Halting Problem
Halting Program

Halting Program

- $H(M, I)$ -- prints YES, if $M$ HALTS on input $I$
- $H(M, I)$ -- prints NO, if $M$ LOOPS FOREVER on input $I$
- Note: $H(M, I)$ halts for all input $M, I$.

Negation of Halting Program

- $K(P)$
  -- Run $H(P, P)$
  -- If Output is YES, then LOOP FOREVER
  -- If Output is NO, then HALT
Halting Problem

Paradox

• $K(K)$
  -- Run $H(K,K)$
  -- If Output is YES, then LOOP FOREVER
  -- If Output is NO, then HALT

• $H(K,K)$
  – If Output is YES, then $K(K)$ LOOPS FOREVER
  -- If Output is NO, then $K(K)$ HALTS

Therefore $H$ FAILS to solve the Halting Problem!
**Diagonalization**

\[
\begin{array}{cccccc}
TM/I & I_1 & I_2 & \cdots & I_n & \cdots \\
M_1 & H & L & \cdots & L & \cdots \\
M_2 & L & L & \cdots & L & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
M_n & H & L & \cdots & H & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\end{array}
\]

**Diagonalization Argument**

- \( M \) Halts on Input \( I_k \) \( \iff \) \( M_k \) Loops on Input \( I_k \)
- \( M \) Differs from Each Machine in the List.
- Therefore Any List of Turing Machine is Incomplete -- Contradiction
<table>
<thead>
<tr>
<th>$TM/I$</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$\cdots$</th>
<th>$I_n$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>$H$</td>
<td>$L$</td>
<td>$\cdots$</td>
<td>$L$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>$L$</td>
<td>$L$</td>
<td>$\cdots$</td>
<td>$L$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$M_n$</td>
<td>$H$</td>
<td>$L$</td>
<td>$\cdots$</td>
<td>$H$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

**Chatter**

- Each Input can be Regarded as a Description of a Turing Machine.
- Each Output Tape can be Regarded as a Statement about a Turing Machine.
**Decidable and Semi-Decidable Languages**

*Decidable*

A language $L$ is **Decidable** if for every string $w$, there is a Turing Machine $M$ that correctly decides whether $w \in L$.

- $M$ Halts and Accepts if $w \in L$
- $M$ Halts and Rejects if $w \notin L$

*Semi-Decidable*

A language $L$ is **Semi-Decidable** if for every string $w$, there is a Turing Machine $M$ that semi-decide whether $w \in L$.

- $M$ Halts and Accepts if $w \in L$
- $M$ Either Loops or Rejects if $w \notin L$
Theorems

*Theorem 1*

There exist Semi-Decidable languages that are not Decidable.

*Theorem 2*

The Language
\[
L = \{ < M, I > | M \text{ Halts on } I \}
\]

is Semi-Decidable, but not Decidable.
Theorem 2

The Language

\[ L = \{ < M, I > | M \text{ Halts on } I \} \]

is Semi-Decidable, but not Decidable.

Proof: Let \( M_{SD}(< M, I >) \) be the following program:

• Run Machine \( M \) on Input \( I \).
• If \( M \) Halts on \( L \), then Accept \( < M, I > \).

\( M_{SD} \) Semi-Decides \( L \) because:

• \( < M, I > \in L \iff M \text{ Halts on } I \iff M_{SD} \text{ Accepts } < M, I > \).
• There are pairs \( < M, I > \) for which \( M \) never Halts on \( I \), so \( M_{SD}(< M, I >) \) does not Halt if \( < M, I > \not\in L \).

\( L \) is not Decidable because the Halting Problem is Undecidable.
Halting Problem and Semi-Decidability

Observation

If the Halting Problem were Decidable, then every Semi-Decidable Language would be Decidable.

Proof: Suppose that $O$ were a Machine that Decides the Halting Problem. Let $M_{SD}(<M,I>)$ Semi-Decide $L$.

Let $M_D(<M,I>)$ be the following program:

• If $O$ say that $M_{SD}(<M,I>)$ does not Halt, then Reject $<M,I>$.

• Otherwise Run $M_{SD}(<M,I>)$:
  -- If $M_{SD}(<M,I>)$ Accepts, then Accept $<M,I>$.
  -- If $M_{SD}(<M,I>)$ Rejects, then Reject $<M,I>$. 