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.
Construct TM $H'$ from $H$

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

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

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

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 $\hat{w}_\hat{H}$?

$\hat{H}(\hat{w}_\hat{H})$ halts if $\hat{H}$ doesn't halt on $\hat{w}_\hat{H}$

$\hat{H}$ halts on $w_M$ if $\hat{H}$ halts on $w_M$

$\Rightarrow$ problem is 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 $H(wm,w)$

- If H says no, w is not in L
- If H says yes, apply M to w, M should halt, tell us if w is in L or not

$\Rightarrow$ can determine if w is in L
$\Rightarrow$ L is recursive, or not
$\Rightarrow$ every RE language is recursive
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 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, 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

  $E'(w_M, w) = \begin{cases} 
  M \text{ halts on } w & \text{if } \text{?} \\
  M \text{ doesn't halt on } w & \text{if } \text{?}
  \end{cases}$
modify whenever \( S \) is not defined for some state \( q_i \) and symbol \( a \) add \( S(q_i, a) = (q_1a, R) \)

\( q_f \) is a new state that is the only final state.
the halting problem is undecidable