Undecidability
Last Lecture We Saw

- There are languages that are not Turing-recognizable
  - 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
Last Lecture We Saw

- There are languages that are not Turing-recognizable
  - 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.** \( A_{TM} = \{ <M, w> \mid M \text{ is a TM and } M \text{ accepts } w \} \).
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?
$A_{\text{TM}}$ is Turing-Recognizable

$U =$ “On input $<M, w>$, where $M$ is a TM and $w$ a string:

1. Simulate $M$ on input $w$.
2. If $M$ ever enters its accept state, accept.
3. If $M$ ever enters its reject state, reject.”
\[ A_{TM} \] is Turing-Recognizable

\[ U = \text{“On input } <M, w>, \text{ where } M \text{ is a TM and } w \text{ a string:}
\]
1. Simulate \( M \) on input \( w \).
2. If \( M \) ever enters its accept state, accept.
3. If \( M \) ever enters its reject state, reject.”

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

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

*Question 1.* Is $A_{TM}$ Turing recognizable? **Yes!**
Towards a Language that is not Turing-Decidable

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

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

**Question 2.** Is $A_{\text{TM}}$ decidable?
Towards a Language that is not Turing-Decidable

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

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

**Proof.**
$A_{TM}$ is not Turing-Decidable

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

**Proof.** Suppose there is Turing Machine $H$ that decides $A_{TM}$. That is,
A_{TM} is not Turing-Decidable

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

**Proof.** Suppose there is Turing Machine \( H \) that decides \( A_{TM} \). 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 $D$:

\[ D = \text{“On input } <M>, \text{ where } M \text{ is a TM:} \]
\[ 1. \text{ Run } H \text{ on input } <M, <M>>. \]
\[ 2. \text{ Output the opposite of what } H \text{ outputs.} \]
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.

Does $M$ accept on input $<M>$?
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 = \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,

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

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

\[ D(<M>) = \begin{cases} 
     \text{Accept} & \text{if } M \text{ does not accept } <M> \\
     \text{Reject} & \text{if } H(<M, <M>>) \text{ accepts} 
\end{cases} \]

Remember that, by assumption, $H$ decides $A_{TM}$.
Calling H as a Subroutine

Define a Turing Machine $D$:

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

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

That is,

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

<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>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td></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></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

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

<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>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td></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></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

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

\[
D(<D>) = \begin{cases} 
  \text{Accept} & \text{if } D \text{ does not accept } <D> \\
  \text{Reject} & \text{if } D \text{ accepts } <D>
\end{cases}
\]
The Diagonalization Trick

<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>$\cdots$</th>
<th>$\langle D \rangle$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td></td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></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>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td></td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
</tr>
</tbody>
</table>
The Diagonalization Trick

<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>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>$\cdots$</td>
<td>accept</td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>$_________$</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
</tbody>
</table>

D must accepts when it rejects, and vice-versa. Contradiction!
$A_{TM}$ is not Turing-Decidable

**Theorem.** $A_{TM} = \{ <M, w> \mid M \text{ is a TM and } M \text{ accepts } w \}$ 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 exists.
- Therefore, $A_{TM}$ is undecidable.
Completing the Picture

All Languages

Recursively-Enumerable Languages
Recognized by Turing Machines

Decidable Languages
Decidable by Turing Machine
$0^n1^n2^n$, $ww$

Context-free Languages
Push-down Automaton
$0^n1^n$, $ww^R$

Regular Languages
Finite Automaton
$1^*0^*$, $(0 \cup 1)^*0$