Reductions
The Limits of Computability

- Regular Languages
- CFLs
- DCFLs

- All Languages

- \( \overline{A_{TM}} \)
- \( L_D \)
- \( \overline{HALT} \)
- \( A_{TM} \)
The language $HALT$ is defined as

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

Equivalently:

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

Thus $\overline{HALT}$ is

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

Equivalently: \{x \mid x = \langle M, w \rangle \text{ for some TM } M \text{ and string } w, \text{ and } M \text{ halts on } w\}.

Thus, \text{HALT} 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 halt on } w\}.

That looks hard.
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:

```cpp
int main() {
    return 0;
}
```
HALT and HALT

- The language $HALT$ is defined as
  \[
  \{\langle M, w \rangle \mid M \text{ is a TM that halts on } w\}.
  \]
- Thus $\overline{HALT}$ is the language
  \[
  \{\langle M, w \rangle \mid M \text{ is a TM that doesn't halt on } w\}.
  \]
- Equivalently:
  \[
  HALT = \{\langle M, w \rangle \mid M \text{ is a TM that loops on } w\}.
  \]
The language $HALT$ is defined as:

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

Thus $\overline{HALT}$ is the language:

$$\{\langle M, w \rangle \mid M \text{ is a TM that doesn't halt on } w\}$$

Equivalently:

$$\overline{HALT} = \{\langle M, w \rangle \mid M \text{ is a TM that loops on } w\}$$
The Takeaway Point

- When dealing with encodings, you don't need to consider strings that aren't valid encodings.
- This will keep our proofs much simpler than before.
Reductions
Finding Unsolvable Problems

- Last time, we found five unsolvable problems.
- We proved that $L_D$ was unrecognizable, then used this fact to show four other languages were either undecidable or unrecognizable.
- In general, to prove that a problem is unsolvable (not $R$ or not $RE$), we don't directly show that it is unsolvable.
- Instead, we show how a solution to that problem would let us solve an unsolvable problem.
Reductions

$\varphi \equiv \psi$?

Can be converted to

Can be used to solve

Tautology
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 $f(w) \in B$
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

  \begin{align*}
  \text{For any } w \in \Sigma_1^*, \ w \in A \iff f(w) \in B
  \end{align*}

- Every $w \in A$ maps to some $f(w)$ in $B$.
- Every $w \notin A$ maps to some $f(w)$ not in $B$.
- $f$ does not have to be injective or surjective.
Reducing $\varphi \equiv \psi$ to Tautology

- Let $EQUIV$ be
  \[
  EQUIV = \{ \langle \varphi, \psi \rangle \mid \varphi \equiv \psi \}
  \]
- Let $TAUTOLOGY$ be
  \[
  TAUTOLOGY = \{ \langle \varphi \rangle \mid \varphi \text{ is a tautology} \}
  \]
- To reduce $EQUIV$ to $TAUTOLOGY$, we want a function $f$ such that
  \[
  \langle \varphi, \psi \rangle \in EQUIV \text{ iff } f(\langle \varphi, \psi \rangle) \in TAUTOLOGY
  \]
- One possible function we could use is
  \[
  f(\langle \varphi, \psi \rangle) = \langle \varphi \leftrightarrow \psi \rangle
  \]
Reducing any RE Language to $A_{TM}$

- Let $L$ be any RE language, and let $R$ be a recognizer for $L$.
- To reduce $L$ to $A_{TM}$, we want a function $f$ such that

$$w \in L \iff f(w) \in A_{TM}$$

- One possible reduction is

$$f(w) = \langle R, w \rangle$$
Why Reductions Matter

• If problem $A$ reduces to problem $B$, we can use a recognizer/decider for $B$ to recognize/decide problem $A$.
  
  • (There's a slight catch – we'll talk about this in a second).

• How is this possible?
$w \in A \iff f(w) \in B$

$H = \text{"On input } w:\n\begin{align*}
\text{Compute } f(w). \\
\text{Run } M \text{ on } f(w). \\
\text{If } M \text{ accepts } f(w), \text{ accept } w. \\
\text{If } M \text{ rejects } f(w), \text{ reject } w.\end{align*}$

$H \text{ accepts } w \iff M \text{ accepts } f(w) \iff f(w) \in B \iff w \in A$
A Problem

- Recall: $f$ is a reduction from $A$ to $B$ iff
  \[ w \in A \iff f(w) \in B \]
- Under this definition, any language $A$ reduces to any language $B$ unless $B = \emptyset$ or $\Sigma^*$.
- Since $B \neq \emptyset$ and $B \neq \Sigma^*$, there is some $w_{yes} \in B$ and some $w_{no} \notin B$.
- Define $f : \Sigma_1^* \rightarrow \Sigma_2^*$ as follows:
  \[
  \begin{align*}
  &\text{If } w \in A, \text{ then } f(w) = w_{yes} \\
  &\text{If } w \notin A, \text{ then } f(w) = w_{no}
  \end{align*}
  \]
- Then $f$ is a reduction from $A$ to $B$. 
A Problem

• Example: let's reduce $L_D$ to $0^*1^*$.

