Reducibility
Part II

Problem Set 7 due in the box up front.
The General Pattern

$H = \text{“On input } w:\text{ “}$

- Transform the input $w$ into $f(w)$.
- Run machine $R$ on $f(w)$.
- If $R$ accepts $f(w)$, then $H$ accepts $w$.
- If $R$ rejects $f(w)$, then $H$ rejects $w$."

Machine $H$

Machine $R$

Compute $f$

$w$
Defining Reductions

A reduction from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$
Defining Reductions

• A **reduction** from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

  For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$
Defining Reductions

- A **reduction** from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

\[
\text{For any } w \in \Sigma_1^*, \; w \in A \text{ iff } f(w) \in B
\]
Defining Reductions

- A **reduction** from $A$ to $B$ is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

  For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$

- Every $w \in A$ maps to some $f(w) \in B$.
- Every $w \notin A$ maps to some $f(w) \notin B$.
- $f$ does not have to be injective or surjective.
$w \in A \iff f(w) \in B$

**Machine $H$**

- On input $w$:
  - Transform the input $w$ into $f(w)$.
  - Run machine $R$ on $f(w)$.
  - If $R$ accepts $f(w)$, then $H$ accepts $w$.
  - If $R$ rejects $f(w)$, then $H$ rejects $w$.

**Machine $R$**

- Accepts $f(w)$ if $f(w) \in B$.

$H$ accepts $w$ iff $R$ accepts $f(w)$ iff $f(w) \in B$ iff $w \in A$
Mapping Reductions

- A function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is called a **mapping reduction** from $A$ to $B$ iff
  - For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$.
  - $f$ is a computable function.
- Intuitively, a mapping reduction from $A$ to $B$ says that a computer can transform any instance of $A$ into an instance of $B$ such that the answer to $B$ is the answer to $A$. 
Mapping Reducibility

• If there is a mapping reduction from language $A$ to language $B$, we say that language $A$ is **mapping reducible** to language $B$.

• Notation: $A \leq_m B$ iff language $A$ is mapping reducible to language $B$.

• Note that we reduce *languages*, not *machines*. 
Why Mapping Reducibility Matters

- **Theorem**: If \( B \in R \) and \( A \leq_{M} B \), then \( A \in R \).
- **Theorem**: If \( B \in \text{RE} \) and \( A \leq_{M} B \), then \( A \in \text{RE} \).
- **Theorem**: If \( B \in \text{co-RE} \) and \( A \leq_{M} B \), then \( A \in \text{co-RE} \).
- **Intuitively**: \( A \leq_{M} B \) means “\( A \) is not harder than \( B \).”
Why Mapping Reducibility Matters

- **Theorem**: If $A \notin R$ and $A \leq^M B$, then $B \notin R$.

- **Theorem**: If $A \notin RE$ and $A \leq^M B$, then $B \notin RE$.

- **Theorem**: If $A \notin co\text{-}RE$ and $A \leq^M B$, then $B \notin co\text{-}RE$.

- *Intuitively*: $A \leq^M B$ means “$B$ is at least as hard as $A$.”
Why Mapping Reducibility Matters

If this one is "easy" $(R, RE, co-RE)$...

$A \leq_M B$

... then this one is "easy" $(R, RE, co-RE)$ too.
Why Mapping Reducibility Matters

If this one is “hard”
(not R, not RE, or not co–RE)...

\[ A \leq_M B \]

... then this one is “hard” (not R, not RE, or not co–RE) too.
Using Mapping Reductions
Revisiting our Proofs

• Consider the language

\[ L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \varepsilon \} \]

• We have already proven that this language is in \( \text{RE} \) by building a TM for it.

• Let's repeat this proof using mapping reductions.

• Specifically, we will prove

\[ L \leq_{M} A_{\text{TM}} \]
$L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \varepsilon \} \$

- To prove $L \leq_A^{TM} A_{TM}$, we will need to find a computable function $f$ such that
  \[ \langle M \rangle \in L \iff f(\langle M \rangle) \in A_{TM} \]

- Since $A_{TM}$ is a language of TM/string pairs, let's assume $f(\langle M \rangle) = \langle N, w \rangle$ for some TM $N$ and string $w$ (which we'll pick later):
  \[ \langle M \rangle \in L \iff \langle N, w \rangle \in A_{TM} \]

- Substituting definitions:
  \[ M \text{ accepts } \varepsilon \iff N \text{ accepts } w \]

- Choose $N = M$, $w = \varepsilon$. So $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. 

One Interpretation of the Reduction

\[ \langle M \rangle \xrightarrow{\text{Compute } f} \langle M, \varepsilon \rangle \xrightarrow{\text{Recognizer for } A_{TM}} \]

Machine \( H \)

Machine \( R \)

YES

NO
One Interpretation of the Reduction

\[ f(\langle M, \varepsilon \rangle) = \text{Accept} \]

Machine \( H \)

\[ H = \text{"On input } \langle M \rangle \text{:} \]
\[ \text{\quad \bullet \text{ Run machine } R \text{ on } } \langle M, \varepsilon \rangle. \]
\[ \text{\quad \bullet \text{ If } R \text{ accepts } \langle M, \varepsilon \rangle, \text{ then } H \text{ accepts } w.} \]
\[ \text{\quad \bullet \text{ If } R \text{ rejects } \langle M, \varepsilon \rangle, \text{ then } H \text{ rejects } w."} \]
Compute \( f \langle M \rangle \) and \( f \langle M, \varepsilon \rangle \) for machine \( R \).

Machine \( H \) accepts \( \langle M \rangle \) if:

- Run machine \( R \) on \( \langle M, \varepsilon \rangle \).
- If \( R \) accepts \( \langle M, \varepsilon \rangle \), then \( H \) accepts \( w \).
- If \( R \) rejects \( \langle M, \varepsilon \rangle \), then \( H \) rejects \( w \).
One Interpretation of the Reduction

\[ H = \text{"On input } \langle M \rangle \text{:} \]

\[ \begin{align*}
\cdot & \text{ Run machine } R \text{ on } \langle M, \varepsilon \rangle. \\
\cdot & \text{ If } R \text{ accepts } \langle M, \varepsilon \rangle, \text{ then } H \text{ accepts } w. \\
\cdot & \text{ If } R \text{ rejects } \langle M, \varepsilon \rangle, \text{ then } H \text{ rejects } w. 
\end{align*} \]

\[ H \text{ accepts } \langle M \rangle \iff R \text{ accepts } \langle M, \varepsilon \rangle \]
One Interpretation of the Reduction

Machine $H$

$H =$ “On input $\langle M \rangle$:

- Run machine $R$ on $\langle M, \varepsilon \rangle$.
- If $R$ accepts $\langle M, \varepsilon \rangle$, then $H$ accepts $w$.
- If $R$ rejects $\langle M, \varepsilon \rangle$, then $H$ rejects $w$.”

