Problem 1

This is a problem in modeling execution of a (tiny!) program with LTL, prior to verifying with an automated model-checker whether some properties are satisfied by that program. Consider a fragment of a concurrent program with two threads, \( P \) and \( Q \), which share the variable \( n \):

\[
\begin{array}{c}
\text{print } a ; \text{ print } b ; \ n := 0 \\
\text{thread } P \hspace{5cm} \text{thread } Q \\
1 \hspace{1cm} \text{repeat} & 1 \hspace{1cm} \text{repeat} \\
2 \hspace{1cm} \text{begin} \text{ print } a ; & 2 \hspace{1cm} \text{begin} \text{ print } b ; \\
3 \hspace{1cm} \quad \ n := n + 1 ; & 3 \hspace{1cm} \quad \ n := n - 1 ; \\
4 \hspace{1cm} 4 \hspace{1cm} \quad \text{if } n = 2 & 4 \hspace{1cm} \quad \text{if } n = -2 \\
5 \hspace{1cm} \quad \text{begin} \text{ print reset } ; & 5 \hspace{1cm} \text{begin} \text{ print target } ; \\
6 \hspace{1cm} \quad \quad \ n := 0 & 5 \hspace{1cm} \quad \quad \ n := 0 ; \text{ stop thread } Q \\
7 \hspace{1cm} \text{end} & 7 \hspace{1cm} \text{end} \\
8 \hspace{1cm} \text{end} & 8 \hspace{1cm} \text{end}
\end{array}
\]

We take a \texttt{begin-end} block in each thread above as a “critical section” which is entirely executed without any interference from the other thread. This means that, once lines 2-8 in thread \( P \) starts executing, there is no interleaving with any of lines 2-8 in thread \( Q \), and vice-versa. Interleaving only occurs between \texttt{full begin-end} blocks, and not between portions of them. There are 5 parts in this problem.

(1) Define a transition system \( \mathcal{M} \) modeling the behavior of the two threads \( P \) and \( Q \) by drawing its diagram. In this part (1), ignore the printouts \{a, b, reset, target\}.

\textit{Hint for (1):} Draw the diagram of \( \mathcal{M} \) as a transition system with 8 states:

- 5 states when both \( P \) and \( Q \) are running and variable \( n \) stores an integer \( i \in \{0, 1, 2, -1, -2\} \). Denote by \( s[P, Q, i] \) the state of \( \mathcal{M} \) when \( P \) and \( Q \) are running and \( n \) stores \( i \).
- 3 states when only \( P \) is running and variable \( n \) stores an integer \( i \in \{0, 1, 2\} \). Denote by \( s[P, i] \) the state of \( \mathcal{M} \) when only \( P \) is running and \( n \) stores \( i \).

Take \( s[P, Q, 0] \) as the start state of \( \mathcal{M} \).

We can incorporate the printouts \{a, b, reset, target\} as labels in the model \( \mathcal{M} \) in two different ways, in parts (2) and (3) first, and then in parts (4) and (5).
(2) Consider the case when \( \{a, b, \text{reset}, \text{target}\} \) are labels for the transitions. For example, we can use label \( a \) to identify the transition from state \( s[P, Q, 0] \) to state \( s[P, Q, 1] \) by writing \( s[P, Q, 0] \xrightarrow{a} s[P, Q, 1] \), and we can use label \( b \) to identify the transition from state \( s[P, Q, 0] \) to state \( s[P, Q, -1] \) by writing \( s[P, Q, 0] \xrightarrow{b} s[P, Q, -1] \), etc.

Your task is to write a regular expression \( E \) over the alphabet \( \{a, b, r, t\} \) that denotes all finite sequences of transitions from \( s[P, Q, 0] \) back to \( s[P, Q, 0] \).

**Hint for (2):** For every \( i \in \{1, 2, -1\} \), define a regular expression \( E[P, Q, i] \) where \( i \neq 0 \) which denotes the sequences of transitions from \( s[P, Q, 0] \) to \( s[P, Q, i] \) without visiting these two states more than once each, followed by the sequences of transitions from \( s[P, Q, i] \) back to \( s[P, Q, 0] \) without visiting these two states more than once each. Try to write the desired \( E \) by using the three regular expressions \( E[P, Q, 1] \), \( E[P, Q, 2] \), and \( E[P, Q, -1] \).

(3) Consider again the case when \( \{a, b, \text{reset}, \text{target}\} \) are labels for the transitions. Your task is to write an \( \omega \)-regular expression \( F \) over the alphabet \( \{a, b, r, t\} \) that denotes all infinite sequences of transitions that start at state \( s[P, Q, 0] \) and visit state \( s[P, 0] \) infinitely often.

**Hint for (3):** Use regular expression \( E \) from part (2).

(4) For this part, we view \( \{a, b, r, t\} \) as atomic propositions which are labels for the states (not the transitions) in \( M \). For example, if \( S \) is the set of states and \( L \) is the labelling function,

\[
S = \{s[P, Q, 0], \ldots, s[P, 2]\} \quad \text{and} \quad L : S \rightarrow \text{propositional WFF's over} \{a, b, r, t\},
\]

then we can write \( L(s[P, Q, 0]) = \{a \land b\} \) to mean that the propositional WFF \( a \land b \) is true at state \( s[P, Q, 0] \); and \( L(s[P, Q, 1]) = \{a\} \) to mean that the atom \( a \) is true at \( s[P, Q, 1] \); etc.

Your task is to complete the definition of the labelling function \( L \).

(5) This is a continuation of part (4). We are given the three LTL formulas:

\[
\varphi_1 \triangleq \mathbf{G} r, \quad \varphi_2 \triangleq \mathbf{G} \left( (r \lor t) \rightarrow (\mathbf{X} a \land \mathbf{X} \mathbf{X} a) \right), \quad \varphi_3 \triangleq \left( (a \lor b \lor r) \mathbf{U} t \right).
\]

For each LTL formula \( \varphi_i \) above, with \( i \in \{1, 2, 3\} \), your task is twofold:

1. Find a path \( \pi_i \) in \( M \) (whose first state is the start state \( s[P, Q, 0] \)) such that \( M, \pi_i \models \varphi_i \).
2. Decide whether \( M \models \varphi_i \).

**Problem 2** [LCS, page 246]: Exercise 3.2.8. You need to extend the algorithm NNF on page 62, which is defined for propositional WFFs only, to all the WFFs of LTL.

---

1Use \( \{a, b, r, t\} \) instead of \( \{a, b, \text{reset}, \text{target}\} \) for simplicity.