Consider the language
\[ L_P = \{ \langle M \rangle \mid |L(M)| = 1 \}. \]

**Problem 1.** Identify what is wrong with the following proof that \( L_P \in \text{co-RE} \) and explain why.

To show that \( L_P \in \text{co-RE} \), we need to show that \( \overline{L_P} \in \text{RE} \). To do that, we need only show that there is a TM \( M \) that recognizes it. To convenience, let \( \omega_1, \omega_2, \ldots \) be the lexicographical (and thus computable) enumeration of the input alphabet \( \Sigma^* \).

**Algorithm 1:** \( M = \) On input \( \langle N \rangle \),
\[
\text{for } i = 1 \text{ to } \infty \text{ do } \\
\quad \text{for } j = 1 \text{ to } i \text{ do } \\
\quad \quad \text{Run } N(\omega_j) \text{ for } i \text{ steps } \\
\quad \quad \text{if } N \text{ accepted at least two distinct strings then } \\
\quad \quad \quad \text{Accept} \\
\]

If \( N \) ever accepts a string, then it must do so in a finite amount of time. If \( N \) accepts two distinct strings, then it must do so in times \( t_1, t_2 < \infty \). Then when \( i \geq \max(t_1, t_2) \), \( N \) will accept both strings, and \( M \) will accept. This implies \( \langle N \rangle \in L(M) \).

On the other hand, when \( N \) does not accept more than one string, \( M \) clearly cannot accept. This implies \( \langle N \rangle \notin L(M) \).

Thus \( L(M) = \overline{L_P} \).

**Solution 1.** The problem is that \( M \) fails to accept Turing machines whose languages are empty.

**Problem 2.** Show that \( L_P \notin \text{DEC} \) by (Turing) reducing \( A_{TM} \) to it.

**Solution 2.** To show that \( A_{TM} \leq_T L_P \), consider the following utility TM \( N_{M,\omega} \).

**Algorithm 2:** \( N_{M,\omega} = \) On input \( \nu \)
\[
\text{if } \nu \neq \omega \text{ then } \\
\quad \text{Reject} \\
\quad \text{Run } M(\omega) \\
\quad \text{if } M \text{ accepted } \omega \text{ then } \\
\quad \quad \text{Accept} \\
\quad \text{Reject} \\
\]
Clearly, we have

\[
L(N_{M,\omega}) = \begin{cases} 
\{\omega\} & M(\omega) \text{ accepts} \\
\emptyset & M(\omega) \text{ does not accept.}
\end{cases}
\]

In other words, \(N_{M,\omega} \in 1_{TM}\) if and only if \(M(\omega)\) accepts.

Now suppose \(1_{TM} \in DEC\). Then there is a decider \(D\) with \(L(D) = 1_{TM}\). Now consider the following Turing machine.

**Algorithm 3:** \(A = \text{On input } \langle M, \omega \rangle\)

- Run \(D(\langle N_{M,\omega} \rangle)\)
  - \textbf{if} \(D\) \textit{accepted} \textbf{then}
    - Accept
  - \textbf{Reject}

Clearly, \(A\) accepts \(\langle M, \omega \rangle\) if and only if \(D(\langle N_{M,\omega} \rangle)\) accepts, but this occurs if and only if \(M(\omega)\) accepts. Thus \(L(A) = A_{TM}\), hence \(A_{TM} \in DEC\). This is, of course, nonsense, so no such \(D\) can exist. Therefore, \(1_{TM} \not\in DEC\).

\(\Box\)