$H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \varepsilon \rangle$ iff $M$ accepts $\varepsilon$
One Interpretation of the Reduction

\[ \langle M \rangle \rightarrow \text{Compute } f \rightarrow \langle M, \varepsilon \rangle \rightarrow \text{Recognizer for } A_{TM} \]

Machine \( H \)

\[ H = \text{“On input } \langle M \rangle: \]

\[ \quad \cdot \text{Run machine } R \text{ on } \langle M, \varepsilon \rangle. \]

\[ \quad \cdot \text{If } R \text{ accepts } \langle M, \varepsilon \rangle, \text{ then } H \text{ accepts } w. \]

\[ \quad \cdot \text{If } R \text{ rejects } \langle M, \varepsilon \rangle, \text{ then } H \text{ rejects } w. \]

\[ \]

\[ H \text{ accepts } \langle M \rangle \]

\[ \iff \]

\[ R \text{ accepts } \langle M, \varepsilon \rangle \]

\[ \iff \]

\[ M \text{ accepts } \varepsilon \]

\[ \iff \]

\[ \langle M \rangle \in L \]
One Interpretation of the Reduction

Machine $H$

$H = \text{"On input } \langle M \rangle: \text{"

$\cdot$ Run machine $R$ on $\langle M, \varepsilon \rangle$.

$\cdot$ If $R$ accepts $\langle M, \varepsilon \rangle$, then $H$ accepts $w$.

$\cdot$ If $R$ rejects $\langle M, \varepsilon \rangle$, then $H$ rejects $w$.”

$H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \varepsilon \rangle$ iff $M$ accepts $\varepsilon$ iff $\langle M \rangle \in L$
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE} \).
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE}. \)

**Proof:** We will prove that \( L \leq_{\text{M}} A_{\text{TM}}. \)
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

Theorem: \( L \in \text{RE} \).

Proof: We will prove that \( L \leq^m A_{\text{TM}} \). Since \( A_{\text{TM}} \in \text{RE} \), this proves \( L \in \text{RE} \) as well.
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE} \).

**Proof:** We will prove that \( L \leq_{m} A_{TM} \). Since \( A_{TM} \in \text{RE} \), this proves \( L \in \text{RE} \) as well.

To prove this, we will give a mapping reduction from \( L \) to \( A_{TM} \).
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE}. \)

**Proof:** We will prove that \( L \leq_M A_{\text{TM}}. \) Since \( A_{\text{TM}} \in \text{RE}, \) this proves \( L \in \text{RE} \) as well.

To prove this, we will give a mapping reduction from \( L \) to \( A_{\text{TM}}. \) For any TM \( M, \) let \( f(\langle M \rangle) = \langle M, \varepsilon \rangle. \) This function can be computed by a Turing machine.
$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$

**Theorem:** $L \in \text{RE}$.

**Proof:** We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \text{RE}$, this proves $L \in \text{RE}$ as well.

To prove this, we will give a mapping reduction from $L$ to $A_{\text{TM}}$. For any TM $M$, let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that $f$ is a mapping reduction by proving for all TMs $M$ that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. 

\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE} \).

**Proof:** We will prove that \( L \leq_{M} A_{\text{TM}} \). Since \( A_{\text{TM}} \in \text{RE} \), this proves \( L \in \text{RE} \) as well.

To prove this, we will give a mapping reduction from \( L \) to \( A_{\text{TM}} \). For any TM \( M \), let \( f(\langle M \rangle) = \langle M, \varepsilon \rangle \). This function can be computed by a Turing machine.

Now, we will prove that \( f \) is a mapping reduction by proving for all TMs \( M \) that \( \langle M \rangle \in L \) iff \( \langle M, \varepsilon \rangle \in A_{\text{TM}} \).

To do this, consider any TM \( M \).
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE} \).

**Proof:** We will prove that \( L \leq_M A_{TM} \). Since \( A_{TM} \in \text{RE} \), this proves \( L \in \text{RE} \) as well.

To prove this, we will give a mapping reduction from \( L \) to \( A_{TM} \). For any TM \( M \), let \( f(\langle M \rangle) = \langle M, \varepsilon \rangle \). This function can be computed by a Turing machine.

Now, we will prove that \( f \) is a mapping reduction by proving for all TMs \( M \) that \( \langle M \rangle \in L \) iff \( \langle M, \varepsilon \rangle \in A_{TM} \).

To do this, consider any TM \( M \). Note that by the definition of \( L \), we see \( \langle M \rangle \in L \) iff \( M \) accepts \( \varepsilon \).
Theorem: $L \in \text{RE}$.
Proof: We will prove that $L \leq_{\text{M}} A_{\text{TM}}$. Since $A_{\text{TM}} \in \text{RE}$, this proves $L \in \text{RE}$ as well.

To prove this, we will give a mapping reduction from $L$ to $A_{\text{TM}}$. For any TM $M$, let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that $f$ is a mapping reduction by proving for all TMs $M$ that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. To do this, consider any TM $M$. Note that by the definition of $L$, we see $\langle M \rangle \in L$ iff $M$ accepts $\varepsilon$. By the definition of $A_{\text{TM}}$, we know that $M$ accepts $\varepsilon$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. 

$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$
Theorem: $L \in \text{RE}$.

Proof: We will prove that $L \leq_A A_{\text{TM}}$. Since $A_{\text{TM}} \in \text{RE}$, this proves $L \in \text{RE}$ as well.

To prove this, we will give a mapping reduction from $L$ to $A_{\text{TM}}$. For any TM $M$, let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that $f$ is a mapping reduction by proving for all TMs $M$ that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. To do this, consider any TM $M$. Note that by the definition of $L$, we see $\langle M \rangle \in L$ iff $M$ accepts $\varepsilon$. By the definition of $A_{\text{TM}}$, we know that $M$ accepts $\varepsilon$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. Combining these statements together, we have that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. 

$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE} \).

**Proof:** We will prove that \( L \leq_{M} A_{TM} \). Since \( A_{TM} \in \text{RE} \), this proves \( L \in \text{RE} \) as well.

To prove this, we will give a mapping reduction from \( L \) to \( A_{TM} \). For any TM \( M \), let \( f(\langle M \rangle) = \langle M, \varepsilon \rangle \). This function can be computed by a Turing machine.

Now, we will prove that \( f \) is a mapping reduction by proving for all TMs \( M \) that \( \langle M \rangle \in L \) iff \( \langle M, \varepsilon \rangle \in A_{TM} \).

To do this, consider any TM \( M \). Note that by the definition of \( L \), we see \( \langle M \rangle \in L \) iff \( M \) accepts \( \varepsilon \). By the definition of \( A_{TM} \), we know that \( M \) accepts \( \varepsilon \) iff \( \langle M, \varepsilon \rangle \in A_{TM} \). Combining these statements together, we have that \( \langle M \rangle \in L \) iff \( \langle M, \varepsilon \rangle \in A_{TM} \).

