Decidability and Undecidability
Major Ideas from Last Time

- Every TM can be converted into a string representation of itself.
  - The **encoding** of $M$ is denoted $\langle M \rangle$.
- The **universal Turing machine** $U^{TM}$ accepts an encoding $\langle M, w \rangle$ of a TM $M$ and string $w$, then simulates the execution of $M$ on $w$.
- The language of $U^{TM}$ is the language $A^{TM}$:
  \[ A^{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w. \} \]
- Equivalently:
  \[ A^{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \in \mathcal{L}(M) \} \]
Major Ideas from Last Time

- The universal Turing machine \( U_{\text{TM}} \) can be used as a subroutine in other Turing machines.

\[ H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a Turing machine:} \]
\[ \begin{align*}
&\text{· Run } M \text{ on } \varepsilon. \\
&\text{· If } M \text{ accepts } \varepsilon, \text{ then } H \text{ accepts } \langle M \rangle. \\
&\text{· If } M \text{ rejects } \varepsilon, \text{ then } H \text{ rejects } \langle M \rangle. 
\end{align*} \]

\[ H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a Turing machine:} \]
\[ \begin{align*}
&\text{· Nondeterministically guess a string } w. \\
&\text{· Run } M \text{ on } w. \\
&\text{· If } M \text{ accepts } w, \text{ then } H \text{ accepts } \langle M \rangle. \\
&\text{· If } M \text{ rejects } w, \text{ then } H \text{ rejects } \langle M \rangle. 
\end{align*} \]
Major Ideas from Last Time

- The **diagonalization language**, which we denote $L_D$, is defined as

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

- That is, $L_D$ is the set of descriptions of Turing machines that do not accept themselves.

- **Theorem:** $L_D \notin \text{RE}$
Outline for Today

- **More non-RE Languages**
  - We now know $L_D \notin \text{RE}$. Can we use this to find other non-RE languages?

- **Decidability and Class R**
  - How do we formalize the idea of an algorithm?

- **Undecidable Problems**
  - What problems admit no algorithmic solution?
Additional Unsolvable Problems
Finding Unsolvable Problems

- We can use the fact that $L_D \notin \text{RE}$ to show that other languages are also not \text{RE}.

- General proof approach: to show that some language $L$ is not \text{RE}, we will do the following:
  - Assume for the sake of contradiction that $L \in \text{RE}$, meaning that there is some TM $M$ for it.
  - Show that we can build a TM that uses $M$ as a subroutine in order to recognize $L_D$.
  - Reach a contradiction, since no TM recognizes $L_D$.
  - Conclude, therefore, that $L \notin \text{RE}$.
The Complement of $A_{TM}$

- Recall: the language $A_{TM}$ is the language of the universal Turing machine $U_{TM}$:

  $$A_{TM} = \mathcal{L}(U_{TM}) = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$$

- The complement of $A_{TM}$ (denoted $\overline{A}_{TM}$) is the language of all strings not contained in $A_{TM}$.

- Questions:
  - What language is this?
  - Is this language $\text{RE}$?
The language $A_{TM}$ is defined as:

$$\{\langle M, w \rangle \mid M \text{ is a TM that accepts } w\}$$

Equivalently:

$$\{x \mid x = \langle M, w \rangle \text{ for some TM } M \text{ and string } w, \text{ and } M \text{ accepts } w\}$$

Thus $\overline{A}_{TM}$ is:

$$\{x \mid x \neq \langle M, w \rangle \text{ for any TM } M \text{ and string } w, \text{ or } M \text{ is a TM that does not accept } w\}$$
The language $A_{TM}$ is defined as

$$\{⟨M, w⟩ | M \text{ is a TM that accepts } w\}$$

Equivalently:

$$\{x | x = ⟨M, w⟩ \text{ for some TM } M \text{ and string } w, \text{ and } M \text{ accepts } w\}$$

Thus $A_{TM}$ is

$$\{x | x \neq ⟨M, w⟩ \text{ for any TM } M \text{ and string } w, \text{ or } M \text{ is a TM that does not accept } w\}$$
Cheating With Math

• As a mathematical simplification, we will assume the following:

  Every string can be decoded into any collection of objects.

• Every string is an encoding of some TM $M$.
• Every string is an encoding of some TM $M$ and string $w$.

• Can do this as follows:
  • If the string is a legal encoding, go with that encoding.
  • Otherwise, pretend the string decodes to some predetermined group of objects.
Cheating With Math

- Example: Every string will be a valid C++ program.
- If it's already a C++ program, just compile it.
- Otherwise, pretend it's this program:
  ```
  int main() {
      return 0;
  }
  ```
$A_{TM}$ and $\overline{A}_{TM}$

- The language $A_{TM}$ is defined as
  \[ \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \} \]
- Thus $\overline{A}_{TM}$ is the language
  \[ \{ \langle M, w \rangle \mid M \text{ is a TM that doesn't accept } w \} \]
\[ \overline{A}_{TM} \not\in \text{RE} \]

- Although the language \( A_{TM} \in \text{RE} \) (since it's the language of \( U_{TM} \)), its complement \( \overline{A}_{TM} \not\in \text{RE} \).

- We will prove this as follows:
  
  - Assume, for contradiction, that \( \overline{A}_{TM} \in \text{RE} \).
  
  - This means there is a TM \( R \) for \( \overline{A}_{TM} \).
  
  - Using \( R \) as a subroutine, we will build a TM \( H \) that will recognize \( L_D \).
  
  - This is impossible, since \( L_D \not\in \text{RE} \).
  
  - Conclude, therefore, that \( \overline{A}_{TM} \not\in \text{RE} \).
Comparing $L_D$ and $\overline{A}_{TM}$

- The languages $L_D$ and $\overline{A}_{TM}$ are closely related:
  - $L_D$: Does $M$ not accept $\langle M \rangle$?
  - $\overline{A}_{TM}$: Does $M$ not accept string $w$?

