Problem 1. Software Modeling, Regular Expressions, LTL Model Checking. Consider a fragment of a concurrent program with two threads, $P$ and $Q$, sharing one variable $n$:

\[
\begin{array}{c|c}
\text{print } a ; & \text{print } b ; \quad n := 0 \\
\hline
\text{thread } P & \text{thread } Q \\
1 \quad \text{repeat} & 1 \quad \text{repeat} \\
2 \quad \text{begin print } a ; & 2 \quad \text{begin print } b ; \\
3 \quad n := n + 1 ; & 3 \quad n := n - 1 ; \\
4 \quad \text{if } n = 3 \text{ repeat} & 4 \quad \text{if } n = -3 \text{ repeat} \\
5 \quad \text{begin print } c ; \langle \ldots \rangle & 5 \quad \text{begin print } d ; \langle \ldots \rangle \\
6 \quad \text{end} ; & 6 \quad \text{end} ; \\
7 \quad \text{end} ; & 7 \quad \text{end} ; \\
\end{array}
\]

We take the outer \texttt{begin-end} block in each of the two threads as a “critical section” which is entirely executed without any interference from the other thread. (The ellipses $\langle \ldots \rangle$ in line 5, in both $P$ and $Q$, are portions of the program that are not relevant for our analysis here.) Once lines 2-7 in thread $P$ starts executing, there is no interleaving with any of lines 2-7 in thread $Q$, and vice-versa. Interleaving only occurs between \textit{full outer} \texttt{begin-end} blocks, and not between portions of them. There are 5 parts in this problem.

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

Draw $M$ with 7 states, denoted $\{s[0], s[1], s[-1], s[2], s[-2], s[3], s[-3]\}$, corresponding to the 7 possible values of the shared variable $n$. Take $s[0]$ as the start state of $M$.

\[\text{Answer:}\]

\[\begin{array}{c|c|c|c}
\text{State} & \text{Transition} & \text{State} & \text{Transition} \\
\hline
s(0) & s(1) & s(2) & s(3) \\
\hline
s(1) & \langle b \rightarrow a \rightarrow a \rightarrow c \rangle & s(2) & \langle a \rightarrow b \rightarrow b \rightarrow d \rangle \\
\hline
\end{array}\]

(2) Consider the case when $\{a, b, c, d\}$ are labels for the transitions. We can thus use label “$a$” to identify the transition from state $s[0]$ to state $s[1]$ by writing $s[0] \xrightarrow{a} s[1]$, and we can use
label “b” to identify the transition from state $s[0]$ to state $s[−1]$ by writing $s[0] \xrightarrow{b} s[−1]$, etc. We can also label with “ab” the arrow identifying $s[0]$ as the start state, because of the “print $a$” and “print $b$” (in that order) before the threads $P$ and $Q$ are entered.

Your task is to write regular expressions $E_0$, $E_1$, and $E_2$, over alphabet $\{a, b, c, d\}$ such that:

- $E_0$ denotes all finite sequences of transitions from $s[0]$ back to $s[0]$ without visiting $s[0]$ in any intermediate step.
- $E_1$ denotes all finite sequences of transitions from $s[0]$ to $s[3]$.
- $E_2$ denotes all finite sequences of transitions from $s[0]$ to $s[−3]$.

**Hint for (2):** Write $E_1$ and $E_2$ using $E_0$.

**Answer:**

\[
E_0 = a (ab)^* b + b (ba)^* a \\
E_1 = (E_0)^* a (ab)^* a \ a c^* \\
E_2 = (E_0)^* b (ba)^* b b d^*
\]

(3) Consider again the case when $\{a, b, c, d\}$ are labels for the transitions. Your task is to write two $\omega$-regular expression $F_1$ and $F_2$ over the alphabet $\{a, b, c, d\}$ such that: $F_1$ denotes all infinite sequences of transitions that start at state $s[0]$ and visit state $s[3]$ infinitely often, and $F_2$ denotes all infinite sequences of transitions that start at state $s[0]$ and visit state $s[−3]$ infinitely often.

**Hint for (3):** Use regular expression $E_1$ and $E_2$ from part (2).

**Answer:** Following the hint,

\[
F_1 = (E_0)^* a (ab)^* a \ a c^\omega \\
F_2 = (E_0)^* b (ba)^* b b d^\omega
\]

where $E_0$ is defined in the answer for (2).

