Mapping Reductions
A Concrete Undecidable Problem: The Acceptance Problem for TMs

$A_{TM} = \{(M, w) \mid M \text{ is a TM that accepts string } w \}$

Theorem [Turing ‘30s]: $A_{TM}$ is recognizable but NOT decidable

Corollary: $\neg A_{TM}$ is not recognizable
The Halting Problem

\[ \text{HALT}_{TM} = \{ (M,w) \mid M \text{ is a TM that halts on string } w \} \]

Theorem: \( \text{HALT}_{TM} \) is undecidable

Proof: Assume (for a contradiction) there is a TM \( H \) that decides \( \text{HALT}_{TM} \)

We use \( H \) to construct a TM \( M' \) that decides \( \text{A}_{TM} \)

\( M'(M,w) \): Run \( H(M,w) \)
If \( H \) rejects then reject
If \( H \) accepts, run \( M \) on \( w \) until it halts:
If \( M \) accepts, then accept
If \( M \) rejects, then reject
If $M$ doesn't halt: reject

If $M$ halts

Does $M$ halt on $w$?
Can often prove a language \( L \) is undecidable by proving: if \( L \) is decidable, then so is \( A_{TM} \).

We reduce \( A_{TM} \) to the language \( L \):

\[
A_{TM} \leq_m L
\]
Mapping Reductions

\( f : \Sigma^* \rightarrow \Sigma^* \) is a computable function if there is a Turing machine \( M \) that halts with just \( f(w) \) written on its tape, for every input \( w \)

A language \( A \) is **mapping reducible** to language \( B \), written as \( A \leq_m B \), if there is a computable \( f : \Sigma^* \rightarrow \Sigma^* \) such that for every \( w \),

\[ w \in A \iff f(w) \in B \]

\( f \) is called a mapping reduction (or many-one reduction) from \( A \) to \( B \)
Let $f : \Sigma^* \to \Sigma^*$ be a computable function such that $w \in A \iff f(w) \in B$

Say: $A$ is mapping reducible to $B$
Write: $A \leq_m B$
Theorem: If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$
Theorem: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable

Proof: Let $M$ decide $B$.
Let $f$ be a mapping reduction from $A$ to $B$
To decide $A$, we build a machine $M'$

$M'(w)$:

1. Compute $f(w)$
2. Run $M$ on $f(w)$, output its answer

- $w \in A \iff f(w) \in B$ so $w \in A \implies M'$ accepts $w$
- $w \notin A \implies M'$ rejects $w$
Theorem: If $A \leq_m B$ and $B$ is recognizable, then $A$ is recognizable.

Proof: Let $M$ recognize $B$. Let $f$ be a mapping reduction from $A$ to $B$. To recognize $A$, we build a machine $M'$.

$M'(w)$:

1. Compute $f(w)$
2. Run $M$ on $f(w)$, output its answer if you ever receive one.
Theorem: If \( A \leq_m B \) and \( B \) is decidable, then \( A \) is decidable

Corollary: If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable

Theorem: If \( A \leq_m B \) and \( B \) is recognizable, then \( A \) is recognizable

Corollary: If \( A \leq_m B \) and \( A \) is unrecognizable, then \( B \) is unrecognizable
The proof that the Halting Problem is undecidable can be seen as constructing a mapping reduction from $A_{TM}$ to $HALT_{TM}$

Theorem: $A_{TM} \leq_m HALT_{TM}$

$f(M, w) := (M', w)$ where

“$M'(w) = \text{accepts if } M(w) \text{ accepts else loops forever}”$ how?

We have $(M, w) \in A_{TM} \iff (M', w) \in HALT_{TM}$
Theorem: $A_{TM} \leq_{m} \text{HALT}_{TM}$

Corollary: $\neg A_{TM} \leq_{m} \neg \text{HALT}_{TM}$

Proof? 

Corollary: $\neg \text{HALT}_{TM}$ is unrecognizable!