- Given this connection, we will show how to turn a hypothetical recognizer for $\overline{A}_{TM}$ into a hypothetical recognizer for $L_D$. 
\( \langle M \rangle \xrightarrow{w} \text{Recognizer for } \overline{A_{TM}} \xrightarrow{} \text{Yes} \)

\( \text{Recognizer for } \overline{A_{TM}} \xrightarrow{} \text{No} \)
\begin{itemize}
\item \(\langle M \rangle\)
\item \(w\)
\end{itemize}

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

\begin{itemize}
\item Machine \(R\)
\item Yes
\item No
\end{itemize}
Recognizer for $\overline{A_{TM}}$

Machine $R$

Machine $H$

$\langle M \rangle$

$w$

Yes

No
$H = "\text{On input } \langle M \rangle:\n\begin{itemize}
\item \text{Construct the string } \langle M, \langle M \rangle \rangle.
\item \text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.
\item \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } \langle M \rangle.
\item \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } \langle M \rangle."
\end{itemize}
$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."
$H = \text{"On input } \langle M \rangle: \text{"
- Construct the string } \langle M, \langle M \rangle \rangle.
- Run R on } \langle M, \langle M \rangle \rangle.
- \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } \langle M \rangle.
- \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } \langle M \rangle."
$H = \text{"On input } \langle M \rangle:\$

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$.

Machine $R$ accepts $\langle M, \langle M \rangle \rangle$
$H = \text{"On input } \langle M \rangle:\n\quad \cdot \text{Construct the string } \langle M, \langle M \rangle \rangle. \\
\quad \cdot \text{Run } R \text{ on } \langle M, \langle M \rangle \rangle. \\
\quad \cdot \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ accepts } \langle M \rangle. \\
\quad \cdot \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ then } H \text{ rejects } \langle M \rangle.\text{"}
$\langle M \rangle$  

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

**Machine $R$**

**Yes**

**No**

**Machine $H$**

$H = "On input $\langle M \rangle$:  
- Construct the string $\langle M, \langle M \rangle \rangle$.  
- Run $R$ on $\langle M, \langle M \rangle \rangle$.  
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.  
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle."

What happens if...  
$M$ does not accept $\langle M \rangle$?  

**Accept**

Machine $R$ accepts $\langle M, \langle M \rangle \rangle$
$H =$ “On input $\langle M \rangle$:
\begin{itemize}
  \item Construct the string $\langle M, \langle M \rangle \rangle$.
  \item Run $R$ on $\langle M, \langle M \rangle \rangle$.
  \item If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
  \item If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$.
\end{itemize}
$H = \text{"On input } \langle M \rangle:\n$ 
- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$."

What happens if...

$M$ does not accept $\langle M \rangle$?

Accept

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

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
  - If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
  - If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$.

What happens if...

$M$ does not accept $\langle M \rangle$?

Accept

$M$ accepts $\langle M \rangle$?

Machine $R$ does not accept $\langle M, \langle M \rangle \rangle$
$H = "On\ input \langle M \rangle:\$

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$.

**What happens if...**

- $M$ does not accept $\langle M \rangle$?
  - **Accept**
- $M$ accepts $\langle M \rangle$?
  - **Reject or Loop**

Machine $R$ does not accept $\langle M, \langle M \rangle \rangle$.
$H = \text{“On input } \langle M \rangle: \text{”}$

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$."

What happens if...

$M$ does not accept $\langle M \rangle$?

Accept

$M$ accepts $\langle M \rangle$?

Reject or Loop
$H$ = “On input $⟨M⟩$:  
  • Construct the string $⟨M, ⟨M⟩⟩$.  
  • Run $R$ on $⟨M, ⟨M⟩⟩$.  
  • If $R$ accepts $⟨M, ⟨M⟩⟩$, then $H$ accepts $⟨M⟩$.  
  • If $R$ rejects $⟨M, ⟨M⟩⟩$, then $H$ rejects $⟨M⟩$.”

$H$ is a TM for $L_D$!
Theorem: \( \overline{A_{TM}} \not\in \text{RE}. \)
Theorem: $\overline{A}_{TM} \notin \text{RE}$.

Proof:
Theorem: \( \overline{A_{TM}} \notin \text{RE} \).

Proof: By contradiction; assume that \( \overline{A_{TM}} \in \text{RE} \).
Theorem: $\overline{A_{TM}} \notin \mathbf{RE}$.

Proof: By contradiction; assume that $\overline{A_{TM}} \in \mathbf{RE}$. Then there must be a recognizer for $\overline{A_{TM}}$; call it $R$.
Theorem: $\overline{A_{TM}} \not\in \text{RE}$.  

Proof: By contradiction; assume that $\overline{A_{TM}} \in \text{RE}$. Then there must be a recognizer for $\overline{A_{TM}}$; call it $R$.

Consider the TM $H$ defined below:

$H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \newline \text{ Construct the string } \langle M, \langle M \rangle \rangle. \newline \text{ Run } R \text{ on } \langle M, \langle M \rangle \rangle. \newline \text{ If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle. \newline \text{ If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."}$
**Theorem:** \( \overline{A_{TM}} \not\in \text{RE} \).

**Proof:** By contradiction; assume that \( \overline{A_{TM}} \in \text{RE} \). Then there must be a recognizer for \( \overline{A_{TM}} \); call it \( R \).

Consider the TM \( H \) defined below:

\[
H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \\
\text{Construct the string } \langle M, \langle M \rangle \rangle. \\
\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle. \\
\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle. \\
\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."
\]

We claim that \( \mathcal{L}(H) = L_D \).
**Theorem:** $\overline{A_{TM}} \notin \text{RE}$.

**Proof:** By contradiction; assume that $\overline{A_{TM}} \in \text{RE}$. Then there must be a recognizer for $\overline{A_{TM}}$; call it $R$.

Consider the TM $H$ defined below:

$$H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}
\text{Construct the string } \langle M, \langle M \rangle \rangle.
\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.
\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ } H \text{ accepts } \langle M \rangle.
\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ } H \text{ rejects } \langle M \rangle."$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$. 

Theorem: $\overline{A_{TM}} \notin \text{RE}$.