This means that \( f \) is a mapping reduction from \( L \) to \( A_{TM} \), so \( L \leq_{M} A_{TM} \), as required.
\[ L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \} \]

**Theorem:** \( L \in \text{RE} \).

**Proof:** We will prove that \( L \leq_M A_{\text{TM}} \). Since \( A_{\text{TM}} \in \text{RE} \), this proves \( L \in \text{RE} \) as well.

To prove this, we will give a mapping reduction from \( L \) to \( A_{\text{TM}} \). For any TM \( M \), let \( f(\langle M \rangle) = \langle M, \varepsilon \rangle \). This function can be computed by a Turing machine.

Now, we will prove that \( f \) is a mapping reduction by proving for all TMs \( M \) that \( \langle M \rangle \in L \) iff \( \langle M, \varepsilon \rangle \in A_{\text{TM}} \). To do this, consider any TM \( M \). Note that by the definition of \( L \), we see \( \langle M \rangle \in L \) iff \( M \) accepts \( \varepsilon \). By the definition of \( A_{\text{TM}} \), we know that \( M \) accepts \( \varepsilon \) iff \( \langle M, \varepsilon \rangle \in A_{\text{TM}} \). Combining these statements together, we have that \( \langle M \rangle \in L \) iff \( \langle M, \varepsilon \rangle \in A_{\text{TM}} \).

This means that \( f \) is a mapping reduction from \( L \) to \( A_{\text{TM}} \), so \( L \leq_M A_{\text{TM}} \), as required. ■
What Did We Prove?

\[ H = \text{"On input } \langle M \rangle:\]
\begin{itemize}
  \item Run machine \( R \) on \( \langle M, \varepsilon \rangle \).
  \item If \( R \) accepts \( \langle M, \varepsilon \rangle \), then \( H \) accepts \( w \).
  \item If \( R \) rejects \( \langle M, \varepsilon \rangle \), then \( H \) rejects \( w \)."
\end{itemize}

\[ H \text{ accepts } \langle M \rangle \iff R \text{ accepts } \langle M, \varepsilon \rangle \iff M \text{ accepts } \varepsilon \iff \langle M \rangle \in L \]
What Did We Prove?

**Machine** $H$

$H = \text{“On input } \langle M \rangle:$$$
\begin{align*}
\cdot \text{ Run machine } R \text{ on } \langle M, \varepsilon \rangle. \\
\cdot \text{ If } R \text{ accepts } \langle M, \varepsilon \rangle, \text{ then } H \text{ accepts } w. \\
\cdot \text{ If } R \text{ rejects } \langle M, \varepsilon \rangle, \text{ then } H \text{ rejects } w."
\end{align*}$
Interpreting Mapping Reductions

• If $A \leq^M B$, there is a known construction to turn a TM for $B$ into a TM for $A$.

• When doing proofs with mapping reductions, you do not need to show the overall construction.

• You just need to prove that
  • $f$ is a computable function, and
  • $w \in A$ iff $f(w) \in B$. 
Another Mapping Reduction
$L_D$ and $\overline{A}_{TM}$

- Earlier, we proved $\overline{A}_{TM} \not\in \text{RE}$ by proving that
  
  If $\overline{A}_{TM} \in \text{RE}$, then $L_D \in \text{RE}$.

- The proof constructed this TM, assuming $R$ was a recognizer for $\overline{A}_{TM}$.

\[
H = "\text{On input } \langle M \rangle:\n\text{• Construct the string } \langle M, \langle M \rangle \rangle.\n\text{• Run } R \text{ on } \langle M, \langle M \rangle \rangle.\n\text{• If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } \langle M \rangle.\n\text{• If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } \langle M \rangle."\n\]

- Let's do another proof using mapping reductions.
\[ L_D \leq_M \overline{A}_{\text{TM}} \]

- To prove that \( \overline{A}_{\text{TM}} \notin \text{RE} \), we will prove \( L_D \leq_M \overline{A}_{\text{TM}} \).

- By our earlier theorem, since \( L_D \notin \text{RE} \), we have that \( \overline{A}_{\text{TM}} \notin \text{RE} \).

- Intuitively: \( \overline{A}_{\text{TM}} \) is “at least as hard” as \( L_D \), and since \( L_D \notin \text{RE} \), this means \( \overline{A}_{\text{TM}} \notin \text{RE} \).
\[ L_D \leq_M \overline{A}_{TM} \]

- Goal: Find a computable function \( f \) such that
  \[ \langle M \rangle \in L_D \iff f(\langle M \rangle) \in \overline{A}_{TM} \]

- Simplifying this using the definition of \( L_D \)
  \[ M \text{ does not accept } \langle M \rangle \iff f(\langle M \rangle) \in \overline{A}_{TM} \]

- Let's assume that \( f(\langle M \rangle) \) has the form \( \langle N, w \rangle \) for some TM \( N \) and string \( w \). This means that
  \[ M \text{ does not accept } \langle M \rangle \iff \langle N, w \rangle \in \overline{A}_{TM} \]
  \[ M \text{ does not accept } \langle M \rangle \iff N \text{ does not accept } w \]

- If we can choose \( w \) and \( N \) such that the above is true, we will have our reduction from \( L_D \) to \( \overline{A}_{TM} \).

- Choose \( N = M \) and \( w = \langle M \rangle \).
One Interpretation of the Reduction

Compute $f$ for $\langle M, \langle M \rangle \rangle$ on Machine $H$ 

Recognizer for $\overline{A}_{TM}$ on Machine $R$
One Interpretation of the Reduction

Compute $f$ in $\langle M, \langle M \rangle \rangle$ and pass it to the recognizer for $\overline{A_{TM}}$.

Machine $H$

- On input $\langle M \rangle$:
  - Run machine $R$ on $\langle M, \langle M \rangle \rangle$.
  - If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $w$.
  - If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $w$.

$H = \text{"On input } \langle M \rangle:\$

- Run machine $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $w$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $w$.\"
One Interpretation of the Reduction

\[ H = \text{"On input } \langle M \rangle \text{:} \]
\[ \quad \text{• Run machine } R \text{ on } \langle M, \langle M \rangle \rangle. \]
\[ \quad \text{• If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } w. \]
\[ \quad \text{• If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } w." \]
One Interpretation of the Reduction

Compute $f$ for $\langle M \rangle$

Recognizer for $\overline{A_{TM}}$

Machine $R$

Machine $H$

$H = \text{"On input } \langle M \rangle :$
- Run machine $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $w$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $w$.

