Computational Theory

Reducibility

Curtis Larsen

Utah Tech University—Computing

Fall 2024

Undecidability

Reading: Sipser §5.1.

Theorem 5.1 $HALT_{\text{TM}}$

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

Theorem 5.1 $HALT_{\text{TM}}$

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

Theorem 5.1: $HALT_{\text{TM}}$ is undecidable.

Theorem 5.1 $HALT_{\text{TM}}$

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

Theorem 5.1: $HALT_{\text{TM}}$ is undecidable.

Proof Idea: Use a proof by contradiction. Assume $HALT_{\text{TM}}$ is decidable. Use $HALT_{\text{TM}}$'s decider to construct a decider for A_{TM}. By theorem 4.11, A_{TM} is undecidable. This is a contradiction. We will call this a reduction of A_{TM} to $HALT_{\text{TM}}$.

Theorem 5.1 $HALT_{\text{TM}}$

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

Theorem 5.1: $HALT_{\text{TM}}$ is undecidable.

Proof: Assume $HALT_{\text{TM}}$ is decidable by R. Construct a decider for A_{TM}.

Let $S =$ "On input $\langle M, w\rangle$ an encoding of a TM and a string:

1. Run 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 accepted, *accept*; if M rejected, *reject*."

Theorem 5.1 $HALT_{\text{TM}}$

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

Theorem 5.1: $HALT_{\text{TM}}$ is undecidable.

Proof: Assume $HALT_{\text{TM}}$ is decidable by R. Construct a decider for A_{TM}.

Let $S =$ "On input $\langle M, w\rangle$ an encoding of a TM and a string:

1. Run 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 accepted, *accept*; if M rejected, *reject*."

If M accepts w, S accepts. If M rejects w, S rejects. If M loops on w, S rejects. This is a decider for A_{TM}. This is the contradiction.

Theorem 5.2 E_{TM}

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

Theorem 5.2 E_{TM}

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

Theorem 5.2: E_{TM} is undecidable.

Theorem 5.2 E_{TM}

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

Theorem 5.2: E_{TM} is undecidable.

Proof Idea: Use a proof by contradiction. Assume E_{TM} is decidable. Use E_{TM}'s decider to construct a decider for A_{TM}. By theorem 4.11, A_{TM} is undecidable. This is a contradiction. We will call this a reduction of A_{TM} to E_{TM}.

Theorem 5.2 E_{TM}

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

Theorem 5.2: E_{TM} is undecidable.

Proof: Assume E_{TM} is decidable by R. Construct a decider for A_{TM}.

Intermediate step.

Let $M_1 =$ "On input x:

1. if $x \neq w$, *reject*.
2. if $x = w$, run M on input w and *accept* if M does."

Note that $L(M_1) = \emptyset$ if M does not accept w, and $L(M_1) = \{w\}$ if M accepts w.

Theorem 5.2 E_{TM}

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

Theorem 5.2: E_{TM} is undecidable.

Proof: Continued

Let $S =$ "On input $\langle M, w\rangle$ an encoding of a TM and a string:

1. Construct M_1 from M and w as described above.
2. Run R on input $\langle M_1\rangle$.
3. If R accepted, *reject*; if R rejected, *accept*."

Theorem 5.2 E_{TM}

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

Theorem 5.2: E_{TM} is undecidable.

Proof: Continued

Let $S =$ "On input $\langle M, w \rangle$ an encoding of a TM and a string:

1. Construct M_1 from M and w as described above.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepted, *reject*; if R rejected, *accept*."

If M accepts w, then the $L(M_1) \neq \emptyset$, so R will reject $\langle M_1 \rangle$, and S accepts. If M does not accept w, then the $L(M_1) = \emptyset$, so R will accept $\langle M_1 \rangle$, and S rejects. This is a decider for A_{TM}. Contradiction with theorem 4.11.

Mapping Reductions

Reading: Sipser §5.3.

Definition 5.17

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

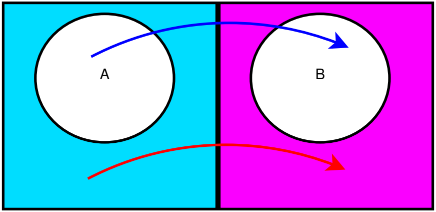
Definition 5.20

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

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

The function f is called the ***mapping reduction*** of A to B or the ***reduction*** of A to B.