Proof: By contradiction; assume that $\overline{A_{TM}} \in \text{RE}$. Then there must be a recognizer for $\overline{A_{TM}}$; call it $R$.

Consider the TM $H$ defined below:

\[
H = "\text{On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \\
\quad \text{Construct the string } \langle M, \langle M \rangle \rangle. \\
\quad \text{Run } R \text{ on } \langle M, \langle M \rangle \rangle. \\
\quad \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle. \\
\quad \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."
\]

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. 
Theorem: $\overline{A}_{TM} \notin RE$.

Proof: By contradiction; assume that $\overline{A}_{TM} \in RE$. Then there must be a recognizer for $\overline{A}_{TM}$; call it $R$.

Consider the TM $H$ defined below:

$$H = "On \text{ input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

Construct the string $\langle M, \langle M \rangle \rangle$.

Run $R$ on $\langle M, \langle M \rangle \rangle$.

If $R$ accepts $\langle M, \langle M \rangle \rangle$, $H$ accepts $\langle M \rangle$.

If $R$ rejects $\langle M, \langle M \rangle \rangle$, $H$ rejects $\langle M \rangle$.”

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$.

Since $R$ is a recognizer for $\overline{A}_{TM}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. But this is impossible, since $L_D \notin RE$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $\overline{A}_{TM} \notin RE$, as required. ■
Theorem: $\overline{A_{TM}} \notin \text{RE}$.  

Proof: By contradiction; assume that $\overline{A_{TM}} \in \text{RE}$. Then there must be a recognizer for $\overline{A_{TM}}$; call it $R$.

Consider the TM $H$ defined below:

$$H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}\n\text{Construct the string } \langle M, \langle M \rangle \rangle.\n\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.\n\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle.\n\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A_{TM}}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. 

We have reached a contradiction, so our assumption must have been incorrect. Thus $\overline{A_{TM}} \notin \text{RE}$, as required.■
Theorem: $\overline{A}_{TM} \notin \text{RE}$.

Proof: By contradiction; assume that $\overline{A}_{TM} \in \text{RE}$. Then there must be a recognizer for $\overline{A}_{TM}$; call it $R$.

Consider the TM $H$ defined below:

$$H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run $R$ on $\langle M, \langle M \rangle \rangle$.
- If $R$ accepts $\langle M, \langle M \rangle \rangle$, $H$ accepts $\langle M \rangle$.
- If $R$ rejects $\langle M, \langle M \rangle \rangle$, $H$ rejects $\langle M \rangle$.”

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A}_{TM}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. Therefore, we have reached a contradiction, so our assumption must have been incorrect. Thus $\overline{A}_{TM} \notin \text{RE}$, as required. ■
Theorem: $\overline{A_{\text{TM}}} \notin \text{RE}$.

Proof: By contradiction; assume that $\overline{A_{\text{TM}}} \in \text{RE}$. Then there must be a recognizer for $\overline{A_{\text{TM}}}$; call it $R$.

Consider the TM $H$ defined below:

$$H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}\$$
$$\text{Construct the string } \langle M, \langle M \rangle \rangle.\$$
$$\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.\$$
$$\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle.\$$
$$\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A_{\text{TM}}}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. But this is impossible, since $L_D \notin \text{RE}$. 
Theorem: $\overline{A}_{TM} \notin \mathbf{RE}$.

Proof: By contradiction; assume that $\overline{A}_{TM} \in \mathbf{RE}$. Then there must be a recognizer for $\overline{A}_{TM}$; call it $R$.

Consider the TM $H$ defined below:

$$H = "\text{On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

$$\text{Construct the string } \langle M, \langle M \rangle \rangle.$$  

$$\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.$$  

$$\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle.$$  

$$\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A}_{TM}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. But this is impossible, since $L_D \notin \mathbf{RE}$.

We have reached a contradiction, so our assumption must have been incorrect.
Theorem: $\overline{A_{TM}} \notin \text{RE}$.

Proof: By contradiction; assume that $\overline{A_{TM}} \in \text{RE}$. Then there must be a recognizer for $\overline{A_{TM}}$; call it $R$.

Consider the TM $H$ defined below:

$$H = \text{"On input } \langle M \rangle \text{, where } M \text{ is a TM:}
\quad \text{Construct the string } \langle M, \langle M \rangle \rangle.
\quad \text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.
\quad \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \ H \text{ accepts } \langle M \rangle.
\quad \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \ H \text{ rejects } \langle M \rangle."
$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A_{TM}}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. But this is impossible, since $L_D \notin \text{RE}$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $\overline{A_{TM}} \notin \text{RE}$, as required.
**Theorem:** $\overline{A}_{TM} \notin \text{RE}$.

**Proof:** By contradiction; assume that $\overline{A}_{TM} \in \text{RE}$. Then there must be a recognizer for $\overline{A}_{TM}$; call it $R$.

Consider the TM $H$ defined below:

$$H = "\text{On input } \langle M \rangle, \text{ where } M \text{ is a TM:}\n\text{Construct the string } \langle M, \langle M \rangle \rangle.\n\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.\n\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, H \text{ accepts } \langle M \rangle.\n\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, H \text{ rejects } \langle M \rangle."$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A}_{TM}, R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$.

Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. But this is impossible, since $L_D \notin \text{RE}$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $\overline{A}_{TM} \notin \text{RE}$, as required. $\blacksquare$
Why All This Matters

• We finally have found concrete examples of unsolvable problems!

• We are starting to see a line of reasoning we can use to find unsolvable problems:
  • Start with a known unsolvable problem.
  • Try to show that the unsolvability of that problem entails the unsolvability of other problems.

• We will see this used extensively in the upcoming weeks.
Revisiting RE
Recall: Language of a TM

- The language of a Turing machine $M$, denoted $\mathcal{L}(M)$, is the set of all strings that $M$ accepts:

$$\mathcal{L}(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

- For any $w \in \mathcal{L}(M)$, $M$ accepts $w$.
- For any $w \notin \mathcal{L}(M)$, $M$ does not accept $w$.
  - It might loop forever, or it might explicitly reject.
- A language is called **recognizable** if it is the language of some TM.
- Notation: $\text{RE}$ is the set of all recognizable languages.

$$L \in \text{RE} \iff L \text{ is recognizable}$$
Why “Recognizable?”

