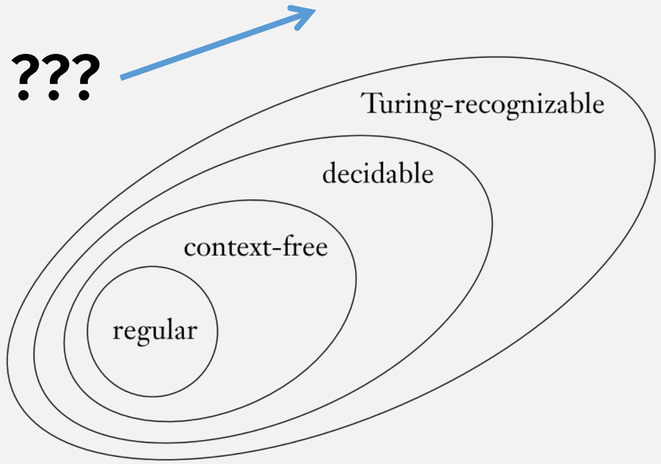
**UMB CS 420**

# Unrecognizability

Tuesday, November 22, 2022



# Announcements

- HW 9 in
    - ~~Due Mon 11/21 11:59pm EST~~
- HW 10 out
    - Due Mon 12/5 11:59pm EST
    - 2 week assignment
- No class Thursday. Happy Thanksgiving!

# *Last Time:* Showing Mapping Reducibility

Step 1:
Show there is computable fn $f$ … by creating a TM

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

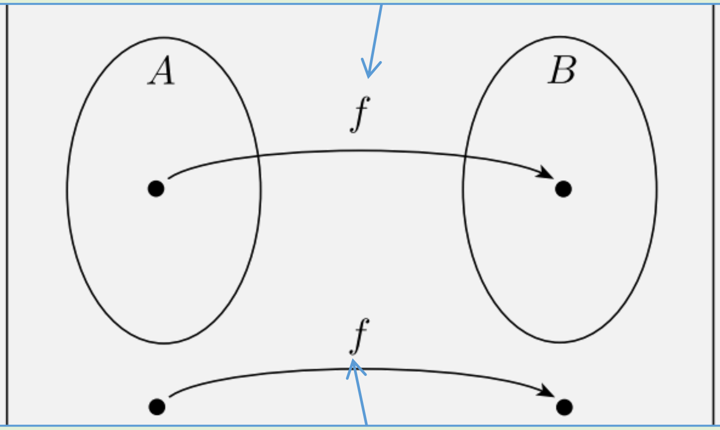
$$\boxed{w \in A \Longleftrightarrow f(w) \in B.}$$

"if and only if"

Step 2:
Prove the iff is true for $f$

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

Step 2a: "forward" direction ($\Rightarrow$): if $w \in A$ then $f(w) \in B$




Step 2b: "reverse" direction ($\Leftarrow$): if $f(w) \in B$ then $w \in A$

Step 2b: Equivalent (contrapositive): if $w \notin A$ then $f(w) \notin B$

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.

# *Last Time:* Using Mapping Reducibility

To prove decidability …

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

Known

Unknown
(want to prove)

To prove undecidability …

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

**Undecidability** Proof Technique #4: **Mapping Reducibility** + this theorem

Be careful with the **direction of the reduction!**

# *Flashback:* $EQ_{\mathsf{TM}}$ is undecidable

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

Proof by contradiction:

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

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

$S =$ "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:* $EQ_{\mathsf{TM}}$ 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 **mapping reducibility:** $E_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$

Step 1: **create computable fn** $f$**, computed by TM** $S$

$S$ = "On input $\langle M\rangle$, where $M$ is a TM:

**1.** Construct: $\langle M, M_1\rangle$, where $M_1$ is a TM that rejects all inputs.

**2.** Output: $\langle M, M_1\rangle$

Step 2: **show iff requirements of mapping reducibility**

Do for HW 10!

And use theorem …

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

# *Flashback:* $E_{\mathsf{TM}}$ is undecidable

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

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 $w$, then $M_1$ not in $E_{\mathrm{TM}}$!
So do the opposite!

$M_1$ :
- accepts $w$ if $M$ does
- rejects everything else

# *Alternate Proof:* $E_{\mathsf{TM}}$ is undecidable

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

Proof, by **mapping reducibility??**: $A_{\mathsf{TM}} \leq_{\mathrm{m}} E_{\mathsf{TM}}$

Step 1: create computable fn $f$: <$M, w$> → <$M_1$>, computed by $S$

$S =$ "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."