Definition 5.20



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

The function f is represented by the two arrows in the diagram above. The blue arrow indicates $w \in A \Rightarrow f(w) \in B$. The red arrow indicates $w \in \overline{A} \Rightarrow f(w) \in \overline{B}$.

Theorem 5.22

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

Theorem 5.22

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

Proof: Let M be the decider for B and f be the reduction from A to B.

Theorem 5.22

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

Proof: Let M be the decider for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

Theorem 5.22

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

Proof: Let M be the decider for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N halts. Why?

Theorem 5.22

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

Proof: Let M be the decider for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N halts. Why? f is a computable function. M is a decider.

Theorem 5.22

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

Proof: Let M be the decider for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N halts. Why? f is a computable function. M is a decider.

N decides A. Why?

Theorem 5.22

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

Proof: Let M be the decider for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N halts. Why? f is a computable function. M is a decider.

N decides A. Why? If $w \in A$, then $f(w) \in B$, and M accepts.
If $w \notin A$, then $f(w) \notin B$, and M rejects.

Corollary 5.23

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

Corollary 5.23

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

Proof: If B is decidable, then A is decidable. Since A is undecidable, B can't be decidable.

Using Reductions

- Our main technique for proving undecidability is to reduce from a known undecidable language to a suspected undecidedable language. This will become standardized with mapping reductions.
- Our main technique for proving decidability has been to provide a decider. We can now also use reduction from a suspected decidable language to a known decidedable language.

Reduction Proofs

Steps to a reduction proof.

- Select the language to be reduced from, A, and describe its instances.
- Select the language to be reduced to, B, and describe its instances.
- Provide the reduction function that maps *any* instance of A into *some* instance of B. Be sure to prove it is a computable function.
- Prove that $w \in A \Rightarrow f(w) \in B$.
- Prove that $w \in \overline{A} \Rightarrow f(w) \in \overline{B}$. Sometimes this is easier to prove by showing the equivalent $f(w) \in B \Rightarrow w \in A$.
- Conclude that $A \leq_m B$.

Reduction Proofs

Due to space considerations in these slides, we will not label each of these steps. But you should observe that they are all there. Assignments and exam questions will require you to use these steps, and label them.

Example 5.24

$HALT_{\textsf{TM}} = \{\langle M, w\rangle | M$ is a TM, w is a string, and M halts on $w\}$

Example 5.24

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

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

Example 5.24

$HALT_{\mathsf{TM}} = \{\langle M, w\rangle | M$ is a TM, w is a string, and M halts on $w\}$

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

From theorem 4.11, A_{TM} is undecidable. From Corollary 5.23 If $A_{\mathsf{TM}} \leq_m HALT_{\mathsf{TM}}$, then $HALT_{\mathsf{TM}}$ is undecidable.

Example 5.24

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

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

From theorem 4.11, A_{TM} is undecidable. From Corollary 5.23 If $A_{\text{TM}} \leq_m HALT_{\text{TM}}$, then $HALT_{\text{TM}}$ is undecidable.

Our task is to provide the reduction. That is, the computable function that mapping reduces A_{TM} to $HALT_{\text{TM}}$.

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w\rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w'\rangle$."

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w \rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w' \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w \rangle) = \langle M', w' \rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w\rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w'\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w\rangle) = \langle M', w'\rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.
- If M accepts w,

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w\rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w'\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w\rangle) = \langle M', w'\rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.
- If M accepts w,M' will accept w', (will halt).

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w \rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w' \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w \rangle) = \langle M', w' \rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.
- If M accepts w,M' will accept w', (will halt).
- If M rejects w,

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w\rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w'\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w\rangle) = \langle M', w'\rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.
- If M accepts w,M' will accept w', (will halt).
- If M rejects w,M' will loop on w', (will not halt).