- Given TM $M$ with language $\mathcal{L}(M)$, running $M$ on a string $w$ will not necessarily tell you whether $w \in \mathcal{L}(M)$.

- If the machine is running, you can't tell whether
  - It is eventually going to halt, but just needs more time, or
  - It is never going to halt.

- However, if you know for a fact that $w \in \mathcal{L}(M)$, then the machine can confirm this (it eventually accepts).

- The machine can't decide whether or not $w \in \mathcal{L}(M)$, but it can recognize strings that are in the language.

- We sometimes call a TM for a language $L$ a recognizer for $L$. 
Deciders

- Some Turing machines always halt; they never go into an infinite loop.
- Turing machines of this sort are called **deciders**.
- For deciders, accepting is the same as not rejecting and rejecting is the same as not accepting.
Decidable Languages

- A language $L$ is called **decidable** iff there is a decider $M$ such that $\mathcal{L}(M) = L$.
- Given a decider $M$, you *can* learn whether or not a string $w \in \mathcal{L}(M)$.
  - Run $M$ on $w$.
  - Although it might take a staggeringly long time, $M$ will eventually accept or reject $w$.
- The set $\mathbb{R}$ is the set of all decidable languages.
  
  \[ L \in \mathbb{R} \iff L \text{ is decidable} \]
R and RE Languages

• Intuitively, a language is in RE if there is some way that you could exhaustively search for a proof that $w \in L$.
  • If you find it, accept!
  • If you don't find one, keep looking!
• Intuitively, a language is in R if there is a concrete algorithm that can determine whether $w \in L$.
  • It tends to be much harder to show that a language is in R than in RE.
Examples of \( \mathbf{R} \) Languages

- All regular languages are in \( \mathbf{R} \).
  - If \( L \) is regular, we can run the DFA for \( L \) on a string \( w \) and then either accept or reject \( w \) based on what state it ends in.

- \( \{ 0^n1^n \mid n \in \mathbb{N} \} \) is in \( \mathbf{R} \).
  - The TM we built last Wednesday is a decider.

- Multiplication is in \( \mathbf{R} \).
  - Can check if \( m \times n = p \) by repeatedly subtracting out copies of \( n \). If the equation balances, accept; if not, reject.
CFLs and $\mathbb{R}$

• Using an NTM, we sketched a proof that all CFLs are in $\mathbb{RE}$.
  • Nondeterministically guess a derivation, then deterministically check that derivation.

• Harder result: all CFLs are in $\mathbb{R}$.
  • Read Sipser, Ch. 4.1 for details.
  • Or come talk to me after lecture!
Why $\mathbf{R}$ Matters

- If a language is in $\mathbf{R}$, there is an algorithm that can decide membership in that language.
  - Run the decider and see what it says.
- If there is an algorithm that can decide membership in a language, that language is in $\mathbf{R}$.
  - By the Church-Turing thesis, any effective model of computation is equivalent in power to a Turing machine.
  - Thus if there is any algorithm for deciding membership in the language, there must be a decider for it.
  - Thus the language is in $\mathbf{R}$.
- A language is in $\mathbf{R}$ iff there is an algorithm for deciding membership in that language.
R \supseteq RE

- Every decider is a Turing machine, but not every Turing machine is a decider.
- Thus R \subseteq RE.
- Hugely important theoretical question:

  Is \( R = RE \)?

- That is, if we can verify that a string is in a language, can we decide whether that string is in the language?
An Important Observation
**R is Closed Under Complementation**

If $L \in R$, then $\overline{L} \in R$ as well.

**Decider for $L$**

$M' = "On input $w$: Run $M$ on $w$. If $M$ accepts $w$, reject. If $M$ rejects $w$, accept."$

Will this work if $M$ is a recognizer, rather than a decider?
*Theorem:* \( \mathbf{R} \) is closed under complementation.

Proof: Consider any \( L \in \mathbf{R} \). We will prove that \( L \in \mathbf{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = L \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[ M' = \text{"On input } w \in \Sigma^*:\text{ Run } M \text{ on } w. \text{ If } M \text{ accepts } w, \text{ reject. If } M \text{ rejects } w, \text{ accept."} \]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = L \).

To show that \( M' \) is a decider, we will prove that it always halts.

Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( \mathcal{L}(M') = L \), we will prove that \( M' \) accepts \( w \) iff \( w \in L \).

Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \). Since \( M \) is a decider, \( M \) rejects \( w \) iff \( M \) does not accept \( w \). \( M \) does not accept \( w \) iff \( w \not\in \mathcal{L}(M) \). Thus \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( w \not\in \mathcal{L}(M) \), so \( M' \) accepts \( w \) iff \( w \in L \). Therefore, \( \mathcal{L}(M') = L \).

Since \( M' \) is a decider with \( \mathcal{L}(M') = L \), we have \( L \in \mathbf{R} \), as required. \( \square \)
Theorem: $R$ is closed under complementation.

Proof: Consider any $L \in R$. 

Let $M$ be a decider for $L$. Then construct the machine $M'$ as follows:

$M' = \text{On input } w \in \Sigma^*:
\quad \text{Run } M \text{ on } w.
\quad \text{If } M \text{ accepts } w, \text{ reject.}
\quad \text{If } M \text{ rejects } w, \text{ accept.}$

To show that $M'$ is a decider, we will prove that it always halts. Consider what happens if we run $M'$ on any input $w$. First, $M'$ runs $M$ on $w$. Since $M$ is a decider, $M$ either accepts $w$ or rejects $w$. If $M$ accepts $w$, $M'$ rejects $w$. If $M$ rejects $w$, $M'$ accepts $w$. Thus $M'$ always accepts or rejects, so $M'$ is a decider.