(4) For this part, we view $\{a, b, c, d\}$ 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,

\[
S = \{s[0], \ldots, s[−3]\} \quad \text{and} \quad L : S \rightarrow \{\text{propositional WFF’s over } \{a, b, c, d\}\},
\]

then we can write $L(s[0]) = (a \lor b)$ to indicate that atom $a$ or atom $b$ (or both) are true at state $s[0]$, i.e., to mean that “if execution is at state $s[0]$, then this follows an action $a$ or an action $b$ (or both)”.

Your task is to complete the definition of the labelling function $L$.

**Answer:** Following the hint,

\[
L(s[0]) \triangleq (a \lor b) \quad L(s[1]) \triangleq (a \lor b) \quad L(s[2]) \triangleq a \quad L(s[3]) \triangleq (a \lor c) \\
L(s[−1]) \triangleq (a \lor b) \quad L(s[−2]) \triangleq b \quad L(s[−3]) \triangleq (b \lor d)
\]
(5) This is a continuation of part (4). We are given three LTL formulas:

\[ \varphi_1 \triangleq GF (c \lor d), \quad \varphi_2 \triangleq G ((a \lor b) \rightarrow XXX (c \lor d)), \quad \varphi_3 \triangleq ((a \lor b) W (c \lor d)). \]

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 \( s[0] \)) such that \( M, \pi_i \models \varphi_i \).
2. Decide whether \( M \models \varphi_i \).

**Answer:** Consider each formula in turn:

- Any \( \omega \)-path \( \pi_1 \) reaching state \( s[3] \) or state \( s[-3] \) is such that \( M, \pi_1 \models \varphi_1 \).
  However, \( M \not\models \varphi_1 \), because there are \( \omega \)-paths \( \rho \) such that \( M, \rho \not\models \varphi_1 \). For example, take \( \rho \) as the path that keeps going around the cycle \( s[0] s[1] s[0] \) for ever.

- Any \( \omega \)-path \( \pi_2 \) reaching state \( s[3] \) or state \( s[-3] \) is such that \( M, \pi_2 \models \varphi_2 \).
  However, \( M \not\models \varphi_2 \), because there are \( \omega \)-paths \( \rho \) such that \( M, \rho \not\models \varphi_2 \). For example, take \( \rho \) as the path that keeps going around the cycle \( s[0] s[1] s[0] \) for ever.

- We can take \( \pi_3 \) to be any \( \omega \)-path from \( s[0] \), without restriction, in order to get \( M, \pi_3 \models \varphi_3 \).
  Hence, a fortiori, \( M \models \varphi_3 \).

Note that if \( \varphi_3 \) were re-written using “U” instead of “W”, to obtain \( \varphi'_3 \), say, then \( M \not\models \varphi'_3 \).

![Figure 1](image_url)

**Figure 1:** Graphical representation of the discrete-time Markov chain \( A \) in Problem 2.

Propositional atom \( a \) is true in states \( s_3 \) and \( s_4 \), and \( b \) is true in state \( s_4 \) (the green states).

**Problem 2. Probabilistic Model Checking.** Figure 1 shows a discrete-time Markov chain \( A \):

\[
A \triangleq (S, P, \text{init}, \text{AP}, L) \quad \text{where}
\]

\[
S \triangleq \{s_0, s_1, s_2, s_3, s_4, s_5\}, \quad \text{(set of states)},
\]

\[
P : S \times S \rightarrow [0, 1] \quad \text{(probabilistic transition function)},
\]

\[
\text{init} \triangleq (1, 0, 0, 0, 0, 0) \quad \text{(initial state distribution)},
\]

\[
\text{AP} \triangleq \{a, b\} \quad \text{(atomic propositions)},
\]

\[
L : S \rightarrow 2^{\text{AP}} \quad \text{(labelling function)}.
\]
(6) Write the right-stochastic matrix $M$ representing the transition function $P$:

**Answer:**

$$M = \begin{bmatrix}
0 & 0.5 & 0.5 & 0 & 0 & 0 \\
0.5 & 0 & 0 & 0.5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.5 & 0.5 \\
0 & 0 & 0 & 1.0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0
\end{bmatrix}$$

We consider three WFF’s in the syntax of LTL:

$$\varphi_1 \triangleq F(a \land b), \quad \varphi_2 \triangleq XF a, \quad \text{and} \quad \varphi_3 \triangleq G(\varphi_1 \lor \varphi_2).$$

before embedding them in PCTL, as follows:

$$\varphi_1' \triangleq F^{\geq 1/2}(a \land b), \quad \varphi_2' \triangleq X^{\geq 1/2}F^{\geq 1/2} a, \quad \text{and} \quad \varphi_3' \triangleq G^p(\varphi_1' \lor \varphi_2').$$

where $p$ is a probability to be yet determined (in part (10) below).