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let F = "On input $\langle M, w\rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   M' = "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w'\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w\rangle) = \langle M', w'\rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.
- If M accepts w,M' will accept w', (will halt).
- If M rejects w,M' will loop on w', (will not halt).
- If M loops on w,

The Reduction of A_{TM} to $HALT_{\text{TM}}$

Let $F =$ "On input $\langle M, w \rangle$:

1. Let $w' = w$.
2. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on input x.
   2. If M accepts, *accept*.
   3. If M rejects, *loop*."
3. Output $\langle M', w' \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow F(\langle M, w \rangle) = \langle M', w' \rangle \in HALT_{\text{TM}}$

- F halts. Note that F does not run M', it only computes it.
- If M accepts w,M' will accept w', (will halt).
- If M rejects w,M' will loop on w', (will not halt).
- If M loops on w,M' will loop on w', (will not halt).

Example 5.26

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

Example 5.26

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

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

Example 5.26

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

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

From theorem 5.2, E_{TM} is undecidable. From Corollary 5.23 If $E_{\text{TM}} \leq_m EQ_{\text{TM}}$, then EQ_{TM} is undecidable.

Example 5.26

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

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

From theorem 5.2, E_{TM} is undecidable. From Corollary 5.23 If $E_{\text{TM}} \leq_m EQ_{\text{TM}}$, then EQ_{TM} is undecidable.

Our task is to provide the reduction. That is, the computable function that mapping reduces E_{TM} to EQ_{TM}.

The Reduction of E_{TM} to EQ_{TM}

Let $G =$ "On input $\langle M \rangle$:

1. Let $M_1 = M$.
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. *reject*."
3. Output $\langle M_1, M_2 \rangle$."

The Reduction of E_{TM} to EQ_{TM}

Let $G =$ “On input $\langle M \rangle$:

1. Let $M_1 = M$.
2. Construct the following machine M_2.
   $M_2 =$ “On input x:
   1. *reject*.”
3. Output $\langle M_1, M_2 \rangle$.”

Analysis: $\langle M \rangle \in E_{\text{TM}} \Leftrightarrow G(\langle M \rangle) = \langle M_1, M_2 \rangle \in EQ_{\text{TM}}$

- G halts.

The Reduction of E_{TM} to EQ_{TM}

Let $G =$ "On input $\langle M \rangle$:

1. Let $M_1 = M$.
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M \rangle \in E_{\text{TM}} \Leftrightarrow G(\langle M \rangle) = \langle M_1, M_2 \rangle \in EQ_{\text{TM}}$

- G halts.
- If $L(M) = \emptyset$,

The Reduction of E_{TM} to EQ_{TM}

Let $G =$ "On input $\langle M \rangle$:

1. Let $M_1 = M$.
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M \rangle \in E_{\text{TM}} \Leftrightarrow G(\langle M \rangle) = \langle M_1, M_2 \rangle \in EQ_{\text{TM}}$

- G halts.
- If $L(M) = \emptyset$, $L(M_1) = L(M_2) = \emptyset$.

The Reduction of E_{TM} to EQ_{TM}

Let $G =$ "On input $\langle M \rangle$:

1. Let $M_1 = M$.
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M \rangle \in E_{\text{TM}} \Leftrightarrow G(\langle M \rangle) = \langle M_1, M_2 \rangle \in EQ_{\text{TM}}$

- G halts.
- If $L(M) = \emptyset$, $L(M_1) = L(M_2) = \emptyset$.
- If $L(M) \neq \emptyset$,

The Reduction of E_{TM} to EQ_{TM}

Let $G =$ "On input $\langle M \rangle$:

1. Let $M_1 = M$.
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M \rangle \in E_{\text{TM}} \Leftrightarrow G(\langle M \rangle) = \langle M_1, M_2 \rangle \in EQ_{\text{TM}}$

- G halts.
- If $L(M) = \emptyset$, $L(M_1) = L(M_2) = \emptyset$.
- If $L(M) \neq \emptyset$, $L(M_1) \neq L(M_2) = \emptyset$.

Example 5.27 E_{TM} is undecidable

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

Example 5.27 E_{TM} is undecidable

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

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

Example 5.27 E_{TM} is undecidable

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

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

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

Example 5.27 E_{TM} is undecidable

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

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

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

From theorem 4.11, A_{TM} is undecidable. From Corollary 5.23 If $A_{\text{TM}} \leq_m \overline{E_{\text{TM}}}$, then $\overline{E_{\text{TM}}}$ is undecidable, and E_{TM} is undecidable.

