Section: Decidability

Computability  A function $f$ with domain $D$ is *computable* if there exists some TM $M$ such that $M$ computes $f$ for all values in its domain.

Decidability  A problem is *decidable* if there exists a TM that can answer yes or no to every statement in the domain of the problem.
The Halting Problem

Domain: set of all TMs and all strings \( w \).

Question: Given coding of M and w, does M halt on w?
Theorem The halting problem is undecidable.

Proof: (by contradiction)

• Assume there is a TM $H$ (or algorithm) that solves this problem. TM $H$ has 2 final states, $q_y$ represents yes and $q_n$ represents no.

$$H(w_M, w) = \begin{cases} 
\text{halts } q_y \text{ if } M \text{ halts on } w \\
\text{halts } q_n \text{ if } M \text{ doesn't halt on } w
\end{cases}$$

TM $H$ always halts in a final state.
$z$ any symbol in $\Gamma$
Construct TM $H'$ from $H$

$$H'(w_M, w) = \begin{cases} \begin{aligned} & \text{halts} \quad \text{if } M \text{ not halt on } w \\ & \text{not halt} \quad \text{if } M \text{ halts on } w \end{aligned} \end{cases}$$

Construct TM $\hat{H}$ from $H'$

$$\hat{H}(w_M) = \begin{cases} \begin{aligned} & \text{halts} \quad \text{if } M \text{ not halt on } w_M \\ & \text{not halt} \quad \text{if } M \text{ halts on } w_M \end{aligned} \end{cases}$$

Note that $\hat{H}$ is a TM.

There is some encoding of it, say $\hat{w}_\hat{H}$.

What happens if we run $\hat{H}$ with input $\hat{w}_\hat{H}$?

$$\hat{H}(\hat{w}_\hat{H}) = \begin{cases} \begin{aligned} & \text{halts if } \hat{H} \text{ doesn't halt on } \hat{w}_\hat{H} \\ & \text{doesn't halt if } \hat{H} \text{ halts on } \hat{w}_\hat{H} \end{aligned} \end{cases}$$

Saying $\hat{H}$ halts on $\hat{w}_\hat{H}$ if $\hat{H}$ doesn't halt on $\hat{w}_\hat{H}$: Contradiction!

$\implies$ There is no algorithm for this problem: Undecidable!
Theorem If the halting problem were decidable, then every recursively enumerable language would be recursive. Thus, the halting problem is undecidable.

- Proof: Let $L$ be an RE language over $\Sigma$.
  Let $M$ be the TM such that $L=L(M)$.
  Let $H$ be the TM that solves the halting problem.

Calculate $H(w_{M|w})$.

1. If $H$ says no, then $w$ is not in $L$.
2. If $H$ says yes, then apply $M$ to $w$. $M$ should halt and tell us if $w$ is in $L$, or not.

Thus we can determine if $w$ is in $L$ or not, so $L$ is recursive.

Thus every Rec Enum language is recursive, contradicting that we already showed $\exists$ Rec Enum
A problem A is reduced to problem B if the decidability of B follows from the decidability of A. Then if we know B is undecidable, then A must be undecidable.
State-entry problem

Given TM $M=\langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$, state $q \in Q$, and string $w \in \Sigma^*$, is state $q$ ever entered when $M$ is applied to $w$?

This is an undecidable problem!

Proof:

TM $E$ solves state-entry problem

$E'(w_M, w) = \begin{cases} 
M \text{ halts on } w & \text{if } \text{yes}\text{?} \\
M \text{ doesn’t halt on } w & \text{if } \text{no}\text{?} 
\end{cases}$
E

does M halt on W

E

enter state?

yes

no

E

E

enter state?

yes

no

halt

doesn't halt

halt

doesn't halt

halting problem is undecidable
Contradiction!
State entry problem is undecidable