Consider the following languages over $\Sigma = \{0, 1\}$.

Let $L_e = \{ \varepsilon(M) \mid I(M) = \emptyset \}$

Let $L_{\neg e} = \{ \varepsilon(M) \mid I(M) \neq \emptyset \}$

Hence: $L_e$ is the set of all strings that encode TMs $M$ that accept the empty language.

$L_{\neg e}$ is the complement of $L_e$.

**Claim 1:** $L_{\neg e}$ is R.E.

**Proof:** Construct NTM $N$ for $L_{\neg e}$

(and then convert $N$ to an ordinary TM.)

$N$ works as follows: on input $\varepsilon(M)$

1) guess a string $w \in \Sigma^*$

2) simulate $M$ on $w$ (like a UTM)

3) accept if $M$ accepts $w$

![Diagram](attachment:image.png)

We have $\varepsilon(M) \in I(N) \iff \exists w \text{ s.t. } \langle M, w \rangle \in I(U) \iff \exists w \text{ s.t. } w \in I(M) \iff \varepsilon(M) \in L_{\neg e}$
Claim 2: \( L_e \) is non-recursively enumerable.

Proof: by reduction from \( L_m \) to \( L_e \).

Reduction \( R \) is a function computable by a halting T.M. with input:\n\[ \langle M, w \rangle \] of \( L_m \) \noutput:\n\[ E(M') \] of \( L_e \) \nend set:\n\[ \langle M, w \rangle \in L_m \iff E(M') \in L_e \]

**Description of \( M' \):**

- \( M' \) ignores completely its own input string \( x \).
- Instead, it replaces its input by the string \( \langle M, w \rangle \) and simulates \( M \) on \( w \) using a UTM.

- If \( M \) accepts \( w \), then \( M' \) accepts \( x \).
- If \( M \) never halts on \( w \) or rejects \( w \), then \( M' \) also never halts or rejects \( x \).

**Note:**
- If \( w \in \Sigma^* \), then \( \Sigma^* \) is empty.
- If \( w \notin \Sigma^* \), then \( \emptyset \) is empty.

hence \[ \langle M, w \rangle \in L_m \iff E(M') \in L_e \]

We can construct a halting T.M. \( M_e \) that, given \( \langle M, w \rangle \) as input, reconsystucts \( E(M') \) for an \( M' \) that behaves as above.

q.e.d.

To sum up, we have that \( L_e \) is R.E. but non-recursively enumerable.

Hence \( L_e \) must be non-R.E.!
The halting problem $L_{halt}$, the set $\{<M,w>|M \text{ halts on } w\}$ is R.E.

M halts on w (with or without accepting) is R.E. but not recursive.

To show R.E., we construct a T.M. $H$ s.t.
$L(H) = L_H = \{<M,w>|M \text{ halts on } w\}$

\[<M,w> \rightarrow H \rightarrow \text{yes/no} \]

To show that $L_H$ is not recursive, we assume by contradiction it is R.E. and derive that $L_{halt}$ is recursive.

By contradiction, let $U$ be an algorithm for $L_H$ and $V$ a procedure for $L_{halt}$

\[<M,w> \rightarrow H \rightarrow \text{yes/no} \]

$A_m$ would be an algorithm for $L_{halt}$.

Contradiction
Let $L$ be R.E. and $\overline{L}$ be non-R.E.

Consider $L' = \{0w \mid w \in L\} \cup \{1w \mid w \notin L\}$.

What do we know about $L'$ and $\overline{L'}$?

We show that $L'$ is non-R.E.

Suppose by contradiction that we have a procedure $M_{L'}$ for $L'$. Then we can construct a procedure $M_{\overline{L}}$ for $\overline{L}$ as follows:

- On input $w$, $M_{\overline{L}}$ changes the input to $1w$ and simulates $M_{L'}$.

- If $M_{L'}$ accepts $1w$, then $w \in L$, and $M_{\overline{L}}$ accepts.

- If $M_{L'}$ does not terminate on $1w$, then $w \notin L$, and $M_{\overline{L}}$ does not terminate or determines and answers no.

$\Rightarrow M_{\overline{L}}$ would accept exactly $\overline{L}$. Contradiction.

$\overline{L'} = \{0w \mid w \in L\} \cup \{1w \mid w \in L\} \cup \{1\}$

Reasoning as for $L'$, we get that $\overline{L'}$ is non-R.E.
The complement of the halting problem, i.e., the set of pairs \(<M, w>\) such that \(M\) on input \(w\) does not halt, is non-R.E.

Proof: By reduction from \(E_m\), which is non-R.E.

Idea: we show how to convert any TM \(M\) into another TM \(M'_w\) such \(M'_w\) halts on \(w\) iff \(M\) accepts \(w\).

Construction.

1) Ensure that \(M'_w\) does not halt unless \(M\) accepts.
   - add to the states of \(M\) a new loop state \(p\), with
     \[\delta(p, x) = (p, \times, \times)\] for all \(x \in \Gamma\)
   - for each \(\delta(q, y)\) that is undefined and \(q \in F\),
     add \(\delta(q, y) = (p, q, \times)\)

2) Ensure that, if \(M\) accepts, then \(M'_w\) halts
   - make \(\delta(q, x)\) undefined for all \(q \in F\) and \(x \in \Gamma\)

3) The other moves of \(M'_w\) are as those of \(M\)