Example 5.27 E_{TM} is undecidable

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

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

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

From theorem 4.11, A_{TM} is undecidable. From Corollary 5.23 If $A_{\text{TM}} \leq_m \overline{E_{\text{TM}}}$, then $\overline{E_{\text{TM}}}$ is undecidable, and E_{TM} is undecidable.

Our task is to provide the reduction. That is, the computable function that mapping reduces A_{TM} to $\overline{E_{\text{TM}}}$.

The Reduction of A_{TM} to $\overline{E_{\text{TM}}}$

Let $H =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1 \rangle$."

The Reduction of A_{TM} to $\overline{E_{\text{TM}}}$

Let $H =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow H(\langle M, w \rangle) = \langle M_1 \rangle \in \overline{E_{\text{TM}}}$

- H halts.

The Reduction of A_{TM} to $\overline{E_{\text{TM}}}$

Let $H =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow H(\langle M, w\rangle) = \langle M_1\rangle \in \overline{E_{\text{TM}}}$

- H halts.
- If M accepts w,

The Reduction of A_{TM} to $\overline{E_{\text{TM}}}$

Let $H =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow H(\langle M, w\rangle) = \langle M_1\rangle \in \overline{E_{\text{TM}}}$

- H halts.
- If M accepts w,M_1 will accept w, $L(M_1) \neq \emptyset$.

The Reduction of A_{TM} to $\overline{E_{\mathsf{TM}}}$

Let $H =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\mathsf{TM}} \Leftrightarrow H(\langle M, w \rangle) = \langle M_1 \rangle \in \overline{E_{\mathsf{TM}}}$

- H halts.
- If M accepts w,M_1 will accept w, $L(M_1) \neq \emptyset$.
- If M rejects w,

The Reduction of A_{TM} to $\overline{E_{\mathsf{TM}}}$

Let $H =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1\rangle$."

Analysis: $\langle M, w\rangle \in A_{\mathsf{TM}} \Leftrightarrow H(\langle M, w\rangle) = \langle M_1\rangle \in \overline{E_{\mathsf{TM}}}$

- H halts.
- If M accepts w,M_1 will accept w, $L(M_1) \neq \emptyset$.
- If M rejects w,M_1 will reject w, $L(M_1) = \emptyset$.

The Reduction of A_{TM} to $\overline{E_{\text{TM}}}$

Let $H =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow H(\langle M, w \rangle) = \langle M_1 \rangle \in \overline{E_{\text{TM}}}$

- H halts.
- If M accepts w,M_1 will accept w, $L(M_1) \neq \emptyset$.
- If M rejects w,M_1 will reject w, $L(M_1) = \emptyset$.
- If M loops on w,

The Reduction of A_{TM} to $\overline{E_{\mathsf{TM}}}$

Let $H =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\mathsf{TM}} \Leftrightarrow H(\langle M, w \rangle) = \langle M_1 \rangle \in \overline{E_{\mathsf{TM}}}$

- H halts.
- If M accepts w,M_1 will accept w, $L(M_1) \neq \emptyset$.
- If M rejects w,M_1 will reject w, $L(M_1) = \emptyset$.
- If M loops on w,M_1 will loop on w, $L(M_1) = \emptyset$.

The Reduction of A_{TM} to $\overline{E_{\mathsf{TM}}}$

Let $H =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. If $x \neq w$, *reject*.
   2. Run M on input w and *accept* if M does.
   3. *reject*."
2. Output $\langle M_1\rangle$."

Analysis: $\langle M, w\rangle \in A_{\mathsf{TM}} \Leftrightarrow H(\langle M, w\rangle) = \langle M_1\rangle \in \overline{E_{\mathsf{TM}}}$

- H halts.
- If M accepts w,M_1 will accept w, $L(M_1) \neq \emptyset$.
- If M rejects w,M_1 will reject w, $L(M_1) = \emptyset$.
- If M loops on w,M_1 will loop on w, $L(M_1) = \emptyset$.

Theorem 5.28

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

Theorem 5.28

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

Proof: Let M be the recognizer for B and f be the reduction from A to B.

