1. (25 points) Let \( L_1, L_2 \) be two Turing-recognizable languages, with the additional property that \( L_1 \cup L_2 = \{0, 1\}^* \). Show that \( L_1 \leq_T (L_1 \cap L_2) \).

2. (25 points) Consider the following transformation that, given a pair \( \langle M, w \rangle \) consisting of a TM \( M \) and input \( w \), constructs the following TM \( M' \).

\( M' \): on an input string \( x \):
- Simulate the computation of \( M \) on \( w \) for \( |x| \) steps (where \( |x| \) is the length of \( x \)).
- If \( M \) does not accept \( w \) within these steps, then accept \( x \) and halt.
- If \( M \) does accept \( w \) within these steps, then reject \( x \) and halt.

(a) Supposed that \( M \) accepts \( w \). What is \( L(M') \)? (justify your answer)
(b) Suppose that \( M \) does not accept \( w \). What is \( L(M') \)? (justify your answer)

3. (50 points) Prove that the following languages are undecidable.

\( a \) PAL\(_{TM} = \{\langle M \rangle : M \text{ is a TM that accepts all palindromes} \}\).

\( b \) SUB\(_{TM} = \{\langle M_1, M_2 \rangle : M_1, M_2 \text{ are TMs and } L(M_1) \subseteq L(M_2) \}\).

\( c \) CONC\(_{TM} = \{\langle M_1, M_2, M_3 \rangle : M_1, M_2, M_3 \text{ are TMs, and } L(M_1) = L(M_2)L(M_3) \}\).

4. Extra Credit: Let \( \text{HALT}_\varepsilon = \{\langle M \rangle : M \text{ is a TM with } \Sigma = \{0, 1\}, \Gamma = \{0, 1, \varepsilon\} \text{ and } M \text{ halts on input } \varepsilon \} \). It is easy to prove that \( \text{HALT}_\varepsilon \) is undecidable (you do not need to show this).

Let COUNT be the problem of calculating, given a positive integer \( n \) as input, how many \( n \)-state TMs are in \( \text{HALT}_\varepsilon \).

First, convince yourself that COUNT \( \leq_T \) \( \text{HALT}_\varepsilon \) (you do not need to show this). Then, prove that a reduction in the other direction is also possible (which implies that there’s no algorithm for computing COUNT). That is, prove that \( \text{HALT}_\varepsilon \leq_T \) COUNT.

Note: While we defined Turing reductions for languages (or decision problems, returning 0/1 on each input), COUNT is a problem of computing a multi-bit output function (computing \( f : \mathbb{N} \to \mathbb{Z} \), where \( f(n) \) counts how many \( n \)-state TMs halt when run on a blank tape). Nonetheless, this should not make a big difference, and the definition of Turing-reducibility is extended in the natural way (i.e., given an oracle computing COUNT, you need to show an algorithm deciding \( \text{HALT}_\varepsilon \)).