Undecidability
A_{TM} is not Turing-Decidable

Theorem. $A_{TM} = \{ <M, w> \mid M \text{ is a TM and } M \text{ accepts } w \}$ is undecidable.

Proof.

- We assumed there was a Turing Machine $H$ that decides $A_{TM}$.
- Created a Turing Machine $D$ that uses $H$.
- Reached a contradiction.
- We can conclude Turing Machine $H$ cannot exist.
- Therefore, $A_{TM}$ is undecidable.
Turing Recognizable vs Turing Decidable

**Theorem.** A language is Turing-decidable if and only if both it and its complement are Turing-recognizable.

**Proof.** (=>) By definition.

(<=) Simulate, in parallel, $M_L$ on tape 1 and $M_{\overline{L}}$ on tape 2
$\overline{A}_{TM}$ is not even Turing-recognizable

Lemma: $\overline{A}_{TM}$ is not Turing-recognizable.

Proof idea: How can we use the previous theorem to show this?
More Undecidability

- We know $A_{TM}$ is undecidable, but what do other undecidable problems look like?
- We will prove some other problems are undecidable (unsolvable by a TM)
- Primary method of showing problems are unsolvable: reductions
Looking at $A_{TM}$ once again

$U = \text{“On input } <M, w>, \text{ where } M \text{ is a TM and } w \text{ a string:} \text{”}$

1. Simulate $M$ on input $w$.

2. If $M$ ever enters its accept state, accept. If $M$ ever enters its reject state, reject.”

3 possibilities:
- $M$ halts and accepts
- $M$ halts and rejects
- $M$ does not halt
Looking at $A_{TM}$ once again

$U = \text{“On input } <M, w>\text{, where } M \text{ is a TM and } w \text{ a string:} \text{”}$

1. Simulate $M$ on input $w$.

2. If $M$ ever enters its accept state, accept. If $M$ ever enters its reject state, reject.”

3 possibilities:
- $M$ halts and accepts
- $M$ halts and rejects
- $M$ does not halt
The Halting Problem

We could use $U$ to decide $A_{TM}$ if we could determine whether $M$ would halt on input $w$.

“On input $<M, w>$, where $M$ is a TM and $w$ a string:

1. Determine whether $M$ on input $w$ will halt. If not, then reject.
2. Simulate $M$ on input $w$.
3. If $M$ enters its accept state, accept. If $M$ enters its reject state, reject.”
The Halting Problem

We could use \( U \) to decide \( A_{TM} \) if we could determine whether \( M \) would halt on input \( w \).

“On input \( <M, w> \), where \( M \) is a TM and \( w \) a string:

1. Determine whether \( M \) on input \( w \) will halt. If not, then reject.
2. Simulate \( M \) on input \( w \).
3. If \( M \) enters its accept state, accept. If \( M \) enters its reject state, reject.”

Definition. \( HALT_{TM} = \{ <M, w> | M \text{ is a TM and } M \text{ halts on input } w \} \).
**HALT**$_{TM}$ is undecidable

**Proof.** Suppose TM $R$ decides $HALT_{TM}$. Define

$S = \text{"On input } <M, w>, \text{ where } M \text{ is a TM and } w \text{ a string:}

1. Run TM $R$ on input $<M, w>$.
2. If $R$ rejects, then reject.
3. If $R$ accepts, simulate $M$ on input $w$ until it halts.
4. If $M$ enters its accept state, accept. If $M$ enters its reject state, reject."
**HALT**\textsubscript{TM} is undecidable

**Proof.** Suppose TM \( R \) decides \( HALT \textsubscript{TM} \). Define

\( S = \) “On input \(<M, w>\), where \( M \) is a TM and \( w \) a string:

1. Run TM \( R \) on input \(<M, w>\).
2. If \( R \) rejects, then reject.
3. If \( R \) accepts, simulate \( M \) on input \( w \) until it halts.
4. If \( M \) enters its accept state, accept. If \( M \) enters its reject state, reject.”

**Question.** Where is the contradiction?
Reducibility
Reducibility

- Problem $A$ reduces to Problem $B$
- If we can use the solution to $B$ to solve $A$, that is,
- Solution to $B$ leads to Solution to $A$
Using Reductions

- We used $A_{TM}$ to show that $HALT_{TM}$ is undecidable

- Which problem played the role of problem A? Which was problem B?

Solution to $B$ leads to Solution to $A$
Does $M$ Accept Anything at All?

**Definition.** $E_{TM} = \{ <M> \mid M \text{ is a TM and } L(M) = \emptyset \}$

**Theorem.** $E_{TM}$ is undecidable.
$E_{TM}$ is undecidable

**Proof.** Suppose TM $R$ decides $E_{TM}$.

**Question.** Can $R$ be used to decide $A_{TM}$?
$E_{\text{TM}}$ is undecidable

Given an input $<M, w>$ we construct an intermediate TM $M_w$ as follows:

$M_w =$ “On input $x$:
1. If $x \neq w$, reject.
2. If $x = w$, run $M$ on input $w$ and accept if $M$ does.”

Note that $w$ is not given as input to $M_w$ (it is hardcoded in its description).
$E_{TM}$ is undecidable

Given an input $<M, w>$ we construct an intermediate TM $M_w$ as follows:

$M_w =$ “On input $x$:

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

Note that $w$ is not given as input to $M_w$ (it is hardcoded in its description).

*Question.* If $M$ accepts $w$, what can we reason about $L(M_w)$?
$E_{TM}$ is undecidable

**Proof.** Suppose TM $R$ decides $E_{TM}$.

Define $S =$ “On input $<M, w>$:

1. Use the description of $M$ and $w$ to construct $M_w$.
2. Run $R$ on input $<M_w>$.
3. If $R$ accepts, reject. If $R$ rejects, accept.”
Construct $M_w$ s.t. $L(M_w)$ is nonempty if and only if $M$ accepts $w$

If $x \neq w$, reject.
Else run $M$ on $w$, accept if $M$ accepts

$S$: Decider for $A_{TM}$

$R$: Black-box decider for $E_{TM}$
Exercise

- \( EQ_{\text{TM}} = \{ <M_1, M_2> | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \).

Show that \( EQ_{\text{TM}} \) is undecidable by reducing \( E_{\text{TM}} \) to \( EQ_{\text{TM}} \).

- \( \text{REGULAR}_{\text{TM}} = \{ <M> | M \text{ is a TM and } L(M) \text{ is regular}\} \).

Show that \( \text{REGULAR}_{\text{TM}} \) is undecidable by reducing \( A_{\text{TM}} \) to \( \text{REGULAR}_{\text{TM}} \).
Mapping Reducibility

- A “computable” function $f$ exists that converts instances of Problem $A$ to Problem $B$.

- If have such a conversion, called reduction, we can use the solver for Problem $B$ to solve Problem $A$. 
Computable Function

- It is a function that a Turing machine can compute

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

- For example, arithmetic functions
  - TM that takes as input \(<m, n>\), and returns \( m+n \) on its tape.
Mapping Reducibility

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

$$w \in A \iff f(w) \in B.$$
$EQ_{TM}$ is undecidable

- We showed this by defining a reduction from .... ?

- Considering this reduction
  - What is A?
  - What is B?
  - What is function $f$?
$EQ_{TM}$ is undecidable

- We showed this by defining a reduction from .... ?

- Considering this reduction
  - What is $A$?
  - What is $B$?
  - What is function $f$?

- We can say that $E_{TM} \leq_{m} EQ_{TM}$