Theorem 5.28

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

Proof: Let M be the recognizer for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

Theorem 5.28

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

Proof: Let M be the recognizer for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N accepts w if M accepts $f(w)$.

Theorem 5.28

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

Proof: Let M be the recognizer for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N accepts w if M accepts $f(w)$. N rejects w if M rejects $f(w)$.

Theorem 5.28

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

Proof: Let M be the recognizer for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N accepts w if M accepts $f(w)$. N rejects w if M rejects $f(w)$. N loops on w if M loops on $f(w)$.

Theorem 5.28

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

Proof: Let M be the recognizer for B and f be the reduction from A to B.

Let $N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$.
3. If M accepts $f(w)$, then *accept*. Otherwise *reject*."

N accepts w if M accepts $f(w)$. N rejects w if M rejects $f(w)$. N loops on w if M loops on $f(w)$.

N recognizes A.

Corollary 5.29

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

Corollary 5.29

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

Proof: If B is Turing-recognizable, then A is Turing-recognizable. Since A is not Turing-recognizable, B can't be Turing-recognizable.

Using Corollary 5.29

A typical use of Corollary 5.29 is to prove that a language, B, not Turing-recognizable.

- $\overline{A_{\textsf{TM}}}$ is not Turing-recognizable. (Corollary 4.23)
- $A \leq_m B \Leftrightarrow \overline{A} \leq_m \overline{B}$. By definition.
- Provide a mapping reduction $A_{\textsf{TM}} \leq_m \overline{B}$.
- This means $\overline{A_{\textsf{TM}}} \leq_m B$.
- By Corollary 5.29, B is not Turing-recognizable.

Theorem 5.30

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

Theorem 5.30

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

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

Theorem 5.30

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

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

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

Theorem 5.30

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

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

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

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

Theorem 5.30

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

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

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

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

We will use reductions from A_{TM} to $\overline{EQ_{\text{TM}}}$ and EQ_{TM}. These reductions will be paired with Corollary 5.29.

EQ_{TM} is not Turing-recognizable

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

EQ_{TM} is not Turing-recognizable

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

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

$EQ_{\textsf{TM}}$ is not Turing-recognizable

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

$A_{\textsf{TM}} = \{\langle M, w\rangle | M$ is a TM, w is a string, and M accepts $w\}$

Proof strategy: Providing a reduction from $A_{\textsf{TM}}$ to $\overline{EQ_{\textsf{TM}}}$. Deduce that $\overline{A_{\textsf{TM}}} \leq_m EQ_{\textsf{TM}}$. By corollary 4.23, $\overline{A_{\textsf{TM}}}$ is not Turing-recognizable. By corollary 5.29, $EQ_{\textsf{TM}}$ is not Turing-recognizable.

Our task is to provide the reduction. That is, the computable function that mapping reduces $A_{\textsf{TM}}$ to $\overline{EQ_{\textsf{TM}}}$.

The Reduction of A_{TM} to $\overline{EQ_{\mathsf{TM}}}$

Let $f =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

The Reduction of A_{TM} to $\overline{EQ_{\text{TM}}}$

Let $f =$ “On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ “On input x:
   1. *reject*.”
2. Construct the following machine M_2.
   $M_2 =$ “On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*.”
3. Output $\langle M_1, M_2\rangle$.”

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow f(\langle M, w\rangle) = \langle M_1, M_2\rangle \in \overline{EQ_{\text{TM}}}$

- f halts.

The Reduction of A_{TM} to $\overline{EQ_{\mathsf{TM}}}$

Let $f =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\mathsf{TM}} \Leftrightarrow f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in \overline{EQ_{\mathsf{TM}}}$

- f halts.
- If M accepts w,

The Reduction of A_{TM} to $\overline{EQ_{\text{TM}}}$

Let $f =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in \overline{EQ_{\text{TM}}}$

- f halts.
- If M accepts w, $L(M_1) \neq L(M_2)$.