Proof: If $\neg \text{HALT}_{TM}$ were recognizable, then $\neg A_{TM}$ would be recognizable...
Theorem: $\text{HALT}_{TM} \leq_m A_{TM}$

Proof: Define the computable function

$f(M, w) := (M', w)$ where

“$M'(w)$ accepts if $M(w)$ halts else loop forever” (how?)

Observe $(M, w) \in \text{HALT}_{TM} \iff (M', w) \in A_{TM}$
Corollary: $\text{HALT}_{TM} \equiv_m A_{TM}$

I can give you the magical power to either compute the halting problem, or the acceptance problem. Which do you want?

Surprise me
The Emptiness Problem

EMPTY\textsubscript{DFA} = \{ M \mid M \text{ is a DFA such that } L(M) = \emptyset \}

Given a DFA, does it reject every input?

Theorem: EMPTY\textsubscript{DFA} is decidable

Why?

EMPTY\textsubscript{NFA} = \{ M \mid M \text{ is a NFA such that } L(M) = \emptyset \}

EMPTY\textsubscript{REX} = \{ R \mid R \text{ is a regexp such that } L(R) = \emptyset \}
The Emptiness Problem for TMs

\[ \text{EMPTY}_{TM} = \{ M \mid M \text{ is a TM such that } L(M) = \emptyset \} \]

Given a program, does it reject every input?

Theorem: \( \text{EMPTY}_{TM} \) is not recognizable

Proof: Show that \( \neg A_{TM} \leq_m \text{EMPTY}_{TM} \)

\[ f(M, w) := M' \text{ where } \]

\[ \text{"}M'(x) := M(x) \text{ if } (x = w), \text{ else reject" } (\text{how?}) \]

\[
\begin{align*}
M, w \in A_{TM} & \iff L(M') \neq \emptyset \\
& \iff M' \notin \text{EMPTY}_{TM} \\
& \iff f(M, w) \notin \text{EMPTY}_{TM}
\end{align*}
\]
The Regularity Problem for Turing Machines

\[ \text{REGULAR}_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \} \]

Given a program, is it equivalent to some DFA?

Theorem: \( \text{REGULAR}_{TM} \) is not recognizable

Proof: Show that \( \neg A_{TM} \leq_m \text{REGULAR}_{TM} \)

\[ f(M, w) := M' : \text{where } M' \text{ is a TM such that} \]

\[ "M'(x) := M(w) \text{ if } (x = 0^n1^n) \text{ else reject" (how?)} \]

\( (M, w) \in A_{TM} \Rightarrow f(M, w) = M' \text{ such that } M' \text{ accepts } \{0^n1^n\} \)

\( (M, w) \notin A_{TM} \Rightarrow f(M, w) = M' \text{ such that } M' \text{ accepts nothing} \)

\( (M, w) \notin A_{TM} \iff f(M, w) \in \text{REGULAR}_{TM} \)
The Equivalence Problem

\[ \text{EQ}_TM = \{ (M, N) \mid M, N \text{ are TMs and } L(M) = L(N) \} \]

Do two programs compute the same function?

Theorem: \( \text{EQ}_TM \) is unrecognizable

Proof: Reduce \( \text{EMPTY}_TM \) to \( \text{EQ}_TM \)

Let \( M_\emptyset \) be a “dummy” TM with no path from start state to accept state

Define \( f(M) := (M, M_\emptyset) \)

\[ M \in \text{EMPTY}_TM \quad \iff \quad L(M) = L(M_\emptyset) = \emptyset \]
\[ \iff \quad (M', M_\emptyset) \in \text{EQ}_TM \]
Post’s Correspondence Problem

Given a collection of domino types, can we build up a match?

\[
\begin{array}{cccc}
  \text{ba} & \text{a} & \text{b} & \text{b} \\
  \underline{a} & \underline{ab} & \underline{bcb} & \underline{a} \\
\end{array}
\]

PCP = \{ P \mid P \text{ is a set of dominos with a match}\}

Theorem: PCP is **undecidable**!
Parting thoughts:

Reductions for impossibilities.

Analyzing programs is really hard.