• Take $w_{\text{yes}} = 01$, $w_{\text{no}} = 10$.

• Then $f(w)$ is defined as
  • If $w \in L_D$, $f(w) = 01$.
  • If $w \notin L_D$, $f(w) = 10$.

• There is no TM that can actually evaluate the function $f(w)$ on all inputs, since no TM can decide whether or not $w \in L_D$. 
Example: let's reduce $L_D$ to $\emptyset^* 1^*$. 

Take $w_{yes} = 01$, $w_{no} = 10$. 

Then $f(w)$ is defined as:

- If $w \in L_D$, $f(w) = 01$.
- If $w \notin L_D$, $f(w) = 10$.

There is no TM that can actually evaluate the function $f(w)$ on all inputs, since no TM can decide whether or not $w \in L_D$. That's bad!
Computable Functions

- This general reduction is mathematically well-defined, but might be impossible to actually compute!
- To fix our definition, we need to introduce the idea of a computable function.
- A function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is called a **computable function** if there is some TM $M$ with the following behavior:
  
  "On input $w$:
  
  Determine the value of $f(w)$.
  
  Write $f(w)$ on the tape.
  
  Move the tape head back to the far left.
  
  Halt."
Computable Functions

\[ f(w) = ww \]
Computable Functions

\[ f(w) = ww \]
Computable Functions

\[ f(w) = \begin{cases} 2^{nm} & \text{if } w = 0^n1^m \\ \epsilon & \text{otherwise} \end{cases} \]
Computable Functions

\[ f(w) = \begin{cases} 
2^{nm} & \text{if } w = 0^n1^m \\
\varepsilon & \text{otherwise}
\end{cases} \]
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 $A$ to $B$, we say that $A$ is mapping reducible to $B$.

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

• This is not a partial order (it's not antisymmetric), but it is reflexive and transitive. (Why?)
Why Mapping Reducibility Matters

- **Theorem**: If $B \in \mathbf{R}$ and $A \leq_M B$, then $A \in \mathbf{R}$.

- **Theorem**: If $B \in \mathbf{RE}$ and $A \leq_M B$, then $A \in \mathbf{RE}$.

- $A \leq_M B$ informally 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 $A \notin \text{RE}$ and $A \leq M B$, then $B \notin \text{RE}$.
- $A \leq M B$ informally means “$B$ is at at least as hard as $A$. ”
Why Mapping Reducibility Matters

$A \leq_{M} B$

If this one is "easy" (R or RE)...

... then this one is "easy" (R or RE) too.
Why Mapping Reducibility Matters

If this one is “hard” (not $R$ or not RE)…

$A \leq_{M} B$

… then this one is “hard” (not $R$ or not RE) too.
\[ A \leq_M B \]

\[ w \leftarrow \text{Compute } f \rightarrow f(w) \rightarrow \text{Machine for } B \]

Machine \( M' \)

**Machine \( M' \)**

\[ M' = \text{"On input } w: \]
\[ \begin{align*}
\text{Compute } f(w). \\
\text{Run } M \text{ on } f(w). \\
\text{If } M \text{ accepts } f(w), \text{ accept } w. \\
\text{If } M \text{ rejects } f(w), \text{ reject } w. 
\end{align*} \]

**\( M' \) accepts \( w \)**

iff

**\( M \) accepts \( f(w) \)**

iff

\[ f(w) \in B \]

iff

\[ w \in A \]
$M'$ accepts $w$ iff $M$ accepts $f(w)$ iff $f(w) \in B$ iff $w \in A$
Using Reductions
Using Reductions

- Recall: The language $A_{TM}$ is defined as

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

- Last time, we proved that $A_{TM} \in \text{RE} - \text{R}$ (that is, $A_{TM} \in \text{RE}$ but $A_{TM} \notin \text{R}$) by showing that a decider for $A_{TM}$ could be converted into a decider for the diagonalization language $L_D$.

- Let's see an alternate proof that $A_{TM}$ is undecidable by using reductions.
The Complement of $A_{TM}$

- Recall: if $A_{TM} \in \mathbb{R}$, then $\overline{A}_{TM} \in \mathbb{R}$ as well.
- To show that $A_{TM}$ is undecidable, we will prove that the complement of $A_{TM}$ (denoted $\overline{A}_{TM}$) is undecidable.
- The language $\overline{A}_{TM}$ is the following:

$$\overline{A}_{TM} = \{(M, w) \mid M \text{ is a TM and } w \notin L(M)\}$$
\[ L_D \leq_M \overline{A}_{TM} \]