The Reduction of A_{TM} to $\overline{EQ_{\mathsf{TM}}}$

Let $f =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\mathsf{TM}} \Leftrightarrow f(\langle M, w\rangle) = \langle M_1, M_2\rangle \in \overline{EQ_{\mathsf{TM}}}$

- f halts.
- If M accepts w, $L(M_1) \neq L(M_2)$.
- If M rejects w,

The Reduction of A_{TM} to $\overline{EQ_{\text{TM}}}$

Let $f =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in \overline{EQ_{\text{TM}}}$

- f halts.
- If M accepts w, $L(M_1) \neq L(M_2)$.
- If M rejects w, $L(M_1) = L(M_2)$.

The Reduction of A_{TM} to $\overline{EQ_{\text{TM}}}$

Let f = "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   M_1 = "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   M_2 = "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in \overline{EQ_{\text{TM}}}$

- f halts.
- If M accepts w, $L(M_1) \neq L(M_2)$.
- If M rejects w, $L(M_1) = L(M_2)$.
- If M loops on w,

The Reduction of A_{TM} to $\overline{EQ_{\mathsf{TM}}}$

Let $f =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *reject*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\mathsf{TM}} \Leftrightarrow f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in \overline{EQ_{\mathsf{TM}}}$

- f halts.
- If M accepts w, $L(M_1) \neq L(M_2)$.
- If M rejects w, $L(M_1) = L(M_2)$.
- If M loops on w, $L(M_1) = L(M_2)$.

$\overline{EQ_{\text{TM}}}$ is not Turing-recognizable

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

$\overline{EQ_{\text{TM}}}$ is not Turing-recognizable

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

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

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

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

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

Proof strategy: Provide a reduction from A_{TM} to EQ_{TM}. Deduce that $\overline{A_{\mathsf{TM}}} \leq_m \overline{EQ_{\mathsf{TM}}}$. By corollary 4.23, $\overline{A_{\mathsf{TM}}}$ is not Turing-recognizable. By corollary 5.29, $\overline{EQ_{\mathsf{TM}}}$ is not Turing-recognizable.

Our task is to provide the reduction. That is, the computable function that mapping reduces A_{TM} to EQ_{TM}.

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow g(\langle M, w\rangle) = \langle M_1, M_2\rangle \in EQ_{\text{TM}}$

- f halts.

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow g(\langle M, w\rangle) = \langle M_1, M_2\rangle \in EQ_{\text{TM}}$

- f halts.
- If M accepts w,

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\mathsf{TM}} \Leftrightarrow g(\langle M, w\rangle) = \langle M_1, M_2\rangle \in EQ_{\mathsf{TM}}$

- f halts.
- If M accepts w, $L(M_1) = L(M_2)$.

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2 \rangle$."

Analysis: $\langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow g(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in EQ_{\text{TM}}$

- f halts.
- If M accepts w, $L(M_1) = L(M_2)$.
- If M rejects w,

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow g(\langle M, w\rangle) = \langle M_1, M_2\rangle \in EQ_{\text{TM}}$

- f halts.
- If M accepts w, $L(M_1) = L(M_2)$.
- If M rejects w, $L(M_1) \neq L(M_2)$.

The Reduction of A_{TM} to EQ_{TM}

Let $g =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\mathsf{TM}} \Leftrightarrow g(\langle M, w\rangle) = \langle M_1, M_2\rangle \in EQ_{\mathsf{TM}}$

- f halts.
- If M accepts w, $L(M_1) = L(M_2)$.
- If M rejects w, $L(M_1) \neq L(M_2)$.
- If M loops on w,