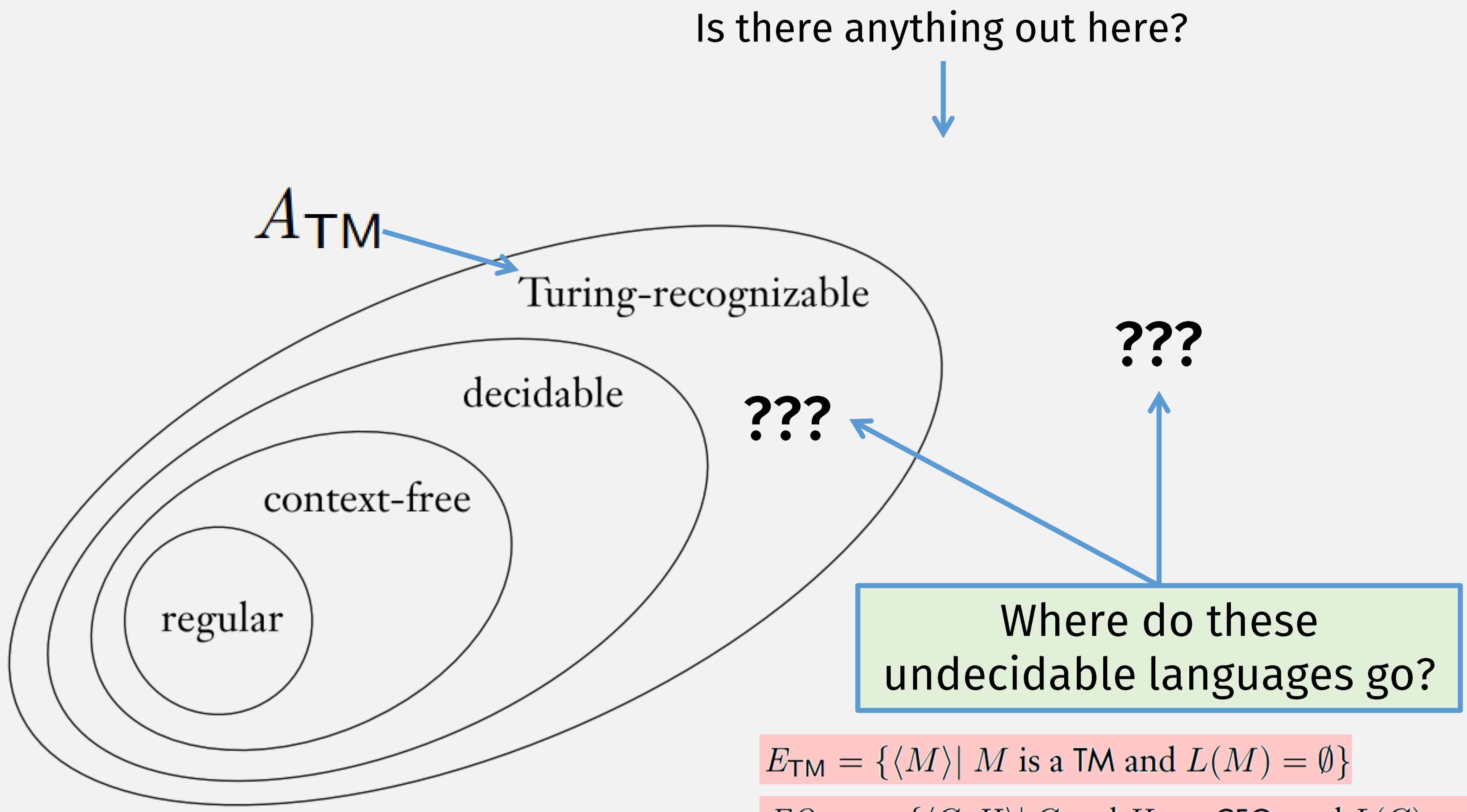
2. Output: $\langle M_1\rangle$.
3. ~~If $R$ accepts, *reject*; if $R$ rejects, *accept*~~."

Step 2: show iff requirements of mapping reducibility:

Do for HW 10!

- This reduces $A_{\mathsf{TM}}$ to $\overline{E_{\mathsf{TM}}}$ !!
- It's good enough, if: **undecidable langs are closed under complement**

# Turing Unrecognizable?




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

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

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

# 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 | $\Sigma^* = \{$ | $\varepsilon,$ | $0,$ | $1,$ | $00,$ | $01,$ | $10,$ | $11,$ | $000,$ | $001,$ | $\cdots \}$ |
| Some Language (subset of above) | $A = \{$ | | $0,$ | | $00,$ | $01,$ | | | $000,$ | $001,$ | $\cdots \}$ |
| Its (unique) Binary Sequence | $\chi_A =$ | $0$ | $1$ | $0$ | $1$ | $1$ | $0$ | $0$ | $1$ | $1$ | $\cdots$ |

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

# Thm: Some langs are not Turing-recognizable

This is an "existence" proof,
but it's not "constructive",
i.e., it doesn't give an example of
an unrecognizable language