- Recall: The diagonalization language \( L_D \) is the language
  \[ L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathcal{L}(M) \} \]
- We directly established that \( L_D \notin \text{RE} \) using a diagonal argument.
- If we can show that \( L_D \leq_M \overline{A}_{TM} \), then since \( L_D \notin \text{RE} \), we have proven that \( \overline{A}_{TM} \notin \text{RE} \).
- Therefore, \( \overline{A}_{TM} \notin \mathbb{R} \), so \( A_{TM} \notin \mathbb{R} \).
Where We're Going

Goal: Choose our function $f(w)$ such that this machine $H$ is a recognizer for $L_D$.
$L_D$ and $\bar{A}_{TM}$

- $L_D$ and $\bar{A}_{TM}$ are similar languages:
  \[
  \langle M \rangle \in L_D \iff \langle M \rangle \notin \mathcal{L}(M) \\
  \langle M, w \rangle \in \bar{A}_{TM} \iff w \notin \mathcal{L}(M)
  \]

- $\bar{A}_{TM}$ is more general than $L_D$:
  - $L_D$ asks if a machine doesn't accept itself.
  - $\bar{A}_{TM}$ asks if a machine doesn't accept some specific string.
\[ L_D \leq M \overline{A}_{TM} \]

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

- Simplifying this using the definition of \( L_D \)
  \[ \langle M \rangle \notin \mathcal{L}(M) \quad \text{iff} \quad f(\langle M \rangle) \in \overline{A}_{TM} \]

- Let's assume that \( f(\langle M \rangle) \) has the form \( \langle M', w \rangle \) for some TM \( M' \) and string \( w \). This means that
  \[ \langle M \rangle \notin \mathcal{L}(M) \quad \text{iff} \quad \langle M', w \rangle \in \overline{A}_{TM} \]
  \[ \langle M \rangle \notin \mathcal{L}(M) \quad \text{iff} \quad w \notin \mathcal{L}(M') \]

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

- Choose \( M' = M \) and \( w = \langle M \rangle \).
What We Just Did

\[ H = \text{"On input } \langle M \rangle \text{:}
\]
\[ \text{Compute } \langle M, \langle M \rangle \rangle.
\]
\[ \text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.
\]
\[ \text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ accept } \langle M \rangle.
\]
\[ \text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ reject } \langle M \rangle."
\]

\[ H \text{ accepts } \langle M \rangle \iff R \text{ accepts } \langle M, \langle M \rangle \rangle \iff \langle M, \langle M \rangle \rangle \in A_{\text{TM}} \iff \langle M \rangle \notin L_M \iff \langle M \rangle \in L_D \]
The final version of our function $f$ is defined here:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

It's reasonable to assume that $f$ is computable; details are left as an exercise.

If we can formally prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$, then we have that $L_D \leq_M \overline{A}_{TM}$. Thus $\overline{A}_{TM} \notin RE$. 

$$L_D \leq_M \overline{A}_{TM}$$
Theorem: $\overline{A}_{TM} \notin RE$. 
Theorem: $\overline{A_{TM}} \notin \text{RE}$.

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A_{TM}}$. 

Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in A_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in A_{TM}$ iff $\langle M, \langle M \rangle \rangle \in A_{TM}$. By definition of $A_{TM}$, $\langle M, \langle M \rangle \rangle \in A_{TM}$ iff $\langle M \rangle \notin \ell_M$. Finally, note that $\langle M \rangle \notin \ell_M$ iff $\langle M \rangle \in L_D$.

Thus $f(\langle M \rangle) \in A_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A_{TM}}$.

Since $f$ is a mapping reduction from $L_D$ to $\overline{A_{TM}}$, we have $L_D \leq_M \overline{A_{TM}}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M \overline{A_{TM}}$, this means $\overline{A_{TM}} \notin \text{RE}$, as required. ■
Theorem: \( \overline{A_{TM}} \notin \text{RE} \).

Proof: We exhibit a mapping reduction \( f \) from \( L_D \) to \( \overline{A_{TM}} \).
Consider the function \( f \) defined as follows:
\[
f(\langle M \rangle) = \langle M, \langle M \rangle \rangle
\]
Theorem: $\overline{A}_{TM} \notin RE$.

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$.

Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof.
Theorem: $\overline{A}_{TM} \notin \text{RE}$.  

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$. Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$.

Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$. Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M \overline{A}_{TM}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M \overline{A}_{TM}$, this means $\overline{A}_{TM} \notin \text{RE}$, as required. ■
**Theorem:** $\overline{A}_{TM} \notin \text{RE}$.  

**Proof:** We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$. Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. 

Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$. Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M \overline{A}_{TM}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M \overline{A}_{TM}$, this means $\overline{A}_{TM} \notin \text{RE}$, as required. ■
Theorem: $\overline{A}_{TM} \notin \text{RE}$.

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$. Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. By definition of $\overline{A}_{TM}$, $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$ iff $\langle M \rangle \notin \mathcal{L}(M)$. 

Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$.

Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M \overline{A}_{TM}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M \overline{A}_{TM}$, this means $\overline{A}_{TM} \notin \text{RE}$, as required. $\blacksquare$
**Theorem:** $\overline{A}_{TM} \notin \text{RE}$.  