To show that $(\mathcal{L}M') = L$, we will prove that $M'$ accepts $w$ iff $w \in L$. Note that $M'$ accepts $w$ iff $w \in \Sigma^*$ and $M$ rejects $w$. Since $M$ is a decider, $M$ rejects $w$ iff $M$ does not accept $w$. $M$ does not accept $w$ iff $w \notin (\mathcal{L}M)$. Thus $M'$ accepts $w$ iff $w \in \Sigma^*$ and $w \notin (\mathcal{L}M)$, so $M'$ accepts $w$ iff $w \in L$. Therefore, $(\mathcal{L}M') = L$.

Since $M'$ is a decider with $(\mathcal{L}M') = L$, we have $L \in R$, as required. ■
Theorem: $R$ is closed under complementation.
Proof: Consider any $L \in R$. We will prove that $\overline{L} \in R$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$. 

$M'$ will be constructed as follows:

- **On input $w \in \Sigma^*$:**
  - Run $M$ on $w$.
  - If $M$ accepts $w$, reject.
  - If $M$ rejects $w$, accept.

We need to show that $M'$ is a decider and that $\mathcal{L}(M') = \overline{L}$.

To show that $M'$ is a decider, we will prove that it always halts.

Consider what happens if we run $M'$ on any input $w$.

First, $M'$ runs $M$ on $w$. Since $M$ is a decider, $M$ either accepts $w$ or rejects $w$.

- If $M$ accepts $w$, $M'$ rejects $w$.
- If $M$ rejects $w$, $M'$ accepts $w$.

Thus $M'$ always accepts or rejects, so $M'$ is a decider.

To show that $\mathcal{L}(M') = \overline{L}$, we will prove that $M'$ accepts $w$ iff $w \notin \mathcal{L}(M')$.

Note that $M'$ accepts $w$ iff $w \in \Sigma^*$ and $M$ rejects $w$. Since $M$ is a decider, $M$ rejects $w$ iff $M$ does not accept $w$.

$M$ does not accept $w$ iff $w \notin \mathcal{L}(M')$. Thus $M'$ accepts $w$ iff $w \in \Sigma^*$ and $w \notin \mathcal{L}(M')$, so $M'$ accepts $w$ iff $w \notin \overline{L}$. Therefore, $\mathcal{L}(M') = \overline{L}$.

Since $M'$ is a decider with $\mathcal{L}(M') = \overline{L}$, we have $L \in R$, as required. ■
Theorem: $R$ is closed under complementation.

Proof: Consider any $L \in R$. We will prove that $\overline{L} \in R$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$.

This is the standard way to show that a language is in $R$. Note that we aren’t just building any arbitrary TM; it has to be a decider.
Theorem: \( R \) is closed under complementation.

Proof: Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( L(M') = \overline{L} \).

Let \( M \) be a decider for \( L \).
Theorem: \( \mathbf{R} \) is closed under complementation.

Proof: Consider any \( L \in \mathbf{R} \). We will prove that \( \overline{L} \in \mathbf{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\
\begin{align*}
\text{Run } M \text{ on } w.
\text{If } M \text{ accepts } w, \text{ reject.}
\text{If } M \text{ rejects } w, \text{ accept."
}\end{align*}
\]
Theorem: $R$ is closed under complementation.

Proof: Consider any $L \in R$. We will prove that $\overline{L} \in R$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$.

Let $M$ be a decider for $L$. Then construct the machine $M'$ as follows:

$$M' = \text{"On input } w \in \Sigma^*:$$

- Run $M$ on $w$.
- If $M$ accepts $w$, reject.
- If $M$ rejects $w$, accept.$$

We need to show that $M'$ is a decider and that $\mathcal{L}(M') = \overline{L}$. 

Theorem: $\mathbf{R}$ is closed under complementation.

Proof: Consider any $L \in \mathbf{R}$. We will prove that $\overline{L} \in \mathbf{R}$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$.

Let $M$ be a decider for $L$. Then construct the machine $M'$ as follows:

$$M' = \text{"On input } w \in \Sigma^*:\n\text{Run } M \text{ on } w.\n\text{If } M \text{ accepts } w, \text{ reject.}\n\text{If } M \text{ rejects } w, \text{ accept."}$$

We need to show that $M'$ is a decider and that $\mathcal{L}(M') = \overline{L}$.

There are two proofs required here, and they're separate from one another. Just showing one or the other isn't sufficient.
Theorem: \( \mathcal{R} \) is closed under complementation.

Proof: Consider any \( L \in \mathcal{R} \). We will prove that \( \overline{L} \in \mathcal{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\n\begin{align*}
\text{Run } M \text{ on } w. \\
\text{If } M \text{ accepts } w, \text{ reject.} \\
\text{If } M \text{ rejects } w, \text{ accept.}\n\end{align*}
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts.
**Theorem:** \( \mathcal{R} \) is closed under complementation.

**Proof:** Consider any \( L \in \mathcal{R} \). We will prove that \( \overline{L} \in \mathcal{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{“On input } w \in \Sigma^*:\n\begin{align*}
\text{Run } M \text{ on } w. \\
\text{If } M \text{ accepts } w, \text{ reject.} \\
\text{If } M \text{ rejects } w, \text{ accept.} 
\end{align*}
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \).
Theorem: \( R \) is closed under complementation.

Proof: Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*: \\
\text{Run } M \text{ on } w. \\
\text{If } M \text{ accepts } w, \text{ reject.} \\
\text{If } M \text{ rejects } w, \text{ accept."
}\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \).
Theorem: $R$ is closed under complementation.

Proof: Consider any $L \in R$. We will prove that $\bar{L} \in R$ by constructing a decider $M'$ such that $L(M') = \bar{L}$.

Let $M$ be a decider for $L$. Then construct the machine $M'$ as follows:

$$M' = \text{"On input } w \in \Sigma^*: \text{ Run } M \text{ on } w. \text{ If } M \text{ accepts } w, \text{ reject. If } M \text{ rejects } w, \text{ accept."

We need to show that $M'$ is a decider and that $L(M') = \bar{L}$.

To show that $M'$ is a decider, we will prove that it always halts. Consider what happens if we run $M'$ on any input $w$. First, $M'$ runs $M$ on $w$. Since $M$ is a decider, $M$ either accepts $w$ or rejects $w$. 


