What Machines Cannot Do

The Universe of Languages

The Sizes of Sets

• Comparing the sizes of two finite sets is easy
• Do all infinite sets have the same size? How can we compare the relative sizes of two infinite sets?

<table>
<thead>
<tr>
<th>W</th>
<th>N</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A set is countable if either it is finite or it has the same size as \( \mathbb{N} \).
We will find $x$ in $\mathbb{R}$ that is not paired with anything in $\mathbb{N}$, which will be our contradiction.

<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>5.55555...</td>
</tr>
<tr>
<td>3</td>
<td>0.12345...</td>
</tr>
<tr>
<td>4</td>
<td>0.50000...</td>
</tr>
<tr>
<td>5</td>
<td>1.414213...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

We show that no correspondence exists between $\mathbb{N}$ and $\mathbb{R}$.

To reach a contradiction, suppose that a correspondence $f$ does exist between $\mathbb{N}$ and $\mathbb{R}$.

$\mathbb{R}$ is uncountable (proof by diagonalization)

A finite representation of a language must itself be a string over some alphabet $\Sigma$. Furthermore, different languages must have distinct representations.

How many strings can we represent over any given alphabet?

How Many is Many?

Theorem. Let $\Sigma$ be any finite alphabet containing at least one element. The set of all strings $\Sigma^*$ over $\Sigma$ is countably infinite.

How Many Languages?

Definition. Let $2^{\Sigma^*}$, known as the power set of $\Sigma^*$, be the set of all subsets of $\Sigma^*$, i.e., the set of all languages over $\Sigma$.

Theorem. The set $2^{\Sigma^*}$ is uncountable.

Proof. For each language $A \in 2^{\Sigma^*}$, create a unique infinite binary sequence.

$\Sigma^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ... \}$

$A = \{ 0, 00, 01, 000, 001, ... \}$

$f(A) = 010110011100111...$
How Many Languages?

**Definition.** Let $2^{\Sigma^*}$, known as the power set of $\Sigma^*$, be the set of all subsets of $\Sigma^*$, i.e., the set of all languages over $\Sigma$.

**Theorem.** The set $2^{\Sigma^*}$ is uncountable.

**Proof.** For each language $A \in 2^{\Sigma^*}$, create a unique infinite binary sequence.

$\Sigma^* = \{ \epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \}$

$A = \{ \epsilon, 0, 01, 10, 001, \ldots \}$

$f(A) = 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ \ldots$

Thus, we have a correspondence $f$ between $2^{\Sigma^*}$ and infinite binary sequences. Since the set of infinite binary sequences is uncountable (see homework), so is $2^{\Sigma^*}$.

The Sad Conclusion...

The Trick is to Get all the Good Ones

Algorithm = Turing Machine
Let's Try This One*

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

* By analogy with our old friends $A_{DFA}$ and $A_{CFG}$.

$A_{TM}$ is Turing-Recognizable

$U =$ "On input $\langle M, w \rangle$, where $M$ is a TM and $w$ 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."

The universal Turing machine.

The Halting Problem

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

"On input $\langle M, w \rangle$, where $M$ is a TM and $w$ a string:

1. Determine whether $M$ on input $w$ will ever halt. If not, then reject.
2. Otherwise, simulate $M$ on input $w$.
3. If $M$ enters its accept state, accept. If $M$ enters its reject state, reject."

Some People Don't Know When to Stop

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

**Proof.** Suppose TM $H$ decides $A_{TM}$. That is,

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

Define the contrary TM $D$:

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

1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.*
2. Output the opposite of what $H$ outputs.

That is,

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

* Think of a Python compiler written in Python.

Calling $D$ on Itself

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

$\tilde{A}_{TM}$ is not even Turing-recognizable

Corollary. $\tilde{A}_{TM}$ is not Turing-recognizable.

Proof. If so, then both $A_{TM}$ and $\tilde{A}_{TM}$ would be Turing-recognizable. But, then ...