$H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$
One Interpretation of the Reduction

**Machine H**

\[ H = \text{“On input } \langle M \rangle \text{:} \]
- Run machine \( R \) on \( \langle M, \langle M \rangle \rangle \).
- If \( R \) accepts \( \langle M, \langle M \rangle \rangle \), then \( H \) accepts \( w \).
- If \( R \) rejects \( \langle M, \langle M \rangle \rangle \), then \( H \) rejects \( w \).\]

**Recognizer for \( \overline{A_{TM}} \)**

\[ H \text{ accepts } \langle M \rangle \text{ iff } R \text{ accepts } \langle M, \langle M \rangle \rangle \text{ iff } M \text{ does not accept } \langle M \rangle \]
One Interpretation of the Reduction

\[ H = \text{"On input } \langle M \rangle \text{:} \]
\[ \quad \text{• Run machine } R \text{ on } \langle M, \langle M \rangle \rangle. \]
\[ \quad \text{• If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } \]
\[ \quad \quad H \text{ accepts } w. \]
\[ \quad \text{• If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } \]
\[ \quad \quad H \text{ rejects } w. \]
One Interpretation of the Reduction

\[ H = \text{"On input } \langle M \rangle: \]
\[ \quad \text{Run machine } R \text{ on } \langle M, \langle M \rangle \rangle. \]
\[ \quad \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } w. \]
\[ \quad \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } w. \]
**Theorem:** $\overline{A_{TM}} \notin \text{RE}.$

**Proof:** We will prove that $L_D \leq_M \overline{A_{TM}}$. Since $L_D \notin \text{RE}$, this proves that $\overline{A_{TM}} \notin \text{RE}$.

To show that $L_D \leq_M \overline{A_{TM}}$, we will give a mapping reduction from $L_D$ to $\overline{A_{TM}}$. For any TM $M$, let $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$. This function $f$ is computable.

To prove that $f$ is a mapping reduction from $L_D$ to $\overline{A_{TM}}$, we will prove for all TMs $M$ that $\langle M \rangle \in L_D$ iff $\langle M, \langle M \rangle \rangle \in \overline{A_{TM}}$. By the definition of $L_D$, we know $\langle M \rangle \in L_D$ iff $M$ does not accept $\langle M \rangle$. Similarly, by definition of $\overline{A_{TM}}$, we know that $M$ does not accept $\langle M \rangle$ iff $\langle M, \langle M \rangle \rangle \in \overline{A_{TM}}$. Combining these statements together, we see $\langle M \rangle \in L_D$ iff $\langle M, \langle M \rangle \rangle \in \overline{A_{TM}}$. Thus $f$ is a mapping reduction from $L_D$ to $\overline{A_{TM}}$, so $L_D \leq \overline{A_{TM}}$, as required. ■
The Amplifier Machine
As we've seen, Turing machines can run other Turing machines as subroutines.
In order to reduce certain problems to one another, it is useful / necessary to embed Turing machines inside of one another.
  - We'll see an example in a second.
One construction, in particular, is useful for reductions like these.
For any TM $M$ and string $w$, let $\text{Amp}(M, w)$ be this TM:

\begin{align*}
\text{Amp}(M, w) &= \text{“On input } x: \\
& \quad \text{Ignore } x. \\
& \quad \text{Run } M \text{ on } w. \\
& \quad \text{If } M \text{ accepts } w, \text{ then } \text{Amp}(M, w) \text{ accepts } x. \\
& \quad \text{If } M \text{ rejects } w, \text{ then } \text{Amp}(M, w) \text{ rejects } x. \\
\end{align*}

Theorem 1: If $M$ accepts $w$, then $\text{Amp}(\mathcal{L} M, w) = \Sigma^*$. If $M$ does not accept $w$, then $\text{Amp}(\mathcal{L} M, w) = \emptyset$.

Corollary 1: $M$ accepts $w$ iff $\text{Amp}(\mathcal{L} M, w) = \Sigma^*$.

Corollary 2: $M$ does not accept $w$ iff $\text{Amp}(\mathcal{L} M, w) = \emptyset$.

Theorem 2: The function $f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle$ is computable.
The Amplifier Machine

For any TM $M$ and string $w$, let $\text{Amp}(M, w)$ be this TM:

$\text{Amp}(M, w) = \text{“On input } x: \text{ Ignore } x. \text{ Run } M \text{ on } w. \text{ If } M \text{ accepts } w, \text{ then } \text{Amp}(M, w) \text{ accepts } x. \text{ If } M \text{ rejects } w, \text{ then } \text{Amp}(M, w) \text{ rejects } x.”$
For any TM $M$ and string $w$, let $\text{Amp}(M, w)$ be this TM:

$\text{Amp}(M, w) = \text{“On input } x:\text{ Ignore } x.\text{ Run } M \text{ on } w.\text{ If } M \text{ accepts } w, \text{ then } \text{Amp}(M, w) \text{ accepts } x.\text{ If } M \text{ rejects } w, \text{ then } \text{Amp}(M, w) \text{ rejects } x.”$ 

**Theorem 1:** If $M$ accepts $w$, then $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. If $M$ does not accept $w$, then $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

**Corollary 1:** $M$ accepts $w$ iff $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$

**Corollary 2:** $M$ does not accept $w$ iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$. 

The Amplifier Machine
For any TM $M$ and string $w$, let $\text{Amp}(M, w)$ be the following TM:

$\text{Amp}(M, w) = "\text{On input } x:\n\text{Ignore } x.\n\text{Run } M \text{ on } w.\n\text{If } M \text{ accepts } w, \text{ then } \text{Amp}(M, w) \text{ accepts } x.\n\text{If } M \text{ rejects } w, \text{ then } \text{Amp}(M, w) \text{ rejects } x." "$

**Theorem:** If $M$ accepts $w$, then $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. If $M$ does not accept $w$, then $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

**Proof:** First, we consider what happens if $M$ accepts $w$. In this case, consider what happens when we run $\text{Amp}(M, w)$ on an arbitrary input string $x$. $\text{Amp}(M, w)$ will run $M$ on $w$, and since $M$ accepts $w$, $\text{Amp}(M, w)$ accepts $x$. Since our choice of $x$ was arbitrary, we see that $\text{Amp}(M, w)$ accepts any input, so $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$.

Otherwise, $M$ does not accept $w$, so $M$ rejects $w$ or $M$ loops on $w$. Consider the result of running $\text{Amp}(M, w)$ on an arbitrary string $x$. If $M$ rejects $w$, then $\text{Amp}(M, w)$ rejects $x$. Otherwise, $\text{Amp}(M, w)$ loops on $x$. In both cases, $\text{Amp}(M, w)$ doesn't accept $x$. Since our choice of $x$ was arbitrary, we see that $\text{Amp}(M, w)$ never accepts any input, so $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$. ■
The Amplifier Machine

For any TM $M$ and string $w$, let $\text{Amp}(M, w)$ be this TM:

$$\text{Amp}(M, w) = \text{"On input } x:\n\text{Ignore } x.\n\text{Run } M \text{ on } w.\n\text{If } M \text{ accepts } w, \text{ then Amp}(M, w) \text{ accepts } x.\n\text{If } M \text{ rejects } w, \text{ then Amp}(M, w) \text{ rejects } x."
$$

