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

$A_{TM}$ is undecidable: (proof by contradiction)

Suppose $H$ is a machine that decides $A_{TM}$

$$H( (M, w) ) = \begin{cases} 
\text{Accept} & \text{if } M \text{ accepts } w \\
\text{Reject} & \text{if } M \text{ does not accept } w
\end{cases}$$

Define a new machine $D$ as follows:

$D(M)$: Run $H$ on $(M, M)$ and output the opposite of $H$

$$D(D) = \begin{cases} 
\text{Reject} & \text{if } D \text{ accepts } D \\
\text{Accept} & \text{if } D \text{ does not accept } D
\end{cases}$$
The table of outputs of $H(x,y)$

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>...</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td></td>
<td>accept</td>
</tr>
<tr>
<td>$M_2$</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td></td>
<td>reject</td>
</tr>
<tr>
<td>$M_3$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td></td>
<td>accept</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td></td>
<td>accept</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td>?</td>
</tr>
</tbody>
</table>
### The outputs of $D(x)$

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>...</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>

$D(x)$ outputs the opposite of $H(x, x)$

$D(D)$ outputs the opposite of $H(D, D) = D(D)$
Let $H$ be a machine that recognizes $A_{TM}$

$H((M,w)) = \begin{cases} 
    \text{Accept} & \text{if } M \text{ accepts } w \\
    \text{Reject or loops} & \text{if } M \text{ does not accept } w 
\end{cases}$

Define a new machine $D_H$ as follows:

$D_H(M): \text{ Run } H \text{ on } (M,M) \text{ until the simulation halts }$

Output the opposite answer
We have an instance \((D_H, D_H)\) which is not in \(\text{A}_{TM}\) but \(H\) fails to tell us that!

\(H(D_H, D_H)\) runs forever.

\(D_H\) must loop on \(D_H\)

\(D_H\) accepts

\(D_H\) rejects

Loops if \(D_H\) loops (i.e. if \(H(D_H, D_H)\) loops)

Note: There is no contradiction here!

There is no contradiction here!
That is:

Given the code of any machine $H$ that recognizes $A_{TM}$ we can effectively construct an instance $(D_H, D_H)$, where:

1. $(D_H, D_H)$ does not belong to $A_{TM}$
2. $H$ runs forever on the input $(D_H, D_H)$

So $H$ cannot decide $A_{TM}$

Given any program that recognizes the Acceptance Problem, we can efficiently construct an input where the program hangs!
Theorem: \( A_{TM} \) is recognizable but NOT decidable

Corollary: \( \neg A_{TM} \) is not recognizable!

Proof: Suppose \( \neg A_{TM} \) is recognizable.
Then \( \neg A_{TM} \) and \( A_{TM} \) are both recognizable...
But that would mean they’re both decidable!
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 \( A_{TM} \)

\( M'(M,w) \): 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 on $w$:

(M,w)

(M,w)

(M,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 $f : \Sigma^* \rightarrow \Sigma^*$ 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 $w \in A \Rightarrow M'$ accepts $w$
- $w \notin A \Rightarrow M'$ 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}”$$

We have $(M, w) \in A_{TM} \iff (M', w) \in HALT_{TM}$
Theorem: \( A_{TM} \leq_m HALT_{TM} \)

Corollary: \( \neg A_{TM} \leq_m \neg HALT_{TM} \)

Proof?

Corollary: \( \neg HALT_{TM} \) is unrecognizable!

Proof: If \( \neg HALT_{TM} \) 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

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

Given a DFA, does it reject every input?

Theorem: \text{EMPTY}_{\text{DFA}} is decidable

Why?

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

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

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

Given a program, does it reject every input?

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

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

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

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

\[ M, w \in A_{TM} \iff L(M') \neq \emptyset \]

\[ \iff M' \notin \text{EMPTY}_{TM} \]

\[ \iff f(M, w) \notin \text{EMPTY}_{TM} \]
The Regularity Problem for Turing Machines

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

Given a program, is it equivalent to some DFA?

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

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

\( f(M, w) := M' \): where \( M' \) is a TM such that

“\( M'(x) := M(w) \) if \( x = 0^n1^n \) else reject” (how?)

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

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

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

$$EQ_{TM} = \{(M, N) \mid M, N \text{ are TMs and } L(M) = L(N)\}$$

*Do two programs compute the same function?*

**Theorem:** $EQ_{TM}$ is unrecognizable

**Proof:** Reduce $EMPTY_{TM}$ to $EQ_{TM}$

Let $M_{\emptyset}$ be a “dummy” TM with no path from start state to accept state

Define $f(M) := (M, M_{\emptyset})$

$$M \in EMPTY_{TM} \iff L(M) = L(M_{\emptyset}) = \emptyset \iff (M', M_{\emptyset}) \in EQ_{TM}$$
Moral:
Analyzing Programs is Really, Really Hard.
Post’s Correspondence Problem

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

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

Theorem: PCP is undecidable!