A_{TM} is Undecidable

• A_{TM} = \{<M,w>: M is a TM that accepts w\}

• Theorem: A_{TM} is Undecidable

• Proof: Suppose there exists a TM H that decides A_{TM}. Then, for any input <M,w>, H accepts if M accepts w and rejects otherwise.

• Derive contradiction using diagonalization
Diagonalization!

### Table

<table>
<thead>
<tr>
<th>H</th>
<th>&lt;M1&gt;</th>
<th>&lt;M2&gt;</th>
<th>&lt;M3&gt;</th>
<th>&lt;M4&gt;</th>
<th>&lt;M5&gt;</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>acc</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td>rej</td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>acc</td>
<td>acc</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>rej</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td>rej</td>
<td></td>
</tr>
<tr>
<td>M4</td>
<td>rej</td>
<td>rej</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

H accepts \{<M, <M>> : M accepts <M>\}
Diagonalization!

<table>
<thead>
<tr>
<th></th>
<th>&lt;M1&gt;</th>
<th>&lt;M2&gt;</th>
<th>&lt;M3&gt;</th>
<th>&lt;M4&gt;</th>
<th>&lt;M5&gt;</th>
<th></th>
<th>&lt;D&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>acc</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td>rej</td>
<td></td>
<td>rej</td>
</tr>
<tr>
<td>M2</td>
<td>acc</td>
<td>acc</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td></td>
<td>rej</td>
</tr>
<tr>
<td>M3</td>
<td>rej</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td>rej</td>
<td></td>
<td>rej</td>
</tr>
<tr>
<td>M4</td>
<td>rej</td>
<td>rej</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td></td>
<td>rej</td>
</tr>
<tr>
<td>M5</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td></td>
<td>acc</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>rej</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td>rej</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

H accepts \{<M, <M>> : M accepts <M>\}
Diagonalization: Let \(D\) be a TM that negates diagonal
\(D\) is a TM: Call \(H\) on <M, <M>> and negate, so on list
But \(D\) is different, by construction, from all \(M_i\). ↳
Theorem: $A_{TM}$ is Undecidable. ($A_{TM} = \{<M,w>: M \text{ is a TM that accepts } w\}$)

Proof: Suppose there exists a TM $H$ that decides $A_{TM}$. Then, for any input $<M,w>$, $H$ accepts if $M$ accepts $w$ and rejects otherwise.

Consider a TM $D$ that takes an input $<M>$, the description of $M$, and takes the following steps.

- Run $H$ on $<M,<M>$
- If $H$ accepts, reject
- If $H$ rejects, accept

Since $H$ is a decider, $D$ is also a decider.

$D$ on $<D> = \text{accept}$

iff \( \text{def. } D \) \(H <D, <D>> = \text{reject}\)

iff \( \text{def. } H \) \(D \text{ on } <D> = \text{reject}\) (Go both directions!) \(\forall\)
Reducibility

- We showed the undecidability of $\text{HALT}_\text{TM}$ by reducing $A_{\text{TM}}$ to $\text{HALT}_\text{TM}$
- We write $A_{\text{TM}} \leq_{\text{M}} \text{HALT}_\text{TM}$
- This is read as “$A_{\text{TM}}$ is mapping reducible to $\text{HALT}_\text{TM}$”
- If $A \leq_{\text{M}} B$ that means there is a *computable* function $f: \Sigma^* \rightarrow \Sigma^*$ s.t. for all $w$
  - $w \in A$ iff $f(w) \in B$
  - $f$ is a *reduction* from $A$ to $B$
- A function is computable if some TM, on every input $w$ halts with $f(w)$ on tape
Reducibility

- Theorem: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable
  - Proof: Let $M$ be a decider for $B$ and $f$ the reduction from $A$ to $B$. Here is a decider, $N$, for $A$
    - Given $w$, compute $f(w)$
    - Run $M$ on $f(w)$, returning same output
  - Why doesn’t the other direction work?
- Corollary: If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable. Proof?
  - Our proof of undecidability of $\text{HALT}_{TM}$ was essentially based on this corollary.
- Mapping reducibility version: $f$ is defined by TM $F$: On input $<M,w>$
  - Construct $M'$: Given $x$: Run $M$ on $x$. If $M$ accepts, accept else loop
  - Output $<M',w>$
  - Note: $<M, w> \in A_{TM}$ iff $f(<M,w>) (= <M',w>) \in \text{HALT}_{TM}$
- Theorem: If $A \leq_m B$ and $B$ is R.E., then $A$ is R.E. (Same proof as above)
- Corollary: If $A \leq_m B$ and $A$ is not R.E., then $B$ is not R.E.
Rice’s Theorem

- P is undecidable if it is a language consisting of TM descriptions s.t.
  - P is nontrivial: \( P \neq \emptyset \) & P does not include all TM descriptions
  - If \( L(M_1) = L(M_2) \) then \(<M_1> \in P \iff <M_2> \in P\)
- Proof: By a reduction from \( A_{TM} \), i.e., we show \( A_{TM} \leq_M P \)
- Let \( E \) be a TM s.t. \( L(E) = \emptyset \). Assume \(<E> \notin P \) (\( A_{TM} \leq_M \neg P \) works also)
- Note: there exists TM \( T \) s.t. \(<T> \in P\)
- \( f(<M, w>) = TM \ M_w \): On input \( x \), simulate \( M \) on \( w \). If \( M \) accepts, simulate \( T \) on \( x \).
- \( f \) is a mapping reduction
  - \(<M, w> \in A_{TM} \Rightarrow L(<M_w>) = L(T) \Rightarrow <M_w> \in P\)
  - \(<M, w> \notin A_{TM} \Rightarrow L(<M_w>) = L(E) \Rightarrow <M_w> \notin P\)
- \{\(<M>: M \text{ always halts}\}, \{\(<M>: L(M) = \Sigma^*\}, \ldots \text{ all undecidable by Rice’s Theorem}\)
Halting Problem

• $\text{HALT}_{TM} = \{<M, w>: M \text{ halts on } w\}$

• Theorem: $\text{HALT}_{TM}$ is undecidable.