(7) Compute the three probabilities $p_1$, $p_2$, and $p_3$, defined by:

$$p_1 \triangleq \Pr(\mathcal{A}, s_0 \models \varphi_1), \quad p_2 \triangleq \Pr(\mathcal{A}, s_0 \models \varphi_2), \quad \text{and} \quad p_3 \triangleq \Pr(\mathcal{A}, s_0 \models (\varphi_1 \lor \varphi_2)).$$

**Answer:** All $\omega$-paths that reach state $s_4$ will satisfy $\varphi_1$, which implies that:

$$p_1 = \sum_{i \geq 0} (0.5 \times 0.5)^i \times 0.5 \times 0.5 = \sum_{i \geq 1} (0.25)^i$$

If you remember something from your calculus course, or you know something about infinite series, you can compute the exact value of $p_1$, which is $p_1 = 1/3$ (not “around 1/3” or $p_1 \approx 1/3$ or $p_1 \sim 1/3$, as some wrote in the exam). But there is a simpler way, further below.

All $\omega$-paths that reach state $s_3$ or state $s_4$ will satisfy $\varphi_2$, which implies that:

$$p_2 = 2 \times \sum_{i \geq 0} (0.5 \times 0.5)^i \times 0.5 \times 0.5 = 2 \times \sum_{i \geq 1} (0.25)^i$$

Further below is a simple way of computing $p_2 = 2/3$.

The $\omega$-paths satisfying $\varphi_2$ are also the $\omega$-paths satisfying $\varphi_3$, so that $p_3 = 2/3$.

The simplest way to compute $p_1$, $p_2$, and $p_3$, is to notice: (a) every $\omega$-path that reaches state $s_3$, or state $s_4$, or state $s_5$, occurs with the same probability:

$$(0.5 \times 0.5)^i \times 0.5 \times 0.5 = (0.25)^i \times 0.25$$

for some $i \geq 0$, and (b), as a consequence, the probabilities of reaching $s_3$, $s_4$, and $s_5$, are equal, and (c) the probability of reaching $s_3$ or $s_4$ or $s_5$ must be $1/3$ (since their sum must add up to 1).

Another way, not as simple, but still producing the exact values of $p_1$, $p_2$, and $p_3$, is to set up a system of linear equations, as on pages 26-27 of Handout 24.
Determine the largest possible probability $p$ such that $A, \text{init} \models \varphi_1$ holds or not in PCTL (at most 3-4 lines).

**Answer:** $A, \text{init} \models \varphi_1$ does not hold. It is worth noting that it holds if execution starts with $(0, 0, 1, 0, 0)$ (initial state $s_2$) or with $(0, 0, 0, 1, 0)$ (initial state $s_4$).

(9) Give a rigorous argument whether the satisfaction statement $A, \text{init} \models \varphi_2$ holds or not in PCTL (at most 3-4 lines).

**Answer:** The states where $a$ holds are $s_3$ and $s_4$. Using the same reasoning as in part (7):

- probability of reaching $s_3$ or $s_4$ from $s_1 = 3/4$
- probability of reaching $s_3$ or $s_4$ from $s_2 = 1/2$

Hence:

$A, \text{init}_1 \models \text{F}^{3/4} a$ where $\text{init}_1 = (0, 1, 0, 0, 0, 0)$, i.e., starting from $s_1$,

$A, \text{init}_2 \models \text{F}^{1/2} a$ where $\text{init}_2 = (0, 0, 1, 0, 0, 0)$, i.e., starting from $s_2$.

It follows that $A, \text{init}_1 \models \text{F}^{3/4} a$ and $A, \text{init}_2 \models \text{F}^{1/2} a$. And because $s_1$ or $s_2$ are reached from $s_0$ with probability $= 1$, i.e., each is reached from $s_0$ with probability $= 1/2$, we conclude $A, \text{init} \models \text{X}^{1/2} \text{F}^{3/4} a$ and, a fortiori, $A, \text{init} \models \text{X}^{1/2} \text{F}^{3/1/2} a$. Hence, $A, \text{init} \models \varphi_2$ does hold.