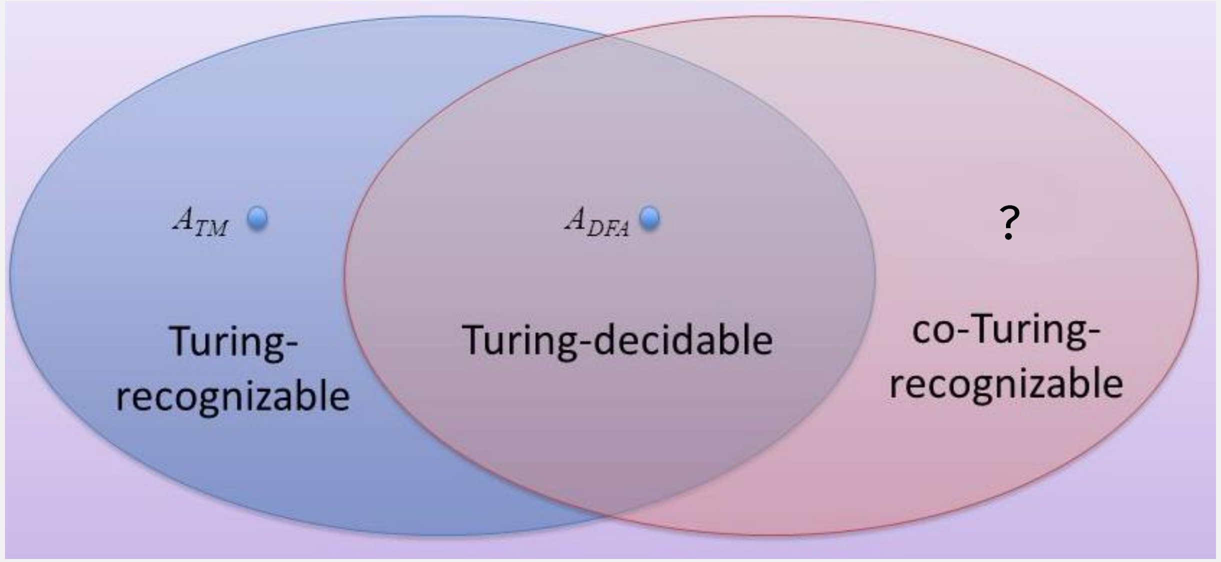
## Proof: requires 2 lemmas

- 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 (exercise)
- 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 (from hw9)
- Therefore, some language is not recognized by a TM ∎

# 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:
  - A decider is a recognizer (that always halts)
- 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**

# Thm: Decidable ⇔ Recognizable & co-Recognizable

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

- Decidable ⇒ Recognizable:
  - A decider is a recognizer (that always halts)
- 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**

- 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

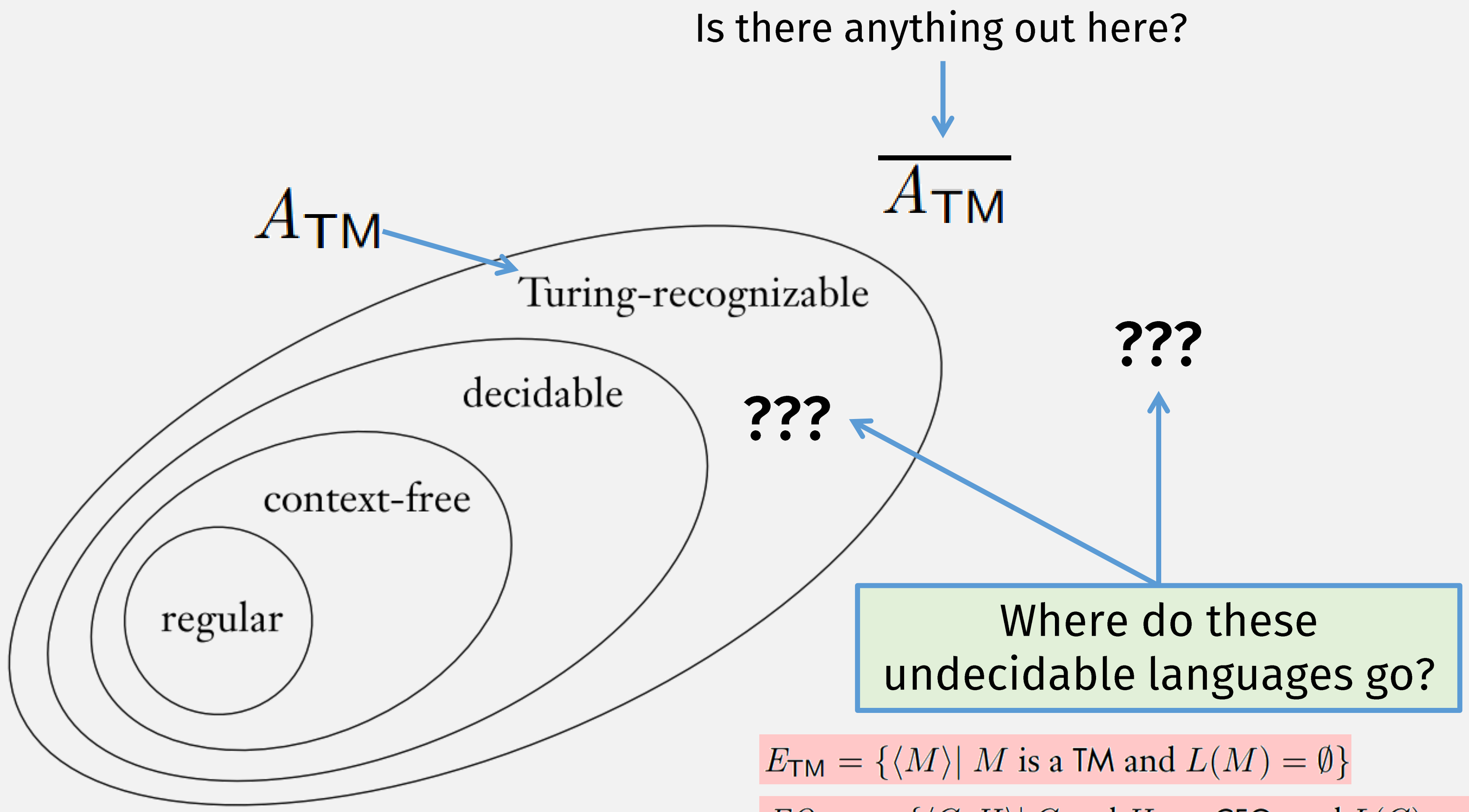
$A_{\mathsf{TM}}$ is undecidable

- So:

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

**Unrecognizability**
Proof Technique #1

- Because: recognizable & co-recognizable ⇒ decidable




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

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

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

# Using 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.

**Unrecognizability**
Proof Technique #2:
Mapping reducibility
+ this theorem

- Same proofs as:

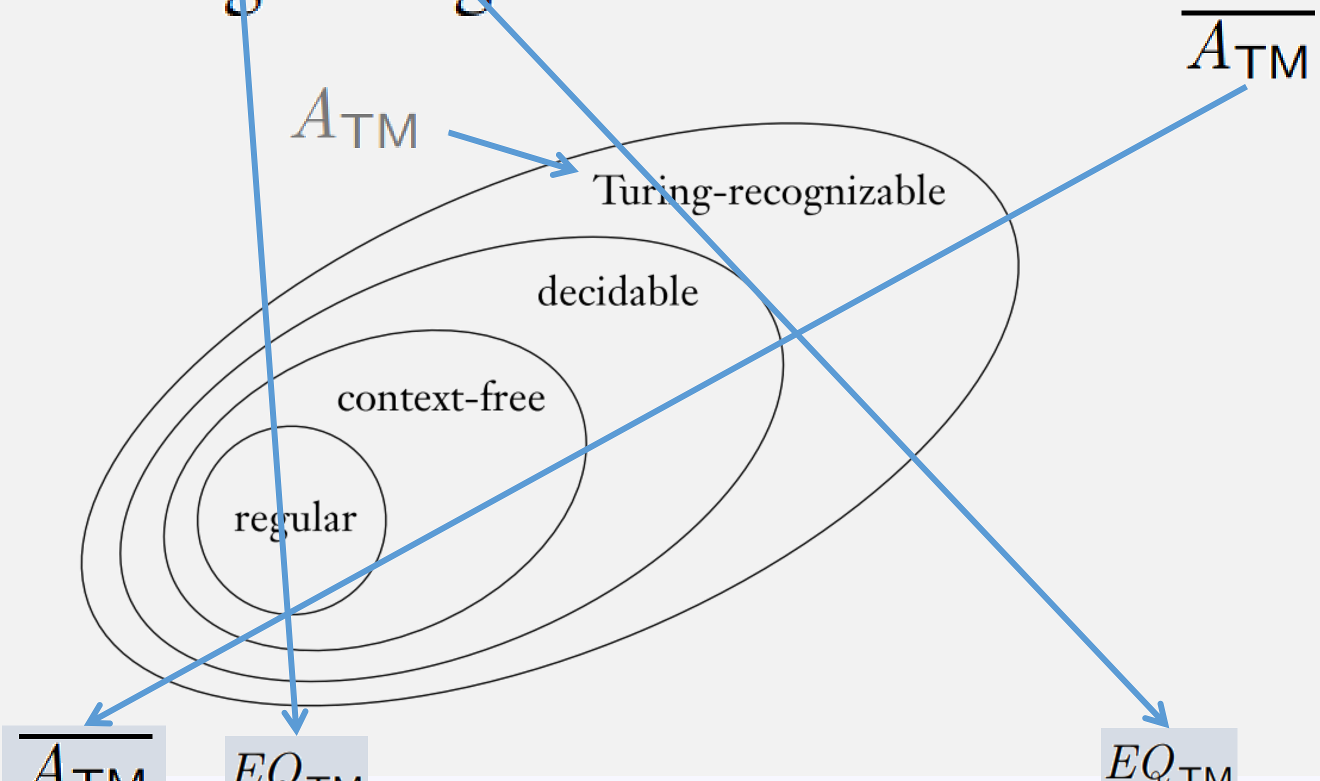
If $A \leq_m B$ and $B$ is $\boxed{\text{decidable,}}$ then $A$ is decidable.

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




If $A \leq_{\mathrm{m}} B$ and $A$ is not Turing-recognizable, then $B$ is 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 $w$,

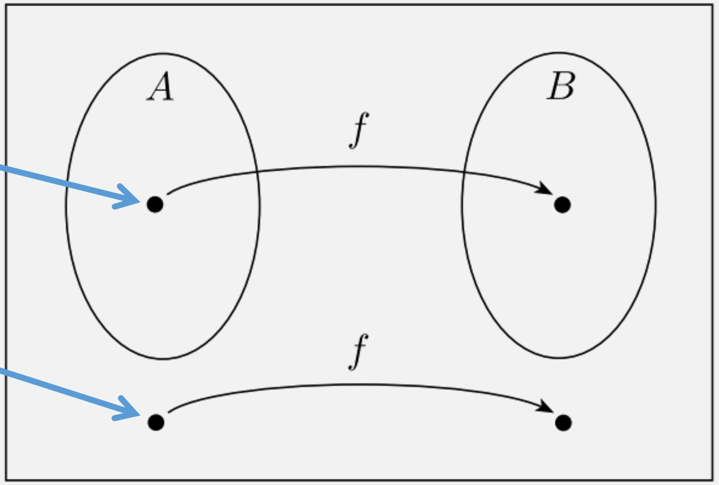
$$\boxed{w \in A \iff f(w) \in B.}$$

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: $\boxed{A_{\mathsf{TM}} \rightarrow \overline{EQ_{\mathsf{TM}}}}$

And use theorem …

If $A \leq_{\mathrm{m}} B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.