**Proof:** We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$. Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. By definition of $\overline{A}_{TM}$, $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Finally, note that $\langle M \rangle \notin \mathcal{L}(M)$ iff $\langle M \rangle \in L_D$.  

Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M A_{TM}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M A_{TM}$, this means $A_{TM} \notin \text{RE}$, as required. $\blacksquare$
Theorem: $\overline{A}_{TM} \notin \text{RE}$.

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$.
Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. By definition of $\overline{A}_{TM}$, $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$ iff $\langle M \rangle \notin \mathcal{R}(M)$. Finally, note that $\langle M \rangle \notin \mathcal{R}(M)$ iff $\langle M \rangle \in L_D$. Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$. 

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

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A_{TM}}$.

Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A_{TM}}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A_{TM}}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A_{TM}}$. By definition of $\overline{A_{TM}}$, $\langle M, \langle M \rangle \rangle \in \overline{A_{TM}}$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Finally, note that $\langle M \rangle \notin \mathcal{L}(M)$ iff $\langle M \rangle \in L_D$.

Thus $f(\langle M \rangle) \in \overline{A_{TM}}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A_{TM}}$.

Since $f$ is a mapping reduction from $L_D$ to $\overline{A_{TM}}$, we have $L_D \leq_M \overline{A_{TM}}$. 
Theorem: $\overline{A}_{TM} \notin RE$.

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$. Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. By definition of $\overline{A}_{TM}$, $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$ iff $\langle M \rangle \notin D(M)$. Finally, note that $\langle M \rangle \notin D(M)$ iff $\langle M \rangle \in L_D$.

Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$.

Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M \overline{A}_{TM}$. Since $L_D \notin RE$ and $L_D \leq_M \overline{A}_{TM}$, this means $\overline{A}_{TM} \notin RE$, as required.
Theorem: $\overline{A}_{TM} \notin \text{RE}$.

Proof: We exhibit a mapping reduction $f$ from $L_D$ to $\overline{A}_{TM}$.

Consider the function $f$ defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that $f$ can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \overline{A}_{TM}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$. By definition of $\overline{A}_{TM}$, $\langle M, \langle M \rangle \rangle \in \overline{A}_{TM}$ iff $\langle M \rangle \notin \mathcal{D}(M)$. Finally, note that $\langle M \rangle \notin \mathcal{D}(M)$ iff $\langle M \rangle \in L_D$.

Thus $f(\langle M \rangle) \in \overline{A}_{TM}$ iff $\langle M \rangle \in L_D$, so $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$.

Since $f$ is a mapping reduction from $L_D$ to $\overline{A}_{TM}$, we have $L_D \leq_M \overline{A}_{TM}$. Since $L_D \notin \text{RE}$ and $L_D \leq_M \overline{A}_{TM}$, this means $\overline{A}_{TM} \notin \text{RE}$, as required. ■
The Halting Problem

• Recall the definition of \( HALT \):

\[
HALT = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on } w \}
\]

• That is, the set of TM / string pairs where the TM \( M \) either accepts or rejects the string \( w \).

• Last time, we proved that \( HALT \in \text{RE} - \text{R} \) by building a TM for it, then by showing a decider for \( HALT \) could be turned into a decider for \( A_{\text{TM}} \).

• Let's explore an alternate proof using mapping reductions.
HALT is RE

- Recall: $A_{TM} \in \mathbf{RE}$.

- To prove that HALT is RE, we will show that HALT $\leq^M A_{TM}$.

- Since $A_{TM} \in \mathbf{RE}$, this proves HALT $\in \mathbf{RE}$.

- Idea: we need to find some function $f$ such that

  $\langle M, w \rangle \in HALT \iff f(\langle M, w \rangle) \in A_{TM}$
Where We're Going

\[ \langle M, w \rangle \xrightarrow{\text{Compute } f} \langle M', w' \rangle \xrightarrow{\text{Machine for } A_{TM}} \]

**Machine H**

**Goal:** Choose our function \( f(w) \) such that this machine \( H \) is a recognizer for \( \text{HALT} \).
\[ \text{HALT} \leq_M A_{\text{TM}} \]

- Goal: Find a function \( f \) such that
  \[ \langle M, w \rangle \in \text{HALT} \iff f(\langle M, w \rangle) \in A_{\text{TM}} \]

- Substituting the definitions:
  \[ M \text{ halts on } w \iff f(\langle M, w \rangle) \in A_{\text{TM}}. \]

- Assume that \( f(\langle M, w \rangle) = \langle M', w' \rangle \) for some TM \( M' \) and string \( w' \). Then we have
  \[ M \text{ halts on } w \iff \langle M', w' \rangle \in A_{\text{TM}} \]
  \[ M \text{ halts on } w \iff w' \in \mathcal{L}(M') \]
  \[ M \text{ halts on } w \iff M' \text{ accepts } w' \]
Choosing $M'$ and $w'$