Theorem: \( \mathbf{R} \) is closed under complementation.

Proof: Consider any \( L \in \mathbf{R} \). We will prove that \( \overline{L} \in \mathbf{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\
\quad \text{Run } M \text{ on } w.
\quad \text{If } M \text{ accepts } w, \text{ reject.}
\quad \text{If } M \text{ rejects } w, \text{ accept."}
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \).
Theorem: \( \mathbb{R} \) is closed under complementation.

Proof: Consider any \( L \in \mathbb{R} \). We will prove that \( \overline{L} \in \mathbb{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\ \\
\quad \text{Run } M \text{ on } w. \\
\quad \text{If } M \text{ accepts } w, \text{ reject.} \\
\quad \text{If } M \text{ rejects } w, \text{ accept."}
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \).
Theorem: \( R \) is closed under complementation.

Proof: Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[ M' = \text{On input } w \in \Sigma^*: \]
  
  Run \( M \) on \( w \).

  If \( M \) accepts \( w \), reject.

  If \( M \) rejects \( w \), accept.

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.
**Theorem:** $\mathbf{R}$ is closed under complementation.

**Proof:** Consider any $L \in \mathbf{R}$. We will prove that $\overline{L} \in \mathbf{R}$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$.

Let $M$ be a decider for $L$. Then construct the machine $M'$ as follows:

$$M' = \text{"On input } w \in \Sigma^*: \text{ Run } M \text{ on } w. \text{ If } M \text{ accepts } w, \text{ reject. If } M \text{ rejects } w, \text{ accept."}$$

We need to show that $M'$ is a decider and that $\mathcal{L}(M') = \overline{L}$.

To show that $M'$ is a decider, we will prove that it always halts. Consider what happens if we run $M'$ on any input $w$. First, $M'$ runs $M$ on $w$. Since $M$ is a decider, $M$ either accepts $w$ or rejects $w$. If $M$ accepts $w$, $M'$ rejects $w$. If $M$ rejects $w$, $M'$ accepts $w$. Thus $M'$ always accepts or rejects, so $M'$ is a decider.

To show that $\mathcal{L}(M') = \overline{L}$, we will prove that $M'$ accepts $w$ iff $w \in \overline{L}$.
**Theorem:** \( \mathbf{R} \) is closed under complementation.

**Proof:** Consider any \( L \in \mathbf{R} \). We will prove that \( \overline{L} \in \mathbf{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*: \\
\text{Run } M \text{ on } w. \\
\text{If } M \text{ accepts } w, \text{ reject.} \\
\text{If } M \text{ rejects } w, \text{ accept.}" 
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( \mathcal{L}(M') = \overline{L} \), we will prove that \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \).
**Theorem:** \( \mathbf{R} \) is closed under complementation.

**Proof:** Consider any \( L \in \mathbf{R} \). We will prove that \( \overline{L} \in \mathbf{R} \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*: \\
\text{Run } M \text{ on } w. \\
\text{If } M \text{ accepts } w, \text{ reject.} \\
\text{If } M \text{ rejects } w, \text{ accept."}
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( \mathcal{L}(M') = \overline{L} \), we will prove that \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \). Since \( M \) is a decider, \( M \) rejects \( w \) iff \( M \) does not accept \( w \).
Theorem: \( R \) is closed under complementation.

**Proof:** Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\text{ Run } M \text{ on } w. \text{ If } M \text{ accepts } w, \text{ reject. If } M \text{ rejects } w, \text{ accept."}
\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( \mathcal{L}(M') = \overline{L} \), we will prove that \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \). Since \( M \) is a decider, \( M \) rejects \( w \) iff \( M \) does not accept \( w \). \( M \) does not accept \( w \) iff \( w \not\in \mathcal{L}(M) \). 

Theorem: $\mathbf{R}$ is closed under complementation.

Proof: Consider any $L \in \mathbf{R}$. We will prove that $\overline{L} \in \mathbf{R}$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$.

Let $M$ be a decider for $L$. Then construct the machine $M'$ as follows:

$$M' = \text{"On input } w \in \Sigma^*:\n\text{Run } M \text{ on } w.\n\text{If } M \text{ accepts } w, \text{ reject.}\n\text{If } M \text{ rejects } w, \text{ accept."}$$

We need to show that $M'$ is a decider and that $\mathcal{L}(M') = \overline{L}$.

To show that $M'$ is a decider, we will prove that it always halts. Consider what happens if we run $M'$ on any input $w$. First, $M'$ runs $M$ on $w$. Since $M$ is a decider, $M$ either accepts $w$ or rejects $w$. If $M$ accepts $w$, $M'$ rejects $w$. If $M$ rejects $w$, $M'$ accepts $w$. Thus $M'$ always accepts or rejects, so $M'$ is a decider.

To show that $\mathcal{L}(M') = \overline{L}$, we will prove that $M'$ accepts $w$ iff $w \in \overline{L}$. Note that $M'$ accepts $w$ iff $w \in \Sigma^*$ and $M$ rejects $w$. Since $M$ is a decider, $M$ rejects $w$ iff $M$ does not accept $w$. $M$ does not accept $w$ iff $w \notin \mathcal{L}(M)$. Thus $M'$ accepts $w$ iff $w \in \Sigma^*$ and $w \notin \mathcal{L}(M)$, so $M'$ accepts $w$ iff $w \in \overline{L}$. 

\[\square\]
Theorem: \( R \) is closed under complementation.

Proof: Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( L(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\n\begin{align*}
&\text{Run } M \text{ on } w. \\
&\text{If } M \text{ accepts } w, \text{ reject.} \\
&\text{If } M \text{ rejects } w, \text{ accept."
}\]

We need to show that \( M' \) is a decider and that \( L(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( L(M') = \overline{L} \), we will prove that \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \). Since \( M \) is a decider, \( M \) rejects \( w \) iff \( M \) does not accept \( w \). \( M \) does not accept \( w \) iff \( w \not\in L(M) \). Thus \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( w \not\in L(M) \), so \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Therefore, \( L(M') = \overline{L} \).
Theorem: \( R \) is closed under complementation.