# 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: $A_{\mathsf{TM}} \rightarrow \overline{EQ_{\mathsf{TM}}}$

Step 1
Computable fn

$\langle M, w\rangle \rightarrow \langle M_1, M_2\rangle$ $M_1$ 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

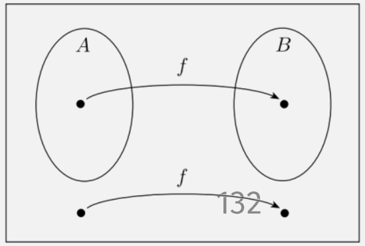
   1. Run $M$ on $w$. If it accepts, *accept*."

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

Step 2, iff:

⇒ If $M$ accepts $w$, then $M_1 \neq M_2$

⇐ If $M$ does not accept $w$, then $M_1 = M_2$



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

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

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

And use theorem …

If $A \leq_{\text{m}} B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.

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

# *Previous:* $EQ_{\mathsf{TM}}$ 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: $A_{\mathsf{TM}} \rightarrow \overline{EQ_{\mathsf{TM}}}$

Step 1

- $\langle M, w\rangle \rightarrow \langle M_1, M_2\rangle$ $M_1$ 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 $\langle M_1, M_2\rangle$."

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

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

- Create Computable fn: $A_{\text{TM}} \rightarrow \overline{EQ}_{\text{TM}}$

Step 1

- $\langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle$ $M_1$ 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*."

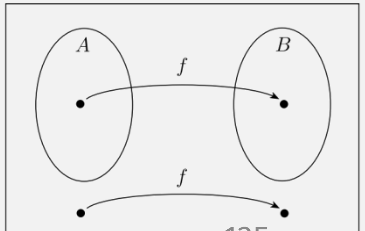
2. Output $\langle M_1, M_2 \rangle$."

Step 2, iff:

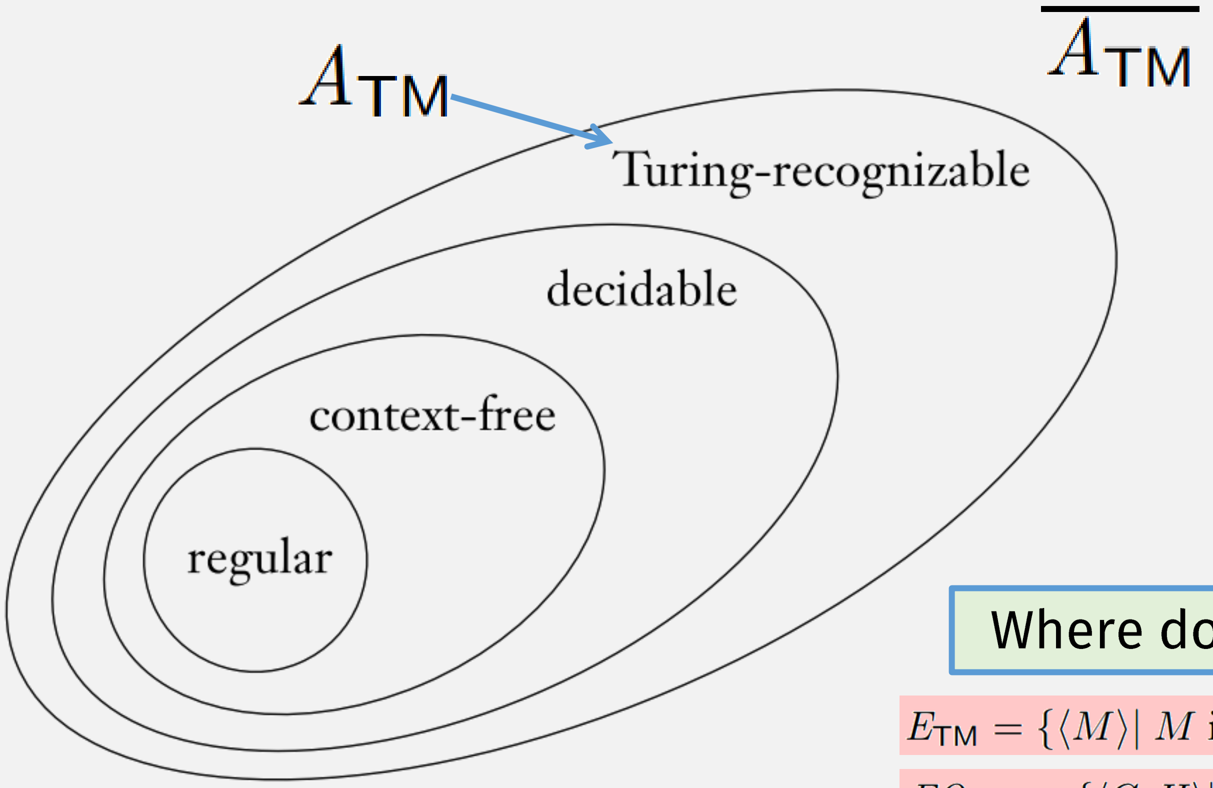
⇒ If $M$ accepts $w$, then $M_1 \boxed{=} M_2$

