Undecidability

Last Lecture We Saw

- There are languages that are not Turing-recognizable
  - Therefore, there are more languages than there are Turing machines
  - Therefore, there must be a language that is not Turing-recognizable
  - We do not have yet an example of such language

- What about not Turing-decidable languages?

Towards a Language that is not Turing-Decidable

Definition. $A_T = \{ <M, w> | M \text{ is a TM and } M \text{ accepts } w \}$
Towards a Language that is not Turing-Decidable

**Definition.** $A_{TM} = \{ <M, w> | M \text{ is a TM and } M \text{ accepts } w \}$.

**Question 1.** Is $A_{TM}$ Turing recognizable?

No!

$A_{TM}$ is Turing-Decidable

$U = \{ <M, w> | M \text{ is a TM and } M \text{ accepts } w \}$.

**Question 1.** Is $A_{TM}$ Turing recognizable? Yes!

Universal Turing Machine receives other turing machines as input and simulates them.
Towards a Language that is not Turing-Decidable

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

Question 1. Is $A_{TM}$ Turing recognizable? Yes!

Question 2. Is $A_{TM}$ decidable?

No! Proof by contradiction.

$A_{TM}$ is not Turing-Decidable

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

Proof. Suppose there is Turing Machine $H$ that decides $A_{TM}$. That is,
\( A^{TM}_{TM} \) is not Turing-Decidable

**Theorem.** \( A^{TM}_{TM} = \{ <M, w> | M \text{ is a TM and } M \text{ accepts } w \} \) is undecidable.

**Proof.** Suppose there is Turing Machine \( H \) that decides \( A^{TM}_{TM} \). 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 a Turing Machine \( D \):

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

1. Run \( H \) on input \( <M, <M>> \).
2. Output the opposite of what \( H \) outputs.

That is,

\[
D(<M>) =
\begin{cases} 
  \text{Accept} & \text{if } H(<M, <M>>) \text{ rejects} \\
  \text{Reject} & \text{if } H(<M, <M>) \text{ accepts}
\end{cases}
\]
Calling H as a Subroutine

Define a Turing Machine $D$:

$D$ = “On input $<M>$, where $M$ is a TM:
1. Run $H$ on input $<M, <M>>$.
2. Output the opposite of what $H$ outputs.

That is,

$$D(<M>) = \begin{cases} 
\text{Accept} & \text{if } H(<M, <M>>) \text{ rejects} \\
\text{Reject} & \text{if } H(<M, <M>>) \text{ accepts} 
\end{cases}$$

Remember that, by assumption, $H$ decides $A_{\text{TM}}$.

The Diagonalization Trick

Entry $i, j$ is the value of $H$ on input $\langle M_i, \langle M_j \rangle \rangle$. 

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

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

Entry $i, j$ is the value of $H$ on input $\langle M_i, M_j \rangle$.
\( A_{TM} \) is not Turing-Decidable

**Theorem.** \( A_{TM} = \{ <M, w> | M \text{ is a TM and } M \text{ accepts } w \} \) is undecidable.

**Proof.**

- We assumed there was a Turing Machine \( H \) that decides \( A_{TM} \)
- Created a Turing Machine \( D \) that uses \( H \)
- Reached a contradiction
- We can conclude Turing Machine \( H \) cannot exist.
- Therefore, \( A_{TM} \) is undecidable.

![Completing the Picture](image-url)