**Theorem 1:** If $M$ accepts $w$, then $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. If $M$ does not accept $w$, then $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

**Corollary 1:** $M$ accepts $w$ iff $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$

**Corollary 2:** $M$ does not accept $w$ iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

**Theorem 2:** The function $f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle$ is computable.
\begin{center}
\begin{tikzpicture}

\node[shape=circle,draw=black,thick] (start) at (0,0) {$q_{\text{start}}$};
\node[shape=circle,draw=red,thick] (rej) at (2,0) {$q_{\text{rej}}$};
\node[shape=circle,draw=green,thick] (acc) at (2,1) {$q_{\text{acc}}$};
\node at (2.5,0) {$M$};

\draw[->,dashed] (start) -- (rej);
\draw[->,dashed] (rej) -- (acc);
\draw[->,dashed] (start) -- (acc);

\draw[->,thick] (start) -- (start) node[midway,above] {start};

\end{tikzpicture}
\end{center}
“On input $x$:
  · Ignore $x$.
  · Run $M$ on $w$.
  · If $M$ accepts $w$, we accept $x$.
  · If $M$ rejects $w$, we reject $x$.\)
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

![Diagram of a Turing machine]
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$."

Hypothetically, assume that $w$ is the string $1101$. 
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

```
... 1 0 1 0 ...
```
“On input \( x \):
- Ignore \( x \).
- Run \( M \) on \( w \).
- If \( M \) accepts \( w \), we accept \( x \).
- If \( M \) rejects \( w \), we reject \( x \).”

Hypothetically, assume that \( w \) is the string 1101.
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

```
... 1 0 1 0 ...
```
“On input $x$:  
- Ignore $x$.  
- Run $M$ on $w$.  
- If $M$ accepts $w$, we accept $x$.  
- If $M$ rejects $w$, we reject $x$."

Hypothetically, assume that $w$ is the string $1101$.  

![Diagram](image)
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string **1101**.
“On input $x$: 
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

---

```
... 0 1 0 ...
```
“On input $x$:
• Ignore $x$.
• Run $M$ on $w$.
• If $M$ accepts $w$, we accept $x$.
• If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

---

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>1</th>
<th>0</th>
<th></th>
</tr>
</thead>
</table>

---

$0 \rightarrow \square, R$
$1 \rightarrow \square, R$

---

$M$

---

$q_{\text{start}}$  $q_{\text{acc}}$  $q_{\text{rej}}$
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$."

Hypothetically, assume that $w$ is the string $1101$. 

\[ \begin{array}{c}
\ldots & \ldots & 0 & \ldots \\
\end{array} \]
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

... ... ... 

$M$

$q_{\text{start}}$ $q_{\text{acc}}$ $q_{\text{rej}}$

$0 \rightarrow \square, R$
$1 \rightarrow \square, R$

Erase
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$."

Hypothetically, assume that $w$ is the string $1101$. 
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 
“On input $x$:
  - Ignore $x$.
  - Run $M$ on $w$.
  - If $M$ accepts $w$, we accept $x$.
  - If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

---

```

<table>
<thead>
<tr>
<th>Start</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>... 1 ...</td>
</tr>
</tbody>
</table>
```

---

Diagram of $M$:
- $q_{\text{start}}$ (start state)
- $q_{\text{acc}}$ (accept state)
- $q_{\text{rej}}$ (reject state)

Transitions:
- $0 \rightarrow \square, R$
- $1 \rightarrow \square, R$
- $\square \rightarrow 1, R$
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. ❄️

---

```
... 1   ... 
```
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

Hypothesized accepting run:

$q_{\text{start}}$ -> $q_{1101}$ -> $q_{110}$ -> $q_{\text{acc}}$

Hypothesized rejecting run:

$q_{\text{start}}$ -> $q_{1101}$ -> $q_{110}$ -> $q_{\text{rej}}$
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 
"On input x:
- Ignore x.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.

Hypothetically, assume that $w$ is the string $1101$.\[\begin{array}{cccccc}
\ldots & 1 & 1 & 0 & 1 & \ldots \\
\end{array}\]
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string $1101$. 

---

### Diagram Explanation

- **Start State**: $q_{\text{start}}$
- **Accept State**: $q_{\text{acc}}$
- **Reject State**: $q_{\text{rej}}$

- **Transitions**:
  - $0 \rightarrow \square, R$
  - $1 \rightarrow \square, R$
  - $0 \rightarrow 0, L$
  - $1 \rightarrow 1, L$
  - $\square \rightarrow 1, R$
  - $\square \rightarrow 0, R$
  - $\square \rightarrow \square, L$

- **Input String**: $1101$

---

### Transition Table

<table>
<thead>
<tr>
<th>Input</th>
<th>Action</th>
<th>Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$\rightarrow$</td>
<td>$R$</td>
</tr>
<tr>
<td>$1$</td>
<td>$\rightarrow$</td>
<td>$R$</td>
</tr>
<tr>
<td>$0$</td>
<td>$\rightarrow$</td>
<td>$L$</td>
</tr>
<tr>
<td>$1$</td>
<td>$\rightarrow$</td>
<td>$L$</td>
</tr>
<tr>
<td>$\square$</td>
<td>$\rightarrow$</td>
<td>$R$</td>
</tr>
<tr>
<td>$\square$</td>
<td>$\rightarrow$</td>
<td>$L$</td>
</tr>
</tbody>
</table>

---

### State Diagram

- **States**: $q_{\text{start}}$, $q_{\text{acc}}$, $q_{\text{rej}}$
- **Transitions**:
  - From $q_{\text{start}}$ to $1$ on $0$
  - From $1$ to $11$ on $0$
  - From $11$ to $1101$ on $1$
  - From $1101$ back to $1$ on $1$
  - From $1$ to $\square$ on $0$
  - From $\square$ to $\square$ on $1$
  - From $\square$ to $\square$ on $\square$

---

Hypothetically, assume that $w$ is the string $1101$. 

---

### Input String Visualization

- **Input String**: $1101$

---

The machine $M$ processes the input string $1101$ starting from the start state $q_{\text{start}}$ and ends in the accept state $q_{\text{acc}}$.
“On input $x$:
- Ignore $x$.
- Run $M$ on $w$.
- If $M$ accepts $w$, we accept $x$.
- If $M$ rejects $w$, we reject $x$.”

Hypothetically, assume that $w$ is the string 1101.
Using the Amplifier
A More Elaborate Reduction

• Since $\overline{A_{TM}} \notin \text{RE}$, there is no algorithm for determining whether a TM will not accept a given string.

• Could we check instead whether a TM never accepts a string?

• Consider the language
  
  $$L_e = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}$$

• How “hard” is $L_e$? Is it $\text{R}$, $\text{RE}$, $\text{co-RE}$, or none of these?
Building an Intuition

- Before we even try to prove how “hard” this language is, we should build an intuition for its difficulty.

- $L_e$ is *probably* not in $\text{RE}$, since if we were convinced a TM never accepted, it would be hard to find positive evidence of this.

