What Machines Cannot Do
Will This Ever End?

The Universe of Languages

Regular

Context-free

Turing-decidable

\( a^n b^n \)

Turing-recognizable

\( D \)

\( a^n b^n c^n \)

?
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>
The Sizes of Sets

- Two sets have the same size if the elements of one set can be paired with the elements of the other set.

- A function that is both one-to-one and onto is called a **correspondence** (bijection). Two sets have the same size if there is a correspondence between them.

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

- 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 $\mathbb{R}$.

- We will find $x$ in $\mathbb{R}$ that is not paired with anything in $\mathbb{N}$, which will be our contradiction.

$$
\begin{array}{c|c}
 n & f(n) \\
 \hline
 1 & 3.14159... \\
 2 & 55.55555... \\
 3 & 0.12345... \\
 4 & 0.50000... \\
 5 & 1.414213... \\
 \ldots & \ldots \\
\end{array}
$$
Finite Representation of Languages

• 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.

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

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

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

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

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

$$f(A) = 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ \ldots$$
How Many Languages?

**Definition.** Let $\mathcal{P}(\Sigma^*)$, 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 $\mathcal{P}(\Sigma^*)$ is uncountable.

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

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

$$A = \{ \varepsilon, 01, 11, 000, 001, \ldots \}$$

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

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

The Universe of Languages

- Turing-recognizable
- Turing-decidable
- Context-free
- Regular

$\alpha^n \beta^n \gamma^n$

$a^n b^n$

$a^* b^*$

$S_1 \xrightarrow{1} S_2 \xrightarrow{0} S_1 \xrightarrow{1} S_2$
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 \( A_{CFG} \).
The universal Turing machine.

A$_{TM}$ is Turing-Recognizable

$U = \text{"On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ 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."
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.** \( A_{TM} = \{ <M, w> \mid M \text{ is a TM and } M \text{ accepts } w \} \) is undecidable.

**Proof.** Suppose TM \( H \) decides \( A_{TM} \). That is,

\[
H(<M, w>) = \begin{cases} 
\text{accept} & \text{if } M \text{ accepts } w \\
\text{reject} & \text{if } M \text{ does not accept } w
\end{cases}
\]
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.

* Think of a Python compiler written in Python.
Calling $H$ as a Subroutine

Define the contrary TM $D$:

$$D = "\text{On input } <M>, \text{ where } M \text{ is a TM:}\n$$

1. Run $H$ on input $<M, <M>>.*$
2. Output the opposite of what $H$ outputs.

That is,

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

* Think of a Python compiler written in Python.
Calling $D$ on Itself

$$D(<D>) = \begin{cases} 
accept & \text{if } D \text{ does not accept } <D> \\
reject & \text{if } D \text{ accepts } <D> 
\end{cases}$$
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 ...
Out of Bounds

The Universe of Languages

- Turing-decidable
- Context-free
- Regular
- $a^n b^n$
- $a^n b^c^n$
- $\tilde{A}_{TM}$
- $A_{TM}$