⇐ If $M$ does not accept $w$, then $M_1 \boxed{\neq} M_2$

**DONE!**



# Unrecognizable Languages?



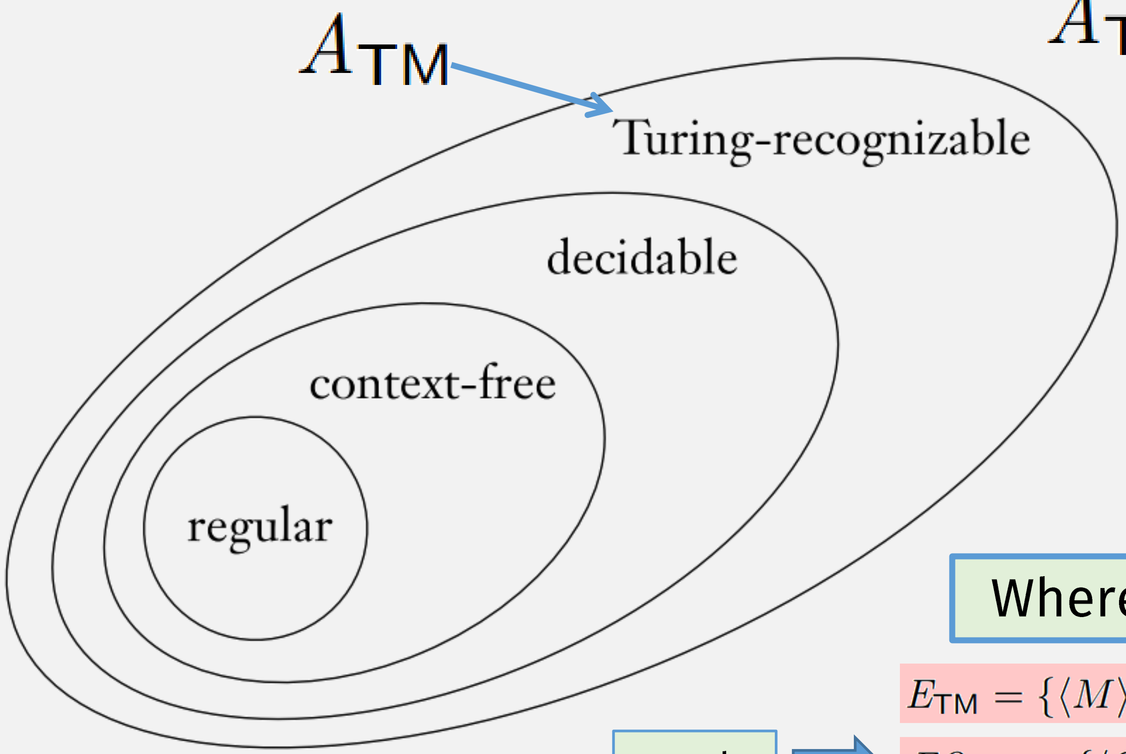

Where do these go?

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

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

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

# Unrecognizable Languages




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

$EQ_{\mathsf{TM}}$

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

Where do these go?

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

next

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

# Thm: $EQ_{\text{CFG}}$ is not Turing-recognizable

Recognizable & co-recognizable implies decidable

**Unrecognizability**
Proof Technique #1

- We've proved:

  $EQ_{\text{CFG}}$ is undecidable

- We now prove:

  $EQ_{\text{CFG}}$ is co-Turing recognizable

- And conclude that:
  - $EQ_{\text{CFG}}$ is not Turing recognizable

# Thm: $EQ_{\text{CFG}}$ is co-Turing-recognizable

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

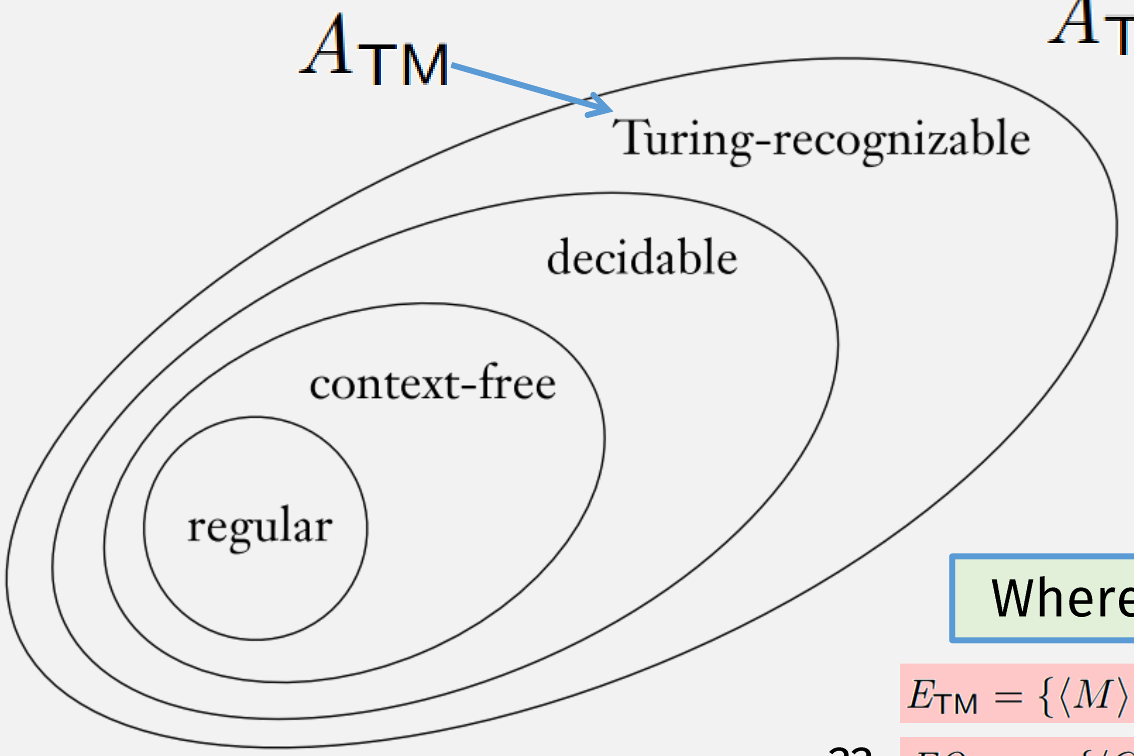
Recognizer for $\overline{EQ}_{\text{CFG}}$:

- On input <$G, H$>:
  - **For** every possible string $w$:
    - Accept if $w \in L(G)$ and $w \notin L(H)$
    - Or accept if $w \in L(H)$ and $w \notin L(G)$
  - **Else** reject

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$

This is only a **recognizer** because it loops for ever when $L(G) = L(H)$

# Unrecognizable Languages




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

$EQ_{\mathsf{TM}}$

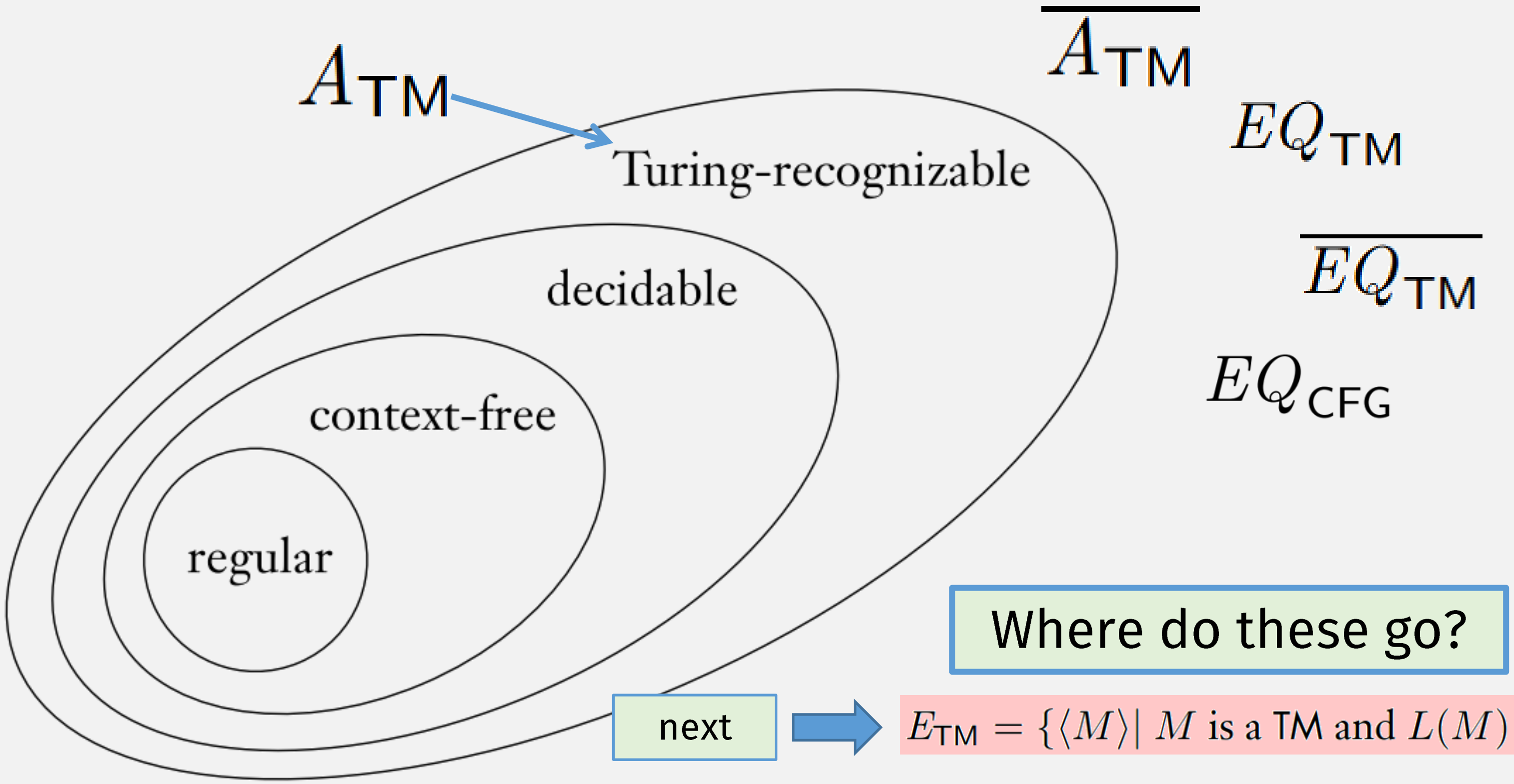
$\overline{EQ_{\mathsf{TM}}}$

**Where do these go?**

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

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

# Unrecognizable Languages




Where do these go?