- $L_e$ is *probably* in $\text{co-RE}$, since if we were convinced that a TM *did* accept some string, we could exhaustively search over all strings and try to find the string it accepts.

- Best guess: $L_e \in \text{co-RE} - \text{R}$. 
We will prove that $L_e \notin \text{RE}$ by showing that $\overline{A_{TM}} \leq_M L_e$. (This also proves $L_e \notin \text{R}$).

We want to find a function $f$ such that

$$\langle M, w \rangle \in \overline{A_{TM}} \iff f(\langle M, w \rangle) \in L_e$$

Since $L_e$ is a language of TM descriptions, let's assume $f(\langle M, w \rangle) = \langle N \rangle$ for some TM $N$. Then

$$\langle M, w \rangle \in \overline{A_{TM}} \iff \langle N \rangle \in L_e$$

Expanding out definitions, we get

$M$ doesn't accept $w$ iff $\mathcal{L}(N) = \emptyset$

How do we pick the machine $N$?
The Reduction

• Choose \( N \) such that this holds:

\[ M \text{ doesn't accept } w \text{ iff } \mathcal{L}(N) = \emptyset \]

• We can pick \( N = \text{Amp}(M, w) \).

• \textbf{Recall:} \( \mathcal{L}(\text{Amp}(M, w)) = \emptyset \) iff \( M \) doesn't accept \( w \).

• Since \( f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle \) is computable, this is the mapping reduction we need!
The Reduction

Machine for $L_e$
The Reduction

\[ \langle M, w \rangle \]

Construct $\text{Amp}(M, w)$

Machine for $L_e$
The Reduction

\[ \langle M, w \rangle \]

Construct Amp\((M, w)\)

Machine for \(L_e\)

Simulate \(M\) on \(w\) (Ignored)

\(x\)

\(\text{Amp}(M, w)\)
The Reduction

\( \langle M, w \rangle \) → Construct \( \text{Amp}(M, w) \) → Machine for \( L_e \)

\( \mathcal{L}(\text{Amp}(M, w)) = \Sigma^* \) if \( M \) accepts \( w \).

\( \mathcal{L}(\text{Amp}(M, w)) = \emptyset \) if \( M \) does not accept \( w \).

\( \chi \) → Simulate \( M \) on \( w \) → (Ignored) → \( \text{Amp}(M, w) \)
The Reduction

\[ \langle M, w \rangle \rightarrow \text{Construct Amp}(M, w) \rightarrow \langle \text{Amp}(M, w) \rangle \rightarrow \text{Machine for } L_e \]

\[ \langle M, w \rangle \rightarrow \text{Simulate } M \text{ on } w \rightarrow \text{Amp}(M, w) \]

(Ignored)
The Reduction

\[ \langle M, w \rangle \rightarrow \text{Construct Amp}(M, w) \rightarrow \langle \text{Amp}(M, w) \rangle \rightarrow \text{Machine for } L_e \]

Machine \( H \)

\[ \chi \rightarrow \text{Simulate } M \text{ on } w \rightarrow \text{(Ignored)} \rightarrow \text{Amp}(M, w) \]
The Reduction

What does $H$ do if $M$ does not accept $w$?

Simulate $M$ on $w$

(Ignored)

$\langle M, w \rangle$ $\rightarrow$ Construct $\text{Amp}(M, w)$ $\rightarrow$ $\langle \text{Amp}(M, w) \rangle$ $\rightarrow$ Machine for $L_e$

Machine $H$
The Reduction

Construct $\text{Amp}(M, w)$

(Amp $\langle M, w \rangle$)

Machine for $L_e$

(Never accepts)

Machine $H$

Simulate $M$ on $w$

(Ignored)

$\langle M, w \rangle$

What does $H$ do if $M$ does not accept $w$?
The Reduction

\[ \langle M, w \rangle \xrightarrow{\text{Construct Amp}(M, w)} \langle \text{Amp}(M, w) \rangle \xrightarrow{\text{(Never accepts)}} \text{Machine for } L_e \]

Machine \( H \)

Simulate \( M \) on \( w \)

(ignored)

What does \( H \) do if \( M \) does not accept \( w \)?
The Reduction

\[ \langle M, w \rangle \xrightarrow{\text{Construct} \ \operatorname{Amp}(M, w)} \langle \operatorname{Amp}(M, w) \rangle \xrightarrow{\text{Machine for } L_e} \]

\[ \text{Machine } H \]

\[ \langle M, w \rangle \xrightarrow{\text{Simulate } M \text{ on } w} (\text{Ignored}) \]

\[ \text{Amp}(M, w) \xrightarrow{\text{Simulate } M \text{ on } w} (\text{Ignored}) \]
The Reduction

What does $H$ do if $M$ accepts $w$?

Simulate $M$ on $w$

(Amp($M$, $w$))

Machine for $L_e$

Machine $H$

What does $H$ do if $M$ accepts $w$?
The Reduction

\[ \langle M, w \rangle \]

Construct \( \text{Amp}(M, w) \)

\( \langle \text{Amp}(M, w) \rangle \)

Machine for \( L_e \)

(Always accepts)

Machine \( H \)

Simulate \( M \) on \( w \)

\( x \)

(Ignored)

What does \( H \) do if \( M \) accepts \( w \)?
The Reduction

\( \langle M, w \rangle \) → Construct \( \text{Amp}(M, w) \) \( \langle \text{Amp}(M, w) \rangle \) → Machine for \( L_e \)

(Always accepts)

Machine \( H \)

Simulate \( M \) on \( w \)

(Ignored)

\( x \)

What does \( H \) do if \( M \) accepts \( w \)?

(or loop infinitely)
The Reduction

\[ \langle M, w \rangle \xrightarrow{\text{Construct}} \text{Amp}(M, w) \xrightarrow{\langle \text{Amp}(M, w) \rangle} \text{Machine for } L_e \]

\[ \chi \xrightarrow{\text{(Ignored)}} \text{Simulate } M \text{ on } w \]

Machine \( H \)
The Reduction

\( \langle M, w \rangle \)

Machine \( H \)

Simulate \( M \) on \( w \) (Ignored)

\( \text{Amp}(M, w) \)
The Reduction

What does $H$ do if $M$ does not accept $w$?

Machine $H$

Simulate $M$ on $w$

(Ignored)

What does $H$ do if $M$ does not accept $w$?

$\langle M, w \rangle$
The Reduction

What does $H$ do if $M$ does not accept $w$?

Simulate $M$ on $w$

$\langle M, w \rangle$

$Ignored$

Machine $H$

$\text{Amp}(M, w)$

What does $H$ do if $M$ does not accept $w$?
The Reduction

\[ \langle M, w \rangle \]

Machine \( H \)

Simulate \( M \) on \( w \)

(Ignored)

\( x \)

\( \text{Amp}(M, w) \)
The Reduction

What does $H$ do if $M$ accepts $w$?