- We need to find $M'$ and $w'$ such that
  \[ M \text{ halts on } w \iff M' \text{ accepts } w'. \]
- This is the creative step of the proof – how do we choose an $M'$ and $w'$ with that property?
- **Key idea that shows up in almost all major reduction proofs**: Construct a machine $M'$ and string $w'$ so that running $M'$ on $w'$ runs $M$ on $w$.
- This causes the behavior of $M'$ running on $w'$ to depend on what $M$ does on $w$. 
Choosing $M'$ and $w'$

• Here is one possible choice of $M'$ and $w'$ we can make:

$$M' = \text{“On input } \langle N, z \rangle:\text{ Run } N \text{ on } z. \text{ If } N \text{ halts on } z, \text{ accept.”}$$

$$w' = \langle M, w \rangle$$

• Now, running $M'$ on $w'$ runs $M$ on $w$. If $M$ halts on $w$, then $M'$ accepts $w'$. If $M$ loops on $w$, then $M'$ does not accept $w'$. 
Machine $H$ = “On input $\langle M, w \rangle$:

- Compute $f$.
- Run $R$ on $\langle M', \langle M, w \rangle \rangle$.
- If $R$ accepts $\langle M', \langle M, w \rangle \rangle$, accept.
- If $R$ rejects $\langle M', \langle M, w \rangle \rangle$, reject.”

$H$ accepts $\langle M, w \rangle$ iff $R$ accepts $\langle M', \langle M, w \rangle \rangle$ iff $\langle M', \langle M, w \rangle \rangle \in A_{TM}$ iff $M'$ accepts $\langle M, w \rangle$ iff $M$ halts on $w$ iff $\langle M, w \rangle \in HALT$.

$M' = \text{“On input } \langle N, z \rangle:\n
\hspace{2em} \text{Run } N \text{ on } z.\n\hspace{2em} \text{If } N \text{ halts, accept.”}
Theorem: $\text{HALT} \leq_{M} A_{\text{TM}}$. 
Theorem: $\text{HALT} \leq_{M} A_{TM}$.

Proof: We exhibit a mapping reduction $f$ from $\text{HALT}$ to $A_{TM}$.
Theorem: \( \text{HALT} \leq_{M} \text{A}_{\text{TM}} \).

Proof: We exhibit a mapping reduction \( f \) from \( \text{HALT} \) to \( \text{A}_{\text{TM}} \). Let the machine \( M' \) be defined as follows:
**Theorem:** $\text{HALT} \leq_A \text{A}_{TM}$.

**Proof:** We exhibit a mapping reduction $f$ from $\text{HALT}$ to $\text{A}_{TM}$. Let the machine $M'$ be defined as follows:

$$M' = \text{"On input } \langle N, z \rangle: \text{ Run } N \text{ on } z. \text{ If } N \text{ halts on } z, \text{ accept."}$$
Theorem: \( \text{HALT} \leq \text{M} \ A_{\text{TM}} \).

Proof: We exhibit a mapping reduction \( f \) from \( \text{HALT} \) to \( A_{\text{TM}} \).

Let the machine \( \text{M}' \) be defined as follows:

\[
\text{M}' = "\text{On input } \langle N, z \rangle:\n\quad \text{Run } N \text{ on } z.\n\quad \text{If } N \text{ halts on } z, \text{ accept.}"\]

Then let \( f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \).
Theorem: \( \text{HALT} \leq_{m} A_{\text{TM}} \).

Proof: We exhibit a mapping reduction \( f \) from \( \text{HALT} \) to \( A_{\text{TM}} \). Let the machine \( M' \) be defined as follows:

\[
M' = \text{"On input } \langle N, z \rangle:\ 
\quad \text{Run } N \text{ on } z.
\quad \text{If } N \text{ halts on } z, \text{ accept."
}\]

Then let \( f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \). We claim that \( f \) is computable and omit the details from this proof.
Theorem: \(\text{HALT} \leq_{M} A_{TM}\).

Proof: We exhibit a mapping reduction \(f\) from \(\text{HALT}\) to \(A_{TM}\).

Let the machine \(M'\) be defined as follows:

\[
M' = \text{"On input }\langle N, z \rangle: \\
\text{Run } N \text{ on } z. \\
\text{If } N \text{ halts on } z, \text{ accept."}
\]

Then let \(f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle\). We claim that \(f\) is computable and omit the details from this proof. We further claim that \(\langle M, w \rangle \in \text{HALT} \iff f(\langle M, w \rangle) \in A_{TM}\).
**Theorem:** $\text{HALT} \leq^M \text{A}_{\text{TM}}$.

