Properties of RE languages

1. Union of $L_1 \cup L_2$ where $L_1, L_2$ are RE languages

   Design of $M$ which must accept
   union of $L_1$ and $L_2$. $M_1$ accepts $L_1$, $M_2$ accepts $L_2$

   Multiple TM which contains the description of $M_1$ and also contains description of $M_2$. It simulates alternately $M_1$ and $M_2$ and accepts if either $M_1$ or $M_2$ accepts.

   Now about intersection of $L_1$ and $L_2$

   Similar construction except that acceptance condition is both $M_1$ and $M_2$ accepts.
How about the case that $L_1, L_2$ are recursive?

Clearly $L_1 \cap L_2$ and $L_1 \cup L_2$ are also recursive even without using alternating simulation.

Also, if $L$ is recursive $\overline{L}$ is recursive.

Claim: If $L$ and $\overline{L}$ are r.e. then $L$ is recursive.

Proof: Simulate $L$ and $\overline{L}$ alternately. At least one of the machines for $L$ or $\overline{L}$ will halt. Report the decision.

If $L$ is r.e., is $\overline{L}$ r.e.?

We will need more tools to address this important question.
The universal language \( L_u \) is:

\[
L_u = \{ \langle M, \omega \rangle \mid M \text{ accepts } \omega \} = \{ \omega \in \Sigma^* \mid \omega \in L(M) \}
\]

\( M \) is the encoding of a TM \( M \):

\[
M = (Q, \Sigma, \delta, \Gamma, \delta, \tau)
\]

\[
\delta = \left[ (q, \Gamma, \delta, q', \tau, \{L, R\}) \right]
\]

**Claim**: The tape alphabet of any TM can be restricted to \( \{0, 1, B\} \).
Claim: \( L \subseteq \text{recursively enumerable}. \)

\( M_n \) simulates \( \langle M \rangle \) on \( w \)

1st tape

\[ M_n \]

2nd tape

contents of tape of \( M \) at any stage

current state of \( M \)

The first task of \( M_n \) is to copy \( w \) on tape 2

Then, simulate \( M \) on \( w \) by consulting \( M_n \)'s \( S \) function as described on the first tape.
Claim: \( L_n \) is r.e.

Claim: \( L_n \) is not recursive

\[
\begin{array}{cccccc}
 & w_1 & w_2 & w_3 & \cdots & w_j & \cdots \\
M_1 & 0 & 1 & 0 & 1 & \cdots & 1 \\
M_2 & 1 & 0 & 1 & 0 & \cdots & 0 \\
M_3 & 1 & 0 & 1 & 1 & \cdots & 0 \\
M_4 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
M_i & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{array}
\]

\( i = \ast \)

\[
T(i, j) = \begin{cases} 
1 & \text{if } w_j \in L(M_i) \\
0 & \text{otherwise} 
\end{cases}
\]

\( L_d = \{ x \mid x = w_i \text{ and } w_i \notin L(M_i) \} \)

Claim: \( L_d \) is not r.e.

There is no TM that accepts all strings of \( L_d \).
Proof (by contradiction)

Suppose there is a TM, say $M_k$ that accepts $L_d$.

Then argue the behavior:
- $M_k$ on $w_k$ is not defined.
- $w_k$ should not be accepted by $M_k$.

Contradiction:
- $M_k$ accepts $w_k$.
- $w_k \notin L(M_k)$.
- $w_k \in L_d$.
- $w_k$ should be accepted by $M_k$.

Contradiction.
Proof that \( L_u \) is not recursive

Suppose \( L_u \) is recursive. Then we can design a TM for \( L_d \)

\[
\begin{align*}
\text{Step 1: } & \quad \text{Given } x, \text{ find } w_i = x \\
& \quad \text{(There is a TM that can compute the correct } i) \\
\text{Step 2: } & \quad \text{Find } M_i \text{ (computable by a TM)} \\
\text{Step 3: } & \quad \text{Run the "recursive" TM for } L_u \text{ on } \langle M_i, w_i \rangle \\
\end{align*}
\]

If it accepts, we reject and vice versa.
Since $L_1$ is non-r.e. the recursive TM for $L_1$ cannot exist.

Thus $L_1$ is not recursive

$\Rightarrow \quad \overline{L_1}$ is not r.e.

Proof by reduction

Given two languages $L_1, L_2$, we say that $L_1 \leq L_2$ ($L_1$ is reducible to $L_2$) if there exists a Turing computable function $f : \Sigma^* \rightarrow \Sigma^*$

s.t. $x \in L_1$ if and only if $f(x) \in L_2$
many-to-one reduction

In the previous proof involving $L_d$ and $L_u$

$L_d \leq L_u$