Diagonalization Method

To prove $\text{Undecidability}$, we will use the diagonalization method.

Overview
Definition: A set $A$ is countable if either it is finite or it has the same size as $\mathbb{N}$.

- Is the set of positive rational numbers countable?
- $\mathbb{Q} = \{ \frac{m}{n} \mid m, n \in \mathbb{N} \}$
- Can we set up a correspondence between $\mathbb{N}$ and $\mathbb{Q}$?

Set up a correspondence between $\mathbb{N}$ and $\mathbb{Q}$:

$\mathbb{Q}$: Set up a correspondence between $\mathbb{N}$ and $\mathbb{Q}$:

Can we set up a correspondence between $\mathbb{N}$ and $\mathbb{Q}$?

Correspondence: A set $A$ is countable if either it is finite or it has the same size as $\mathbb{N}$.

Definition: A set $A$ is countable if either it is finite or it has the same size as $\mathbb{N}$.

Definition: Assume we have sets $A$ and $B$ and a function $f$ from $A$ to $B$. We say that $f$ is one-to-one if $f$ is a correspondence between $A$ and $B$.

Definition: We say that $f$ is a correspondence if $f$ is one-to-one and onto.

Definition: We say that $A$ and $B$ are the same size if there is a correspondence between $A$ and $B$.

Definition: We say that $A$ and $B$ are the same size if there is a correspondence between $A$ and $B$. We say that $f$ is one-to-one if $f$ is a correspondence between $A$ and $B$.

Definition: We say that $f$ is one-to-one if $f$ is a correspondence between $A$ and $B$.

Definition: We say that $f$ is one-to-one if $f$ is a correspondence between $A$ and $B$.

Definition: We say that $f$ is one-to-one if $f$ is a correspondence between $A$ and $B$.

Definition: We say that $f$ is one-to-one if $f$ is a correspondence between $A$ and $B$.
Proof by contradiction

Set of Real Numbers is Uncountable

Definition: A real number is one that has a decimal representation.

Definition: A set of numbers is uncountable if there is no correspondence between N and R.

Definition: An infinite set that does not have a correspondence.
How many possible languages are there?

Let $L$ be the set of languages over $\Sigma$ ($L = P(\Sigma^*)$)

- How many languages are in $L$? i.e. How many subsets are in $P(\Sigma^*)$?

If $|X| = n$,

$|P(X)| = 2^n$

- What if $|X|$ is infinite, but countable?

Set of all infinite binary sequences $\mathcal{B}$ is uncountable

- Can use the diagonalization method used to show $\mathbb{R}$ is uncountable

If $L$ be the set of languages over $\Sigma$,

• $b \in \mathcal{B}$ indicates which strings of $\Sigma^*$ to include in a $A \in L$

So, uncountable number of languages

Some languages are not Turing-recognizable

- The set of all Turing machines is countable:
  - Each has an encoding as a string
  - Subset of a countable number is a countable number

- Set of all strings is countable:
  - Only finitely many strings of each length
  - Write down all strings of length 0, length 1, length 2, etc.

- Some languages are not Turing-recognizable:
  - Each has an encoding as a string
  - Subset of a countable number is a countable number

- Each has an encoding as a string
  - Subset of a countable number is a countable number
Overview

• Diagonalization

Undecidability

• Set of Turing machines is countable, set of languages are not countable, so must be some languages that a Turing machine cannot recognize

• This is pretty powerful - There are some languages that we cannot build a Turing machine that can accept (or equivalently recognize)

Another way to think about it (not in textbook)

- We haven't said what any of those languages are, but we know they exist.

- There are some languages that we cannot build a Turing machine that can accept (or equivalently recognize)

Set of Turing machines is countable, set of languages are not countable
Halting Problem is Undecidable

Proof by contradiction

We use $H$ as a subroutine

Construct a new TM that takes $\langle M \rangle$ as input. When $M$ accepts $w$:

- Because $H$ decides whether $M$ accepts
- Must accept if $M$ does not accept
- Must reject if $M$ accepts
- Assume $H$ is decidable.

Halting Problem is Undecidable

Then we will give a language that is not Turing recognizable

Next we will show that the Halting problem is not decidable

Showed that Halting problem is Turing-recognizable

- Showed that there are some languages that are not Turing-recognizable
- Showed how to use the diagonalization method with real numbers

Introduction
Where is the Diagonalization?

- Entry \( i, j \) is \( M_i \) on \( \langle M_j \rangle \)

- Running \( H \) that simulates \( M_1 \) \( M_2 \) \( M_3 \) \( M_4 \) ...

Where all TMs as rows (countable number), all computations on TMs as columns.

- Entry \( i, j \) is \( H \) on \( \langle M_j \rangle, \langle M_j \rangle \rangle \)

Thus neither TM \( D \) nor TM \( H \) can exist.

\[
\langle D \rangle \begin{cases} 
\text{reject if } D \text{ accepts } \langle D \rangle \\ 
\text{accept if } D \text{ does not accept } \langle D \rangle 
\end{cases} = \langle D \rangle
\]

\[
\langle IV \rangle \begin{cases} 
\text{reject if } IV \text{ accepts } \langle IV \rangle \\ 
\text{accept if } IV \text{ does not accept } \langle IV \rangle 
\end{cases} = \langle IV \rangle
\]

No matter what \( D \) does, it is forced to the opposite, which is obviously a contradiction.

- \( D \) \( IV \) \( D \text{ accepts } \langle IV \rangle \) \( D \text{ does not accept } \langle IV \rangle \)
Definition: 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.

Continued...
Proof

Let $A$ be Turing recognizable and co-Turing recognizable

Construct a decider for $A$ as follows:

1. Run both $M_1$ and $M_2$ on input $w$ in parallel.
2. If $M_1$ accepts, accept; if $M_2$ accepts, reject.

As every string $w$ is either in $A$ or not in $A$, one of $M_1$ or $M_2$ must accept $w$.

So if $M_1$ accepts, it accepts $w$, and if $M_2$ accepts, it rejects $w$.

Hence $M$ is a decider.

Thus $A$ is decidable.

Proof

Let $A$ be decidable.

Hence there is a deterministic TM $M_1$ that decides $A$.

Construct $M_2$ that accepts any string that $M_1$ does not accept, and vice versa.

So $A$ is Turing recognizable and co-Turing recognizable.

Hence both $M_1$ recognizes $A$ and $M_2$ recognizes $A$.

Therefore, $A$ is Turing-recognizable and co-Turing-recognizable.

Conclusion: $A$ is Turing-recognizable and co-Turing-recognizable.

Proof:

Let $A$ be decidable.
A Turing-Unrecognizable Language

Corollary: \( A_{TM} \) is not Turing-recognizable

- There is no Turing machine that can tell whether a certain TM will loop forever or reject on a certain input.
- So, \( A_{TM} \) is not Turing-recognizable.

Assume \( A_{TM} \) is Turing-recognizable.

Then \( A_{TM} \) is decidable.

Contradiction.

We know that \( A_{TM} \) is Turing-recognizable.

- So, \( A_{TM} \) is not Turing-recognizable.