**Proof:** We exhibit a mapping reduction $f$ from $\text{HALT}$ to $\text{A}_{\text{TM}}$. Let the machine $M'$ be defined as follows:

$$M' = \text{"On input } \langle N, z \rangle:\text{ Run } N \text{ on } z. \text{ If } N \text{ halts on } z, \text{ accept."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in \text{A}_{\text{TM}}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in \text{A}_{\text{TM}}$ iff $M'$ accepts $\langle M, w \rangle$. 


Theorem: $\text{HALT} \leq_{\text{m}} \text{A}_\text{TM}$.

Proof: We exhibit a mapping reduction $f$ from $\text{HALT}$ to $\text{A}_\text{TM}$.

Let the machine $M'$ be defined as follows:

\[
M' = \text{“On input } \langle N, z \rangle:\n\hspace{1cm} \text{Run } N \text{ on } z.
\hspace{1cm} \text{If } N \text{ halts on } z, \text{ accept.”}
\]

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in \text{A}_\text{TM}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in \text{A}_\text{TM}$ iff $M'$ accepts $\langle M, w \rangle$. By construction, $M'$ accepts $\langle M, w \rangle$ iff $M$ halts on $w$. 
Theorem: $\text{HALT} \leq_{M} A_{\text{TM}}$.

Proof: We exhibit a mapping reduction $f$ from $\text{HALT}$ to $A_{\text{TM}}$. Let the machine $M'$ be defined as follows:

$$M' = \text{"On input } \langle N, z \rangle:\text{ Run } N \text{ on } z. \text{ If } N \text{ halts on } z, \text{ accept."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in A_{\text{TM}}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in A_{\text{TM}}$ iff $M'$ accepts $\langle M, w \rangle$. By construction, $M'$ accepts $\langle M, w \rangle$ iff $M$ halts on $w$. Finally, note that $M$ halts on $w$ iff $\langle M, w \rangle \in \text{HALT}$. 

Theorem: \( \text{HALT} \leq_{M} A_{\text{TM}} \).

Proof: We exhibit a mapping reduction \( f \) from \( \text{HALT} \) to \( A_{\text{TM}} \).

Let the machine \( M' \) be defined as follows:

\[
M' = \text{“On input } \langle N, z \rangle:\]
  \[\text{Run } N \text{ on } z.\]
  \[\text{If } N \text{ halts on } z, \text{ accept.”}\]

Then let \( f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \). We claim that \( f \) is computable and omit the details from this proof. We further claim that \( \langle M, w \rangle \in \text{HALT} \) iff \( f(\langle M, w \rangle) \in A_{\text{TM}} \). To see this, note that \( f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in A_{\text{TM}} \) iff \( M' \) accepts \( \langle M, w \rangle \). By construction, \( M' \) accepts \( \langle M, w \rangle \) iff \( M \) halts on \( w \). Finally, note that \( M \) halts on \( w \) iff \( \langle M, w \rangle \in \text{HALT} \). Thus \( \langle M, w \rangle \in \text{HALT} \) iff \( f(\langle M, w \rangle) \in A_{\text{TM}} \).

\[\square\]
Theorem: $\text{HALT} \leq_{M} A_{TM}$.

Proof: We exhibit a mapping reduction $f$ from $\text{HALT}$ to $A_{TM}$.

Let the machine $M'$ be defined as follows:

$$M' = \text{"On input } \langle N, z \rangle:\$$
  Run $N$ on $z$.
  If $N$ halts on $z$, accept."

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in A_{TM}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in A_{TM}$ iff $M'$ accepts $\langle M, w \rangle$. By construction, $M'$ accepts $\langle M, w \rangle$ iff $M$ halts on $w$. Finally, note that $M$ halts on $w$ iff $\langle M, w \rangle \in \text{HALT}$. Thus $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in A_{TM}$. Therefore, $f$ is a mapping reduction from $\text{HALT}$ to $A_{TM}$, so $\text{HALT} \leq_{M} A_{TM}$. ■
**Theorem:** $\text{HALT} \leq_m A_{\text{TM}}$.  

**Proof:** We exhibit a mapping reduction $f$ from $\text{HALT}$ to $A_{\text{TM}}$. Let the machine $M'$ be defined as follows:

$$M' = \text{"On input } \langle N, z \rangle:\$$
$$\text{Run } N \text{ on } z.$$
$$\text{If } N \text{ halts on } z, \text{ accept."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in A_{\text{TM}}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in A_{\text{TM}}$ iff $M'$ accepts $\langle M, w \rangle$. By construction, $M'$ accepts $\langle M, w \rangle$ iff $M$ halts on $w$. Finally, note that $M$ halts on $w$ iff $\langle M, w \rangle \in \text{HALT}$. Thus $\langle M, w \rangle \in \text{HALT}$ iff $f(\langle M, w \rangle) \in A_{\text{TM}}$. Therefore, $f$ is a mapping reduction from $\text{HALT}$ to $A_{\text{TM}}$, so $\text{HALT} \leq_m A_{\text{TM}}$. ■
A Math Joke
A Math Joke
HALT is Undecidable

- We proved $HALT \in \text{RE}$ by showing that $HALT \leq^m A_{TM}$.