Proof: Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\
    \begin{align*}
    \text{Run } M \text{ on } w. \\
    \text{If } M \text{ accepts } w, \text{ reject.} \\
    \text{If } M \text{ rejects } w, \text{ accept."
}\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( \mathcal{L}(M') = \overline{L} \), we will prove that \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \). Since \( M \) is a decider, \( M \) rejects \( w \) iff \( M \) does not accept \( w \). \( M \) does not accept \( w \) iff \( w \notin \mathcal{L}(M) \). Thus \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( w \notin \mathcal{L}(M) \), so \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Therefore, \( \mathcal{L}(M') = \overline{L} \).

Since \( M' \) is a decider with \( \mathcal{L}(M') = \overline{L} \), we have \( \overline{L} \in R \), as required.
Theorem: \( R \) is closed under complementation.

Proof: Consider any \( L \in R \). We will prove that \( \overline{L} \in R \) by constructing a decider \( M' \) such that \( \mathcal{L}(M') = \overline{L} \).

Let \( M \) be a decider for \( L \). Then construct the machine \( M' \) as follows:

\[
M' = \text{"On input } w \in \Sigma^*:\n\quad \text{Run } M \text{ on } w. \\
\quad \text{If } M \text{ accepts } w, \text{ reject.} \\
\quad \text{If } M \text{ rejects } w, \text{ accept."
}\]

We need to show that \( M' \) is a decider and that \( \mathcal{L}(M') = \overline{L} \).

To show that \( M' \) is a decider, we will prove that it always halts. Consider what happens if we run \( M' \) on any input \( w \). First, \( M' \) runs \( M \) on \( w \). Since \( M \) is a decider, \( M \) either accepts \( w \) or rejects \( w \). If \( M \) accepts \( w \), \( M' \) rejects \( w \). If \( M \) rejects \( w \), \( M' \) accepts \( w \). Thus \( M' \) always accepts or rejects, so \( M' \) is a decider.

To show that \( \mathcal{L}(M') = \overline{L} \), we will prove that \( M' \) accepts \( w \) iff \( w \in \overline{L} \).

Note that \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( M \) rejects \( w \). Since \( M \) is a decider, \( M \) rejects \( w \) iff \( M \) does not accept \( w \). \( M \) does not accept \( w \) iff \( w \notin \mathcal{L}(M) \). Thus \( M' \) accepts \( w \) iff \( w \in \Sigma^* \) and \( w \notin \mathcal{L}(M) \), so \( M' \) accepts \( w \) iff \( w \in \overline{L} \). Therefore, \( \mathcal{L}(M') = \overline{L} \).

Since \( M' \) is a decider with \( \mathcal{L}(M') = \overline{L} \), we have \( \overline{L} \in R \), as required. ■
\( R = \text{RE} \)

- We can now resolve the question of \( R = \text{RE} \).
- If \( R = \text{RE} \), we need to show that if there is a recognizer for any \( \text{RE} \) language \( L \), there has to be a decider for \( L \).
- If \( R \neq \text{RE} \), we just need to find a single language in \( \text{RE} \) that is not in \( R \).
$A_{TM}$

- Recall: the language $A_{TM}$ is the language of the universal Turing machine $U_{TM}$.
- Consequently, $A_{TM} \in \text{RE}$.
- Is $A_{TM} \in \text{R}$?
Theorem: $A_{TM} \notin R$. 
Theorem: $A_{TM} \notin R$.

Proof: By contradiction; assume $A_{TM} \in R$. Since $R$ is closed under complementation, this means that $A_{TM} \in R$. Since $R \subseteq RE$, this means that $A_{TM} \in RE$. But this is impossible, since we know $A_{TM} \notin RE$. We have reached a contradiction, so our assumption must have been incorrect. Thus $A_{TM} \notin R$, as required. ■
**Theorem:** \( A_{TM} \notin R. \)

**Proof:** By contradiction; assume \( A_{TM} \in R. \) Since \( R \) is closed under complementation, this means that \( \overline{A}_{TM} \in R. \) But this is impossible, since we know \( A_{TM} \notin RE. \) We have reached a contradiction, so our assumption must have been incorrect. Thus \( A_{TM} \notin R, \) as required. ■
**Theorem:** $A_{TM} \notin R$.

**Proof:** By contradiction; assume $A_{TM} \in R$. Since $R$ is closed under complementation, this means that $\overline{A_{TM}} \in R$. Since $R \subseteq RE$, this means that $\overline{A_{TM}} \in RE$. But this is impossible, since we know $A_{TM} \notin RE$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $A_{TM} \notin R$, as required. ■
Theorem: $A_{TM} \notin R$.

Proof: By contradiction; assume $A_{TM} \in R$. Since $R$ is closed under complementation, this means that $\overline{A}_{TM} \in R$. Since $R \subseteq RE$, this means that $\overline{A}_{TM} \in RE$. But this is impossible, since we know $\overline{A}_{TM} \notin RE$. 

Theorem: $A_{TM} \notin R$.

Proof: By contradiction; assume $A_{TM} \in R$. Since $R$ is closed under complementation, this means that $\overline{A}_{TM} \in R$. Since $R \subseteq RE$, this means that $\overline{A}_{TM} \in RE$. But this is impossible, since we know $\overline{A}_{TM} \notin RE$.

We have reached a contradiction, so our assumption must have been incorrect.
Theorem: $A_{TM} \notin R$.

Proof: By contradiction; assume $A_{TM} \in R$. Since $R$ is closed under complementation, this means that $\overline{A}_{TM} \in R$. Since $R \subseteq RE$, this means that $\overline{A}_{TM} \in RE$. But this is impossible, since we know $\overline{A}_{TM} \notin RE$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $A_{TM} \notin R$, as required.
**Theorem:** \( A_{TM} \notin \mathbb{R} \).

