Undecidability

“This sentence is false.”
Fundamental Realization of the Theory of Computing

“There is a specific problem that is algorithmically unsolvable.”

Problems are not just theoretical, but practical.

E.g., **Software Verification**. Given a computer program, and a precise specification, there is no algorithm that will prove that the program performs to the specification.
Theorem: \( \mathcal{A}_{\text{TM}} \) is undecidable.

\( \mathcal{A}_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \)

Clarification: \( \mathcal{A}_{\text{TM}} \) is Turing-recognizable.

Realization: Recognizers are more powerful than Deciders.

\( U = \) “On input \( \langle M, w \rangle \), where \( M \) is a TM and \( w \) is a string:
1. Simulate \( M \) on input \( w \).
2. If \( M \) ever enters its accept state, accept; if \( M \) ever enters its reject state, reject.”

This is a Recognizer. Why isn’t it a Decider?

This example is called the Universal Turing Machine, because it can simulate any other Turing Machine from its description.

Proof of undecidability requires diagonalization, discovered by Georg Cantor in 1873.

Diagonalization is a process used in comparing the sizes of infinite sets.

Idea: Two finite sets are considered to be the same size if each element of one set can be paired with a distinct element of the other set. This idea is extended to infinite sets.
Diagonalization

Assume that we have sets $A$ and $B$ and a function $f$ from $A$ to $B$. Say that $f$ is **one-to-one** if it never maps two different elements to the same place—that is, if $f(a) \neq f(b)$ whenever $a \neq b$.

Say that $f$ is **onto** if it hits every element of $B$—that is, if for every $b \in B$ there is an $a \in A$ such that $f(a) = b$.

Say that $A$ and $B$ are the **same size** if there is a one-to-one, onto function $f: A \rightarrow B$. A function that is both one-to-one and onto is called a **correspondence**.

In a correspondence, every element of $A$ maps to a unique element of $B$ and each element of $B$ has a unique element of $A$ mapping to it.

A correspondence is simply a way of **pairing** the elements of $A$ with the elements of $B$.

**Alternative terminologies:**
- **Injective:** one-to-one
- **Surjective:** onto
- **Bijective:** one-to-one and onto
Countable Example

\( \mathcal{N} = \{1, 2, 3, \ldots\} \), the set of natural numbers

\( \mathcal{E} = \{2, 4, 6, \ldots\} \), the set of even natural numbers

The correspondence \( f \) mapping \( \mathcal{N} \) to \( \mathcal{E} \) is \( f(n) = 2n \)

\( \mathcal{N} \) and \( \mathcal{E} \) have the same size.

Intuitively, \( \mathcal{E} \) seems smaller than \( \mathcal{N} \) because \( \mathcal{E} \) is a proper subset of \( \mathcal{N} \).

Pairing each member of \( \mathcal{N} \) with its own member of \( \mathcal{E} \) is possible, so the two sets are the same size (according to Cantor’s criteria).

Definition: A set \( A \) is countable if either it is finite or it has the same size as \( \mathcal{N} \).
Countable Example 2

\[ Q = \{m/n \mid m, n \in \mathcal{N}\} \]

\( Q \) is the set of positive rational numbers.

Although \( Q \) seems larger than \( \mathcal{N} \), they are the same size.

Must give a correspondence which is a one-to-one mapping (no repetition) between the sets.

Build an infinite matrix containing all positive rational numbers, and then turn this into a list (without repetitions) by listing numbers in the diagonals.
Theorem: \( \mathcal{R} \) is Uncountable

For some infinite sets, no correspondence with \( \mathcal{N} \) exists. These sets are **uncountable**.

\( \mathcal{R} \) is the set of real numbers.

**Idea: Proof by contradiction.** Suppose correspondence \( f \) between \( \mathcal{N} \) and \( \mathcal{R} \) exists. We will show \( f \) failing by finding an \( x \) in \( \mathcal{R} \) that is not paired with anything in \( \mathcal{N} \), hence a contradiction.

Hypothetical \( f \) given, showing possible pairings between \( n \) and \( f(n) \).

Create \( x \) as a value between 0 and 1, where the \( n \)th decimal place is different from the \( n \)th decimal place of the number \( f(n) \), ensuring that \( x \) is **unique**.

**Contradiction** is shown, and therefore \( \mathcal{R} \) must be uncountable.

This is **Diagonalization**!

<table>
<thead>
<tr>
<th>( n )</th>
<th>( f(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.14159…</td>
</tr>
<tr>
<td>2</td>
<td>55.55555…</td>
</tr>
<tr>
<td>3</td>
<td>0.12345…</td>
</tr>
<tr>
<td>4</td>
<td>0.50000…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

\( x = 0.4641… \)
Corollary: Some languages are not Turing-recognizable

The set of all strings $\Sigma^*$ is **countable** for any $\Sigma$.

- Finitely many strings for each length.
- We can enumerate all strings of length 0, length 1, length 2, etc.

The set of all Turing machines is **countable** because each TM $M$ has a string encoding $\langle M \rangle$. Simply omit the strings that are not legal encodings of TMs to obtain a list of all TMs.

An **infinite binary sequence** is an unending sequence of 0s and 1s.

Let $\mathcal{B}$ be the set of all infinite binary sequences.

$\mathcal{B}$ is **uncountable**, using diagonalization proof.

*This is part 1. The fact that $\mathcal{B}$ is uncountable will be important in showing that the set of all languages is uncountable (part 2).*
Corollary: Some languages are not Turing-recognizable (cont.)