• Proof: We show that if $\text{HALT}_{TM}$ is decidable, then so is $A_{TM}$.

• Preview of reduction: We reduce from $A_{TM}$ to $\text{HALT}_{TM}$ ($A_{TM} \leq_M \text{HALT}_{TM}$).

• Suppose $H$ is the decider for $\text{HALT}_{TM}$. Then define a decider $A$ for $A_{TM}$ as follows. On input $<M, w>$, $A$ calls $H$ on input $<M, w>$. If $H$ accepts, then $A$ runs $M$ on $w$ and accepts if $M$ accepts $w$, rejecting otherwise. If $H$ rejects, then $A$ rejects.

• Consider $<M,w>$ in $A_{TM}$. Since $M$ accepts $w$, $M$ halts on $w$. So $H$ accepts $<M, w>$. $A$ calls $H$, which accepts, and then runs $M$ on $w$, which accepts, so $A$ accepts.

• Consider $<M,w>$ not in $A_{TM}$. If $M$ does not halt on $w$, $H$ rejects $<M, w>$, and so does $A$. Otherwise, $M$ halts on $w$ and rejects $w$. So $A$ calls $H$, which accepts $<M, w>$. $A$ then calls $M$ on $w$, which terminates in a reject state, so $A$ rejects.
$E_{TM}$ is undecidable

- $E_{TM} = \{ <M> | L(M) = \emptyset \}$ is undecidable
- Proof: Suppose it is decidable. Let $R$ be a TM deciding it.
- Define $S$, a decider for $A_{TM}$: On input $<M,w>$
  - Construct Machine $M_1$: if input $\neq w$, reject else run $M$ on $w$
    - Note: language of $M_1$ is either $\emptyset$ or $\{w\}$
  - Runs $R$ on $<M_1>$
    - If $R$ accepts, reject; if $R$ rejects, accept
- $S$ is a decider for $A_{TM}$
- Note: $S$ has to construct $M_1$: add extra states to check input $= w$
- Reduction: $f$ takes $<M,w>$ and produces $<M_1>$. $M$ accepts $w$ iff $L(M_1) \neq \emptyset$, so we showed
  - $A_{TM} \leq_M \neg E_{TM}$
  - which implies $E_{TM}$ is not decidable (decidability is not affected by complementation)
EQ\textsubscript{TM} is undecidable

- EQ\textsubscript{TM} = \{ <M, N> \mid L(M) = L(N) \} is undecidable

- Proof: E\textsubscript{TM} is just a special case where L(N) = \emptyset. So, show E\textsubscript{TM} \leq\textsubscript{M} EQ\textsubscript{TM}. Let R be a TM deciding EQ\textsubscript{TM}.

- Define S, a decider for E\textsubscript{TM}: On input <M>
  - Runs R on <M, N> where N is a TM that rejects all inputs
  - If R accepts, accept; if R rejects, reject

- S is a decider for A\textsubscript{TM}

- Reduction: f takes <M> and produces <M, N> where N is a TM that always rejects. L(M)=\emptyset iff L(M)=L(N) (where L(N) = \emptyset)
EQ_{TM} is not R.E.

- EQ_{TM} = \{ <M, N> | L(M) = L(N) \} is not R.E.

- Recall the corollary: If A \leq_M B and A is not R.E., then B is not R.E.

- But A \leq_M B iff \neg A \leq_M \neg B so to show B is not R.E. we can instead show A_{TM} \leq_M \neg B

- Plan: Show A_{TM} \leq_M \neg EQ_{TM}

- Proof: F = Given <M, w> (1) construct M_1: always reject and M_2: Run M on w (2) Output <M_1, M_2>
  
  - If M accepts w, M_2 accepts everything, so M_1, M_2 are not equivalent
  
  - If M doesn’t accept w, M_2 accepts nothing, so M_1, M_2 are equivalent
¬EQ_{TM} is not R.E.

• ¬EQ_{TM} = \{ <M, N> | L(M) \neq L(N) \} is not R.E.

• Plan: Show A_{TM} \leq_{M} EQ_{TM}

• Proof: G = Given <M, w> (1) construct M_1: always accept and M_2: Run M on w (2) Output <M_1, M_2>

  • If M accepts w, M_2 accepts everything, so M_1, M_2 are equivalent
  • If M doesn’t accept w, M_2 accepts nothing, so M_1, M_2 are not equivalent

• We showed that neither of EQ_{TM}, ¬EQ_{TM} are R.E. so EQ_{TM} is neither R.E. nor co-R.E.!
REGULAR\textsubscript{TM} is undecidable

- $\text{REGULAR}_{\text{TM}} = \{ <M> \mid L(M) \text{ is a regular language} \}$

- Plan: $A_{\text{TM}} \leq_{M} E\text{Q}_{\text{TM}}$

- Proof: Let $R$ be a TM that decides $\text{REGULAR}_{\text{TM}}$ and construct $S$, which decides $A_{\text{TM}}$ as follows

  - $S$: Given $<M,w>$
    - (1) Construct $N$: On input $x$: If $x \in 0^n1^n$, accept, otherwise run $M$ on $w$
    - (2) Run $R$ on $<N>$
    - (3) If $R$ accepts, accept, else reject.

  - If $M$ accepts $w$, $N$ accepts everything, so $N$ is regular
  - If $M$ doesn’t accept $w$, $N$ accepts $\{x \in 0^n1^n\}$ so $N$ is not regular