The Reduction of A_{TM} to EQ_{TM}

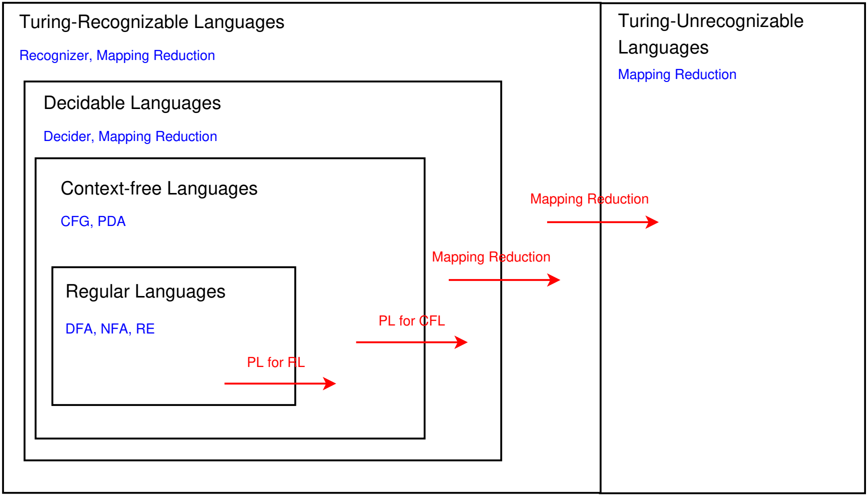
Let $g =$ "On input $\langle M, w\rangle$:

1. Construct the following machine M_1.
   $M_1 =$ "On input x:
   1. *accept*."
2. Construct the following machine M_2.
   $M_2 =$ "On input x:
   1. Run M on w. If M accepts w, *accept*; otherwise *reject*."
3. Output $\langle M_1, M_2\rangle$."

Analysis: $\langle M, w\rangle \in A_{\mathsf{TM}} \Leftrightarrow g(\langle M, w\rangle) = \langle M_1, M_2\rangle \in EQ_{\mathsf{TM}}$

- f halts.
- If M accepts w, $L(M_1) = L(M_2)$.
- If M rejects w, $L(M_1) \neq L(M_2)$.
- If M loops on w, $L(M_1) \neq L(M_2)$.

Summary


Turing-Recognizable Languages
Recognizer, Mapping Reduction
Decidable Languages
Decider, Mapping Reduction
Context-free Languages
CFG, PDA
Regular Languages
DFA, NFA, RE
PL for RL
PL for CFL
Mapping Reduction
Mapping Reduction
Turing-Unrecognizable Languages
Mapping Reduction

- **THEOREM 5.1** $HALT_{\mathsf{TM}}$ is undecidable.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.

- **THEOREM 5.1** $HALT_{\mathsf{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\mathsf{TM}}$ is undecidable.

- **Theorem 5.1** $HALT_{\text{TM}}$ is undecidable.
- **Theorem 5.2** E_{TM} is undecidable.
- **Theorem 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **Theorem 5.4** EQ_{TM} is undecidable.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.
- **DEFINITION 5.20** mapping reducible.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.
- **DEFINITION 5.20** mapping reducible.
- **THEOREM 5.22** If $A \leq_m B$ and B is decidable, then A is decidable.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.
- **DEFINITION 5.20** mapping reducible.
- **THEOREM 5.22** If $A \leq_m B$ and B is decidable, then A is decidable.
- **COROLLARY 5.23** If $A \leq_m B$ and A is undecidable, then B is undecidable.

- **THEOREM 5.1** $HALT_{\mathsf{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\mathsf{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.
- **DEFINITION 5.20** mapping reducible.
- **THEOREM 5.22** If $A \leq_m B$ and B is decidable, then A is decidable.
- **COROLLARY 5.23** If $A \leq_m B$ and A is undecidable, then B is undecidable.
- **THEOREM 5.28** If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.
- **DEFINITION 5.20** mapping reducible.
- **THEOREM 5.22** If $A \leq_m B$ and B is decidable, then A is decidable.
- **COROLLARY 5.23** If $A \leq_m B$ and A is undecidable, then B is undecidable.
- **THEOREM 5.28** If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.
- **COROLLARY 5.29** If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

- **THEOREM 5.1** $HALT_{\text{TM}}$ is undecidable.
- **THEOREM 5.2** E_{TM} is undecidable.
- **THEOREM 5.3** $REGULAR_{\text{TM}}$ is undecidable.
- **THEOREM 5.4** EQ_{TM} is undecidable.
- **DEFINITION 5.17** computable function.
- **DEFINITION 5.20** mapping reducible.
- **THEOREM 5.22** If $A \leq_m B$ and B is decidable, then A is decidable.
- **COROLLARY 5.23** If $A \leq_m B$ and A is undecidable, then B is undecidable.
- **THEOREM 5.28** If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.
- **COROLLARY 5.29** If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.
- **THEOREM 5.30** If EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.