next

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

# Thm: $E_{\text{TM}}$ is not Turing-recognizable

Recognizable & co-recognizable implies decidable

**Unrecognizability**
Proof Technique #1

- We've proved:
  - $E_{\text{TM}}$ is undecidable
- We now prove:
  $E_{\text{TM}}$ is co-Turing recognizable
- And then conclude that:
  - $E_{\text{TM}}$ is not Turing recognizable

# Thm: $E_{\mathsf{TM}}$ is co-Turing-recognizable

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

Recognizer for $\overline{E_{\mathsf{TM}}}$: Let $s_1, s_2, \ldots$ be a list of all strings in $\Sigma^*$

"On input $\langle M \rangle$, where $M$ is a TM:

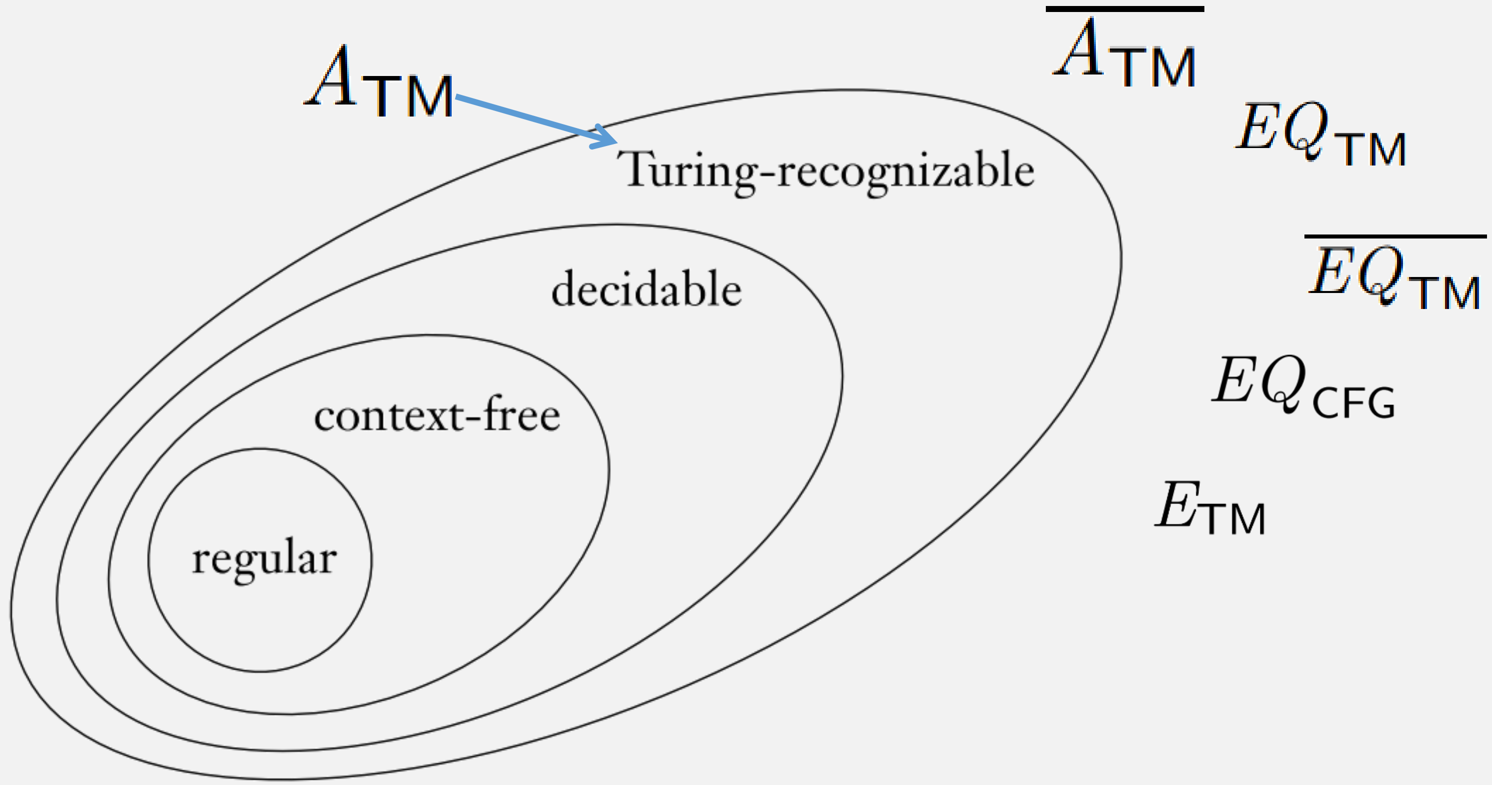
**1.** Repeat the following for $i = 1, 2, 3, \ldots$ .

**2.** Run $M$ for $i$ steps on each input, $s_1, s_2, \ldots, s_i$.

**3.** If $M$ has accepted any of these, *accept*. Otherwise, continue."

This is only a **recognizer** because it loops for ever when $L(M)$ is empty

# Unrecognizable Languages



# Check-in Quiz 11/22

On gradescope