Let \( \mathcal{L} \) be the set of all languages over alphabet \( \Sigma \).

- Intent is to show that \( \mathcal{L} \) is uncountable by giving a correspondence with \( \mathcal{B} \).

Let \( \Sigma^* = \{ s_1, s_2, s_3, \ldots \} \).

Each language \( A \in \mathcal{L} \) has a unique sequence in \( \mathcal{B} \).

This will serve as a characteristic sequence of \( A \).

Creating the characteristic sequence: the \( i \)th bit of that sequence is a 1 if \( s_i \in A \), and is a 0 if \( s_i \notin A \).

The function \( f: \mathcal{L} \to \mathcal{B} \), where \( f(A) \) equals the characteristic sequence of \( A \), is one-to-one and onto, and hence a correspondence.

Therefore, as \( \mathcal{B} \) is uncountable, \( \mathcal{L} \) is uncountable as well.

<table>
<thead>
<tr>
<th>( \Sigma^* )</th>
<th>( \varepsilon ), 0, 1, 00, 01, 10, 11, 000, 001, \ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( 0, 00, 01, \quad 000, 001, \ldots )</td>
</tr>
<tr>
<td>( \chi_A )</td>
<td>0 1 0 1 1 0 0 1 1 \ldots</td>
</tr>
</tbody>
</table>
Conclusion?

No, the lecture isn’t finished yet. We still haven’t proved that $A_{TM}$ is undecidable!

Set of all Turing Machines is **countable**.

Set of all Languages is **uncountable**.

There **must** be some languages not recognized by any Turing Machine.
Theorem: $A_{TM}$ is undecidable.

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$

Proof by contradiction. Assume $A_{TM}$ is decidable and obtain a contradiction.

Suppose $H$ is a decider for $A_{TM}$. On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is a string, $H$ halts and accepts if $M$ accepts $w$. Furthermore, $H$ halts and rejects if $M$ fails to accept $w$.

New TM $D$ with $H$ as a subroutine:

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

1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.
2. Output the opposite of what $H$ outputs. That is, if $H$ accepts, reject; and if $H$ rejects, accept."

Notice the contradiction when the input to $D$ is $\langle D \rangle$ itself:

$H(\langle M, w \rangle) = \begin{cases} accept & \text{if } M \text{ accepts } w \\ reject & \text{if } M \text{ does not accept } w. \end{cases}$

$D(\langle D \rangle) = \begin{cases} accept & \text{if } D \text{ does not accept } \langle D \rangle \\ reject & \text{if } D \text{ accepts } \langle D \rangle. \end{cases}$
Understanding the Proof

1. \( H \) accepts \( \langle M, w \rangle \) exactly when \( M \) accepts \( w \).
   a. This is straightforward. We assume that an \( H \) exists that can decide this.
   b. Realize that \( M \) represents any TM, and \( w \) represents any string.

2. \( D \) rejects \( \langle M \rangle \) exactly when \( M \) accepts \( \langle M \rangle \).
   a. What happened to \( w \)?
   b. \( w \) is just a string. So is \( \langle M \rangle \). All that we are doing, in essence, is defining which string we are feeding into the machine.

3. \( D \) rejects \( \langle D \rangle \) exactly when \( D \) accepts \( \langle D \rangle \).
   a. This is the contradiction.
Understanding the Proof--Where is Diagonalization?

Consider the following table, in which TMs are the rows, inputs are the columns, and the intersection is *accept* if the TM accepts the input, and *blank* if the machine rejects or loops.

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

$H$ fills in the blanks with a *reject*.

$D$ is a TM, and must therefore appear on the matrix.

Notice that $D$’s entry is always the **opposite** of what appears on the **diagonal**.

What should appear at the question mark?
A Turing-Unrecognizable Language.

We have already shown that $A_{TM}$ is recognizable.

We have shown that the set of all TMs is countable.

We have shown that the set of all languages is uncountable.

Therefore, there must be a language that is not Turing-recognizable.

Unfortunately, there is still one thing that we must do before we can demonstrate a Turing-unrecognizable language.

We will now show that if a language and its complement are Turing-recognizable, then the language is decidable.

We say a language is co-Turing-recognizable if it is the complement of a Turing-recognizable language.
Theorem

A language is decidable iff it is Turing-recognizable and co-Turing recognizable.

Must show both directions.

First, if \( A \) is decidable, it is obvious that \( A \) and its complement are both Turing-recognizable, and that the complement of a decidable language is also decidable.

(How? Switch the accept and reject states.)

For the other direction, \( M_1 \) is the recognizer for \( A \), and \( M_2 \) is the recognizer for the complement.

\[
M = \text{“On input } w: \\
1. \text{ Run both } M_1 \text{ and } M_2 \text{ on input } w \text{ in parallel.} \\
2. \text{ If } M_1 \text{ accepts, accept; if } M_2 \text{ accepts, reject.”}
\]

Every string \( w \) is either in \( A \) or the complement, and will thus be recognized by either \( M_1 \) or \( M_2 \).

\( M \) always halts when \( w \) is accepted by either \( M_1 \) or \( M_2 \), and is therefore a decider. Thus \( A \) is decidable.
\( \overline{A_{TM}} \) is not Turing-recognizable

It’s obvious, right?

If the complement of \( A_{TM} \) were also Turing-recognizable, then \( A_{TM} \) would be decidable.

We have previously shown that \( A_{TM} \) is not decidable, thus it’s complement cannot be Turing-recognizable.
Overview

What lies outside the circles?