**Proof:** By contradiction; assume \( A_{TM} \in \mathbb{R} \). Since \( \mathbb{R} \) is closed under complementation, this means that \( \overline{A}_{TM} \in \mathbb{R} \). Since \( \mathbb{R} \subseteq \mathbb{RE} \), this means that \( \overline{A}_{TM} \in \mathbb{RE} \). But this is impossible, since we know \( \overline{A}_{TM} \notin \mathbb{RE} \).

We have reached a contradiction, so our assumption must have been incorrect. Thus \( A_{TM} \notin \mathbb{R} \), as required. \( \blacksquare \)
The Limits of Computability
What this Means

- The undecidability of $A_{TM}$ means that we cannot "cheat" with Turing machines.

- We cannot necessarily build a TM to do an exhaustive search over a space (i.e. a recognizer), then decide whether it accepts without running it.

- **Intuition:** In most cases, you cannot decide what a TM will do without running it to see what happens.

- In some cases, you can recognize when a TM has performed some task.

- In some cases, you can't do either. For example, you cannot always recognize that a TM will not accept a string.
What this Means

- **Major result:** $R \neq RE$.

- There are some problems where we can only give a “yes” answer when the answer is “yes” and cannot necessarily give a yes-or-no answer.

- Solving a problem is *fundamentally harder* than recognizing a correct answer.
Another Undecidable Problem
$L_D$ Revisited

- The diagonalization language $L_D$ is the language

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

- As we saw before, $L_D \notin \text{RE}$.

- But what about $\overline{L_D}$?
The language $L_D$ is the language

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

Therefore, $\overline{L}_D$ is the language

$\overline{L}_D = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \in \mathcal{L}(M)\}$

Two questions:

- What is this language?
- Is this language RE?
All Turing machines, listed in some order.
All descriptions of TMs, listed in the same order.
<table>
<thead>
<tr>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\langle M_5 \rangle$</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_0</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>M_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>...</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>-----</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>...</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
<td>…</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M₀</td>
<td>M₁</td>
<td>M₂</td>
<td>M₃</td>
<td>M₄</td>
<td>M₅</td>
<td>…</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---</td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>…</td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>$\langle M_0 \rangle$</td>
<td>$\langle M_1 \rangle$</td>
<td>$\langle M_2 \rangle$</td>
<td>$\langle M_3 \rangle$</td>
<td>$\langle M_4 \rangle$</td>
<td>$\langle M_5 \rangle$</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>\langle M_0 \rangle</td>
<td>\langle M_1 \rangle</td>
<td>\langle M_2 \rangle</td>
<td>\langle M_3 \rangle</td>
<td>\langle M_4 \rangle</td>
<td>\langle M_5 \rangle</td>
<td>...</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>\text{Acc}</td>
<td>No</td>
<td>No</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>...</td>
</tr>
<tr>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>\text{Acc}</td>
<td>No</td>
<td>No</td>
<td>\text{Acc}</td>
<td>\text{Acc}</td>
<td>...</td>
</tr>
<tr>
<td>\text{Acc}</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>\text{Acc}</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

\text{Acc} \quad \text{Acc} \quad \text{Acc} \quad \text{No} \quad \text{Acc} \quad \text{No} \quad \ldots
<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\langle M_5 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>\langle M_0 \rangle</td>
<td>\langle M_1 \rangle</td>
<td>\langle M_2 \rangle</td>
<td>\langle M_3 \rangle</td>
<td>\langle M_4 \rangle</td>
<td>\langle M_5 \rangle</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>M_0</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>M_1</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>M_2</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>M_3</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>M_4</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>M_5</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

“The language of all TMs that accept their own description.”
<table>
<thead>
<tr>
<th>\langle M_0 \rangle</th>
<th>\langle M_1 \rangle</th>
<th>\langle M_2 \rangle</th>
<th>\langle M_3 \rangle</th>
<th>\langle M_4 \rangle</th>
<th>\langle M_5 \rangle</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M_0</strong></td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td><strong>M_1</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td><strong>M_2</strong></td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td><strong>M_3</strong></td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
</tr>
<tr>
<td><strong>M_4</strong></td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
</tr>
<tr>
<td><strong>M_5</strong></td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

\{ \langle M \rangle \mid M \text{ is a TM that accepts } \langle M \rangle \}
<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\langle M_5 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_1$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$M_5$</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

$\{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \in \mathcal{L}(M) \}$
<table>
<thead>
<tr>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\langle M_5 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

This language is $\mathcal{L}_D$. 

$\{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \in \mathcal{L}(M) \}$
Here's an TM for $\overline{L}_D$:

$$R = \text{"On input } \langle M \rangle:\text{ run } M \text{ on } \langle M \rangle.\text{ If } M \text{ accepts } \langle M \rangle, \text{ accept. If } M \text{ rejects } \langle M \rangle, \text{ reject."}$$

Then $R$ accepts $\langle M \rangle$ iff $\langle M \rangle \in \mathcal{L}(M)$ iff $\langle M \rangle \in \overline{L}_D$, so $\mathcal{L}(R) = \overline{L}_D$. 

$\overline{L}_D \in \text{RE}$
Is $\overline{L_D}$ Decidable?

- We know that $\overline{L_D} \in \text{RE}$. Is $\overline{L_D} \in \text{R}$?

- No – by a similar argument from before.
  - If $\overline{L_D} \in \text{R}$, then $\overline{\overline{L_D}} = L_D \in \text{R}$.
  - Since $\text{R} \subset \text{RE}$, this means that $L_D \in \text{RE}$.
  - This contradicts that $L_D \notin \text{RE}$.
  - So our assumption is wrong and $\overline{L_D} \notin \text{R}$. 

The Limits of Computability

- Regular Languages
- DCFLs
- CFLs
- All Languages
- $A_{TM}$
- $L_D$
- $\overline{L_D}$
- $RE$
- $\overline{A_{TM}}$
- $L_D$

All Languages
Finding Unsolvable Problems

\[ L_D \rightarrow \overline{A}_{TM} \rightarrow A_{TM} \]

\[ \overline{L}_D \rightarrow \text{Not } \mathbf{R} \]

\[ \text{Not } \mathbf{RE} \]

\[ \text{Not } \mathbf{RE} \]

\[ \text{Not } \mathbf{R} \]