Simulate $M$ on $w$

$\langle M, w \rangle$

Machine $H$

What does $H$ do if $M$ accepts $w$?

$Ignored$
The Reduction

What does $H$ do if $M$ accepts $w$?

Simulate $M$ on $w$

(Ignored)

What does $H$ do if $M$ accepts $w$?

(or loop infinitely)

Machine $H$

$\langle M, w \rangle$

$\text{Amp}(M, w)$

$x$
The Reduction

\[ \langle M, w \rangle \xrightarrow{\text{Construct Amp}(M, w)} \langle \text{Amp}(M, w) \rangle \xrightarrow{\text{Machine for } L_e} \]

Machine \( H \)

\[ x \xrightarrow{\text{Simulate M on } w} \xrightarrow{\text{Amp}(M, w)} \]

(Ignored)
The Reduction

\[ \langle M, w \rangle \]

Construct \( \text{Amp}(M, w) \)

\[ \langle \text{Amp}(M, w) \rangle \]

Machine for \( L_e \)

Machine \( H \)

Simulate \( M \) on \( w \)

(ignored)

\[ x \]

This is a recognizer for \( \overline{A_{TM}} \)!
**Theorem:** \( L_e \notin \text{RE} \)

**Proof:** We will prove \( \overline{A_{TM}} \leq_M L_e \). Since \( \overline{A_{TM}} \notin \text{RE} \), this proves that \( L_e \notin \text{RE} \), as required. To do so, we will exhibit a mapping reduction from \( \overline{A_{TM}} \) to \( L_e \). For any TM/string pair \( \langle M, w \rangle \), let \( f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle \). By our earlier theorem, this function is computable.

We claim this is a mapping reduction from \( \overline{A_{TM}} \) to \( L_e \). To prove this, we will prove that \( \langle M, w \rangle \in \overline{A_{TM}} \) iff \( \langle \text{Amp}(M, w) \rangle \in L_e \). By definition of \( \overline{A_{TM}} \), we see \( \langle M, w \rangle \) iff \( M \) does not accept \( w \). By our earlier theorem, \( M \) does not accept \( w \) iff \( \mathcal{L}(\text{Amp}(M, w)) = \emptyset \). Finally, by definition of \( L_e \), we see \( \mathcal{L}(\text{Amp}(M, w)) = \emptyset \) iff \( \langle \text{Amp}(M, w) \rangle \in L_e \). Taken together, we see that \( \langle M, w \rangle \in \overline{A_{TM}} \) iff \( \langle \text{Amp}(M, w) \rangle \in L_e \), so \( f \) is a mapping reduction from \( \overline{A_{TM}} \) to \( L_e \). Therefore, we see \( \overline{A_{TM}} \leq_M L_e \), as required. \( \square \)
A Math Joke
Time-Out For Announcements
Problem Set 6 Graded

• On-time Problem Set 6's have all been graded and should be returned after lecture today.
  • Online submissions: contact us if you don't hear back soon.
• Late Problem Set 6's will be returned this Wednesday.
Problem Set 8 Out

● Problem Set 8 goes out right now. It's due the Monday after Thanksgiving break (December 2).

● Some contradictory information:
  ● This is the last problem set on which you can use a late period.
  ● We *strongly* recommend that you don't, since you'll be pinched trying to finish Problem Set 9 if you do.

● TAs and I will figure out an OH schedule during Thanksgiving week.
Your Questions
“The fact we can't create a TM for $\overline{A_{TM}}$ and $L_D$ is very cool. But it is tough to see why we would want to solve those problems in the first place – what are problems that we actually want to solve but can't, because of limits of computability?”
“Aren't there some cases where we can know a TM is infinite looping? Couldn't we modify the $U$ so it keeps a record of IDs and then if it sees the same one twice know it was in a loop? This doesn't guarantee to find all loops, but would it be useful?”
“What's the difference between a language being decidable and having a decider for a language?”
“The generalized hailstone sequence terminating is proven to be undecidable (http://link.springer.com/chapter/10.1007%2F978-3-540-72504-6_49). What purpose is there to prove something as undecidable? Is undecidable better than not solvable?”
Back to CS103
The Limits of Computability

What's out here?
RE ∪ co-RE is Not Everything

- Using the same reasoning as the first day of lecture, we can show that there must be problems that are neither RE nor co-RE.
- There are more sets of strings than TMs.
- There are more sets of strings than twice the number of TMs.
- What do these languages look like?
There are infinitely many pairs of Turing machines with the same language as one another.

Good exercise: think about why this is.

Consider the following language:

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

Questions:

- Is \( EQ_{TM} \in \text{co-RE} \)?
- Is \( EQ_{TM} \in \text{RE} \)?
Is $EQ_{TM} \in \text{co-RE}$?

- Intuitively, would we expect $EQ_{TM}$ to be a co-$\text{RE}$ language?

- Suppose TM $M_1$ accepts a string $w$. We'd need to know whether $M_2$ accepts $w$ as well.

- Co-recognizing this would require us to have a corecognizer that detects whether $\langle M_2, w \rangle \in A_{TM}$, but that's not an co-$\text{RE}$ language!

- Our guess: $EQ_{TM}$ is probably not co-$\text{RE}$.
Proving $\text{EQ}_{TM} \notin \text{co-RE}$

- To prove that $\text{EQ}_{TM} \notin \text{co-RE}$, we can try to find a language $L$ where
  - $L \notin \text{co-RE}$, and
  - $L \leq_{M} \text{EQ}_{TM}$

- A good candidate would be something like $\text{A}_{TM}$, which is a “canonical” non-co-RE languages.

- **Goal:** Prove $\text{A}_{TM} \leq_{M} \text{EQ}_{TM}$. 
Proving $A_{TM} \leq_M EQ_{TM}$

- Goal: Find a computable function $f$ where
  \[
  \langle M, w \rangle \in A_{TM} \iff f(\langle M, w \rangle) \in EQ_{TM}
  \]

- Since $EQ_{TM}$ is a language of pairs of TMs, let's assume $f(\langle M \rangle) = \langle M_1, M_2 \rangle$. Then we want to pick $M_1$ and $M_2$ such that
  \[
  \langle M, w \rangle \in A_{TM} \iff \langle M_1, M_2 \rangle \in EQ_{TM}
  \]

- Substituting definitions, we want
  \[
  M \text{ accepts } w \iff \mathcal{L}(M_1) = \mathcal{L}(M_2)
  \]

- What do we do now?
Using the Amplifier

- We want

\[
M \text{ accepts } w \text{ iff } \mathcal{L}(M_1) = \mathcal{L}(M_2)
\]

- What happens if we pick \( M_1 \) to be \( \text{Amp}(M, w) \)?
  - If \( M \) accepts \( w \), then \( \mathcal{L}(M_1) = \Sigma^* \).
  - If \( M \) does not accept \( w \), then \( \mathcal{L}(M_1) = \emptyset \).