- We can prove $HALT \notin \text{R}$ by showing that $A_{TM} \leq^m HALT$.

- Note that this has to be a completely separate reduction! We're transforming $A_{TM}$ into $HALT$ this time, not the other way around.
\[ A_{\text{TM}} \leq M \text{HALT} \]

- We want to find a computable function \( f \) such that
  \[ \langle M, w \rangle \in A_{\text{TM}} \iff f(\langle M, w \rangle) \in \text{HALT}. \]
- Assume \( f(\langle M, w \rangle) \) has the form \( \langle M', w' \rangle \) for some TM \( M' \) and string \( w' \).
- We want
  \[ \langle M, w \rangle \in A_{\text{TM}} \iff \langle M', w' \rangle \in \text{HALT}. \]
- Substituting definitions:
  \[ M \text{ accepts } w \iff M' \text{ halts on } w'. \]
- How might we design \( M' \) and \( w' \)?
\[ A_{TM} \leq_M HALT \]

- We need to choose a TM/string pair \( M' \) and \( w' \) such that \( M' \) halts on \( w' \) iff \( M \) accepts \( w \).

- Repeated idea: Construct \( M' \) and \( w' \) such that running \( M' \) on \( w' \) simulates \( M \) on \( w \) and bases its decision on what happens.

- One option:

  \[
  M' = \text{"On input } \langle N, z \rangle: \\
  \text{Run } N \text{ on } z. \\
  \text{If } N \text{ accepts } z, \text{ accept.} \\
  \text{If } N \text{ rejects } z, \text{ loop infinitely."}
  \]

  \[ w' = \langle M, w \rangle \]
Machine $H$ = “On input $\langle M, w \rangle$:
    Compute $\langle M', \langle M, w \rangle \rangle$.
    Run $R$ on $\langle M', \langle M, w \rangle \rangle$.
    If $R$ accepts $\langle M', \langle M, w \rangle \rangle$, accept.
    If $R$ rejects $\langle M', \langle M, w \rangle \rangle$, reject.”

$M' = “On input $\langle N, z \rangle$:
    Run $N$ on $z$.
    If $N$ accepts, accept.
    If $N$ rejects, loop infinitely.”

$H$ accepts $\langle M, w \rangle$ iff $R$ accepts $\langle M', \langle M, w \rangle \rangle$ iff $\langle M', \langle M, w \rangle \rangle \in HALT$ iff $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$ iff $\langle M, w \rangle \in A_{TM}$. 
An Important Detail

• In the course of this reduction, we construct a new machine $M'$.

• We never actually run the machine $M'$! That might loop forever.

• We instead just build a description of that machine and fed it into our machine for $HALT$.

• The answer given back by this machine about what $M'$ would do if we were to run it can then be used to solve $A_{TM}$. 
Theorem: $A_{TM} \leq_M HALT$. 
Theorem: $A_{\text{TM}} \leq_M \text{HALT}$.

Proof: We exhibit a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$.
Theorem: $A_{TM} \leq_M HALT$.

Proof: We exhibit a mapping reduction from $A_{TM}$ to $HALT$. Let $M'$ be the following TM:

Let $M'$ be the following TM:
Theorem: $A_{TM} \leq_M HALT$.

Proof: We exhibit a mapping reduction from $A_{TM}$ to $HALT$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\n\text{Run } N \text{ on } z.\n\text{If } N \text{ accepts, accept.}\n\text{If } N \text{ rejects, loop infinitely."}$$
Theorem: $A_{\text{TM}} \leq_{M} \text{HALT}$.  

Proof: We exhibit a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$. Let $M'$ be the following TM:

\begin{quote}
$M' = "\text{On input } \langle N, z \rangle:"
\begin{itemize}
  \item Run $N$ on $z$.
  \item If $N$ accepts, accept.
  \item If $N$ rejects, loop infinitely."
\end{itemize}
\end{quote}

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. 
Theorem: $A_{TM} \leq_M HALT$.

Proof: We exhibit a mapping reduction from $A_{TM}$ to $HALT$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\n\text{Run } N \text{ on } z.\n\text{If } N \text{ accepts, accept.}\n\text{If } N \text{ rejects, loop infinitely."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof.
Theorem: $A_{\text{TM}} \leq_{\text{M}} \text{HALT}$.

Proof: We exhibit a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$. Let $M'$ be the following TM:

$$M' = \text{“On input } \langle N, z \rangle:\text{ }
\begin{align*}
\text{Run } N \text{ on } z. \\
\text{If } N \text{ accepts, accept.} \\
\text{If } N \text{ rejects, loop infinitely."
}\end{align*}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$. 
**Theorem:** \( A_{\text{TM}} \leq_M \text{HALT} \).