- Choose \( M_1 \) to be the amplifier machine and \( M_2 \) to be any TM with language \( \Sigma^* \).
  Then the above statement is true!
What's Going On?

- Suppose we have an oracle for $\text{EQ}_{\text{TM}}$.
- We want to know whether $M$ accepts $w$.
- To do this:
  - Find a TM $S$ we know has language $\Sigma^*$.
  - Ask the oracle “does TM $\text{Amp}(M, w)$ have the same language as TM $S$?”
  - If so, then $M$ accepts $w$.
  - If not, then $M$ does not accept $w$. 
**Theorem:** \( EQ_{TM} \not\in \text{co-RE}. \)

**Proof:** We will prove \( A_{TM} \leq_{M} EQ_{TM}. \) Since \( A_{TM} \not\in \text{co-RE}, \) this proves that \( EQ_{TM} \not\in \text{co-RE}. \) To show \( A_{TM} \leq_{M} EQ_{TM}, \) we will exhibit a mapping reduction from \( A_{TM} \) to \( EQ_{TM}. \)

For any TM/string pair \( \langle M, w \rangle, \) define \( f(\langle M, w \rangle) \) to be the pair of TMs \( \langle \text{Amp}(M, w), S \rangle, \) where \( S \) is the TM “On input x, accept x.” This function is computable, and note that \( \mathcal{L}(S) = \Sigma^*. \)

We claim that \( \langle M, w \rangle \in A_{TM} \) iff \( \langle \text{Amp}(M, w), E \rangle \in EQ_{TM}. \) To see this, note by definition of \( A_{TM} \) that \( \langle M, w \rangle \in A_{TM} \) iff \( M \) accepts \( w. \) By our earlier theorem, \( M \) accepts \( w \) iff \( \mathcal{L}(\text{Amp}(M, w)) = \Sigma^*. \) Since \( \mathcal{L}(S) = \Sigma^*, \) we see \( M \) accepts \( w \) iff \( \mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(S). \) Finally, by definition of \( EQ_{TM}, \( \mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(S) \) iff \( \langle \text{Amp}(M, w), S \rangle \in EQ_{TM}. \)

Collectively, we see \( \langle M, w \rangle \in A_{TM} \) iff \( \langle \text{Amp}(M, w), S \rangle \in EQ_{TM}. \)

Thus \( f \) is a mapping reduction from \( A_{TM} \) to \( EQ_{TM}, \) so \( A_{TM} \leq_{M} EQ_{TM}, \) as required. ■
Is $EQ_{TM} \in RE$?

- Intuitively, would we expect $EQ_{TM}$ to be a RE language?
- Suppose TM $M_1$ doesn't accept a string $w$. We'd need to know whether $M_2$ also doesn't accept $w$.
- Recognizing this would require us to have a recognizer that detects whether $\langle M_2, w \rangle \in \overline{A}_{TM}$, but that's not an RE language!
- Our guess: $EQ_{TM}$ is probably not RE.
Proving $\overline{A}_{TM} \leq_M EQ_{TM}$

- Goal: Find a computable function $f$ where
  \[ \langle M, w \rangle \in \overline{A}_{TM} \iff f(\langle M, w \rangle) \in EQ_{TM} \]

- Since $EQ_{TM}$ is a language of pairs of TMs, let's assume $f(\langle M \rangle) = \langle M_1, M_2 \rangle$. Then we want to pick $M_1$ and $M_2$ such that
  \[ \langle M, w \rangle \in \overline{A}_{TM} \iff \langle M_1, M_2 \rangle \in EQ_{TM} \]

- Substituting definitions, we want
  \[ M \text{ does not accept } w \iff \mathcal{L}(M_1) = \mathcal{L}(M_2) \]

- What do we do now?
Using the Amplifier

- We want

  \( M \) does not accept \( w \) iff \( \mathcal{L}(M_1) = \mathcal{L}(M_2) \)

- What happens if we pick \( M_1 \) to be \( \text{Amp}(M, w) \)?
  - If \( M \) accepts \( w \), then \( \mathcal{L}(M_1) = \Sigma^* \).
  - If \( M \) does not accept \( w \), then \( \mathcal{L}(M_1) = \emptyset \).

- Choose \( M_1 \) to be the amplifier machine and \( M_2 \) to be any TM with language \( \emptyset \). Then the above statement is true!
What's Going On?

- Suppose we have an oracle for $\text{EQ}_{\text{TM}}$.
- We want to know whether $M$ accepts $w$.
- To do this:
  - Find a TM $E$ we know has language $\emptyset$.
  - Ask the oracle “does TM $\text{Amp}(M, w)$ have the same language as TM $E$?”
  - If so, then $M$ does not accept $w$.
  - If not, then $M$ accepts $w$. 
**Theorem:** $\text{EQ}_{\text{TM}} \notin \text{RE}$.

**Proof:** We will prove $\overline{A}_{\text{TM}} \leq_{M} \text{EQ}_{\text{TM}}$. Since $\overline{A}_{\text{TM}} \notin \text{RE}$, this proves that $\text{EQ}_{\text{TM}} \notin \text{RE}$. To show $\overline{A}_{\text{TM}} \leq_{M} \text{EQ}_{\text{TM}}$, we will exhibit a mapping reduction from $\overline{A}_{\text{TM}}$ to $\text{EQ}_{\text{TM}}$.

For any TM/string pair $\langle M, w \rangle$, define $f(\langle M, w \rangle)$ to be the pair of TMs $\langle \text{Amp}(M, w), E \rangle$, where $E$ is the TM “On input $x$, reject $x$.” This function is computable, and note that $\mathcal{L}(E) = \emptyset$.

We claim that $\langle M, w \rangle \in \overline{A}_{\text{TM}}$ iff $\langle \text{Amp}(M, w), E \rangle \in \text{EQ}_{\text{TM}}$. To see this, note by definition of $\overline{A}_{\text{TM}}$ that $\langle M, w \rangle \in \overline{A}_{\text{TM}}$ iff $M$ does not accept $w$. By our theorem, $M$ does not accept $w$ iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$. Since $\mathcal{L}(E) = \emptyset$, we see $M$ does not accept $w$ iff $\mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(E)$. Finally, by definition of $\text{EQ}_{\text{TM}}$, $\mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(E)$ iff $\langle \text{Amp}(M, w), E \rangle \in \text{EQ}_{\text{TM}}$.

Collectively, we see $\langle M, w \rangle \in \overline{A}_{\text{TM}}$ iff $\langle \text{Amp}(M, w), E \rangle \in \text{EQ}_{\text{TM}}$. Thus $f$ is a mapping reduction from $\overline{A}_{\text{TM}}$ to $\text{EQ}_{\text{TM}}$, so $\overline{A}_{\text{TM}} \leq_{M} \text{EQ}_{\text{TM}}$, as required. ■
The Limits of Computability