**Proof:** We exhibit a mapping reduction from \( A_{\text{TM}} \) to \( \text{HALT} \).

Let \( M' \) be the following TM:

\[
M' = \text{"On input } \langle N, z \rangle: \\
\text{Run } N \text{ on } z. \\
\text{If } N \text{ accepts, accept.} \\
\text{If } N \text{ rejects, loop infinitely."}
\]

Then let \( f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \). We claim that \( f \) is computable and omit the details from this proof. We further claim that \( \langle M, w \rangle \in A_{\text{TM}} \) iff \( f(\langle M, w \rangle) \in \text{HALT} \). To see this, note that \( f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in \text{HALT} \) iff \( M' \) halts on \( \langle M, w \rangle \).
**Theorem:** $A_{\text{TM}} \leq_M \text{HALT}$.  

**Proof:** We exhibit a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\text{ Run } N \text{ on } z. \text{ If } N \text{ accepts, accept. If } N \text{ rejects, loop infinitely."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in \text{HALT}$ iff $M'$ halts on $\langle M, w \rangle$. By construction, $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$. 
Theorem: $A_{TM} \leq_M HALT$.

Proof: We exhibit a mapping reduction from $A_{TM}$ to $HALT$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\$$
  - Run $N$ on $z$.
  - If $N$ accepts, accept.
  - If $N$ rejects, loop infinitely."

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in HALT$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in HALT$ iff $M'$ halts on $\langle M, w \rangle$. By construction, $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$. Finally, $M$ accepts $w$ iff $\langle M, w \rangle \in A_{TM}$. ■
Theorem: $A_{TM} \leq_M HALT$.

Proof: We exhibit a mapping reduction from $A_{TM}$ to $HALT$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\n\quad \text{Run } N \text{ on } z.\n\quad \text{If } N \text{ accepts, accept.}\n\quad \text{If } N \text{ rejects, loop infinitely."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in HALT$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in HALT$ iff $M'$ halts on $\langle M, w \rangle$. By construction, $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$. Finally, $M$ accepts $w$ iff $\langle M, w \rangle \in A_{TM}$. Thus we have that $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in HALT$. 

Thus we have that $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in HALT$. ■
**Theorem:** $A_{\text{TM}} \leq_M \text{HALT}$.  

**Proof:** We exhibit a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$. Let $M'$ be the following TM:

$$M' = "\text{On input } \langle N, z \rangle:\"$$

- **Run** $N$ on $z$.
- **If** $N$ accepts, accept.
- **If** $N$ rejects, loop infinitely.”

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in \text{HALT}$ iff $M'$ halts on $\langle M, w \rangle$. By construction, $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$. Finally, $M$ accepts $w$ iff $\langle M, w \rangle \in A_{\text{TM}}$. Thus we have that $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$. Therefore, $f$ is a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$, so $A_{\text{TM}} \leq_M \text{HALT}$. 

$\blacksquare$
Theorem: $A_{\text{TM}} \leq_M \text{HALT}$.  

Proof: We exhibit a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\n    \text{Run } N \text{ on } z.\n    \text{If } N \text{ accepts, accept.}\n    \text{If } N \text{ rejects, loop infinitely."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$. To see this, note that $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle \in \text{HALT}$ iff $M'$ halts on $\langle M, w \rangle$. By construction, $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$. Finally, $M$ accepts $w$ iff $\langle M, w \rangle \in A_{\text{TM}}$. Thus we have that $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$. Therefore, $f$ is a mapping reduction from $A_{\text{TM}}$ to $\text{HALT}$, so $A_{\text{TM}} \leq_M \text{HALT}$. ■
Theorem: $A_{TM} \leq_M HALT$.

Proof: We exhibit a mapping reduction from $A_{TM}$ to $HALT$. Let $M'$ be the following TM:

$$M' = \text{"On input } \langle N, z \rangle:\text{ Run } N \text{ on } z.\text{ If } N \text{ accepts, accept. If } N \text{ rejects, loop infinitely."}$$

Then let $f(\langle M, w \rangle) = \langle M', \langle M, w \rangle \rangle$. We claim that $f$ is computable and omit the details from this proof. We further claim that $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in HALT$. To see this, note that $M'$ halts on $\langle M, w \rangle$ iff $M$ accepts $w$. Thus we have that $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in HALT$. Therefore, $f$ is a mapping reduction from $A_{TM}$ to $HALT$, so $A_{TM} \leq_M HALT$. ■
A Note on Directionality
Note the Direction

- To show that a language $A$ is $\text{RE}$, reduce it to something that is known to be $\text{RE}$:
  \[ A \leq^M \text{some-RE-problem} \]

- To show that a language $A$ is not $\text{R}$, reduce a problem that is known not to be $\text{R}$ to $A$:
  \[ \text{some-non-R-problem} \leq^M A \]

- The single most common mistake with reductions is doing the reduction in the wrong direction.
Next Time

• **co-RE and Beyond**
  • What lies outside of RE? How much of it can be solved by computers?

• **More Reductions**
  • More examples of mapping reductions.