(8) Give a rigorous argument whether the satisfaction statement $A, \text{init} \models \varphi_1$ holds or not in PCTL (at most 3-4 lines).

(10) Determine the largest possible probability $p$ such that: $A, \text{init} \models \varphi_3$. Justify your answer carefully (at most 3-4 lines).

**Answer:** $A, \text{init} \models \varphi_3$ holds for every init except init $= (0, 0, 0, 0, 1)$ (state $s_5$). Hence, $p = 2/3$.

**Problem 3.** **Modeling Transition Systems in Propositional Logic.** Consider the transition system $\mathcal{N}$ shown in Figure 2: It has four states $\{s_0, s_1, s_2, s_3\}$ and its transitions are labelled with one of two actions: $m$ or $n$. There are two atomic propositions: $x$ and $y$. This is a deterministic system because, from every state $s_i$, there are two transitions, one labelled $m$ and one labelled $n$.

We can fully describe $\mathcal{N}$’s behavior by a program $\mathcal{P}$ written in pseudo-code as follows:

$$\mathcal{P} \triangleq \text{if current} = s_0 \text{ then } (\text{if action} = m \text{ then next} = s_1 \text{ else next} = s_3) \text{ else}$

$$\text{if current} = s_1 \text{ then } (\text{if action} = m \text{ then next} = s_2 \text{ else next} = s_3) \text{ else}$

$$\text{if current} = s_2 \text{ then } (\text{if action} = m \text{ then next} = s_0 \text{ else next} = s_2) \text{ else}$

$$\text{if current} = s_3 \text{ then } (\text{if action} = m \text{ then next} = s_3 \text{ else next} = s_3) \text{ else no op}$

Observe that program $\mathcal{P}$ mentions action $m$ but not action $n$, but this does not create any ambiguity in representing the behavior of $\mathcal{N}$, because whenever the test “action $= m$” fails, it implicitly means that the other action $n$ succeeds. Moreover, program $\mathcal{P}$ does not mention atoms $x$ and $y$ explicitly, but this too does not create any ambiguity in representing $\mathcal{N}$, because there is a one-one correspondence between the states in $\{s_0, s_1, s_2, s_3\}$ and the truth values of $x$ and $y$.
Atom $x$ is true at $s_0$, atom $y$ is true at $s_1$, both atoms $x$ and $y$ are true at $s_2$, and neither $x$ nor $y$ are true in $s_3$. The green states are the states where only one of $\{x, y\}$, not both, is true.

- $s_0$ uniquely corresponds to $x = true$ and $y = false$,
- $s_1$ uniquely corresponds to $x = false$ and $y = true$,
- $s_2$ uniquely corresponds to $x = true$ and $y = true$,
- $s_3$ uniquely corresponds to $x = false$ and $y = false$.

(11) We can view the programming construct if-then-else as a ternary logical connective in propositional logic, whose meaning is given by the following truth-table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>if $p$ then $q$ else $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Show that if-then-else is an adequate logical connective, i.e., every propositional WFF $\varphi$ is equivalent to a propositional WFF $\varphi'$ where no connective other than if-then-else appears.
Answer: It suffices to show that any other set of connectives, already known adequate (for example \(\{\neg, \lor\}\) or \(\{\neg, \land\}\)), can be simulated with \textbf{if-then-else}. If we take \(\{\neg, \land\}\), we can define \(\neg\) and \(\land\) as follows:

\[
\begin{align*}
\neg x & \triangleq \text{if } x \text{ then } \bot \text{ else } \top \\
x \land y & \triangleq \text{if } x \text{ then } y \text{ else } \bot
\end{align*}
\]

(12) Show that program \(P\) can be translated into a single propositional WFF \(\Phi_N\) such that:

\begin{itemize}
  \item the only logical connective in \(\Phi_N\) is \textbf{if-then-else},
  \item there are no more than 5 propositional variables in \(\Phi_N\).
\end{itemize}

**Hint for (12):** Use part (11). Moreover, use 5 variables \(\{x, y, x', y', a\}\) such that the truth values of \(\{x, y\}\) correspond to “current state”, the truth values of \(\{x', y'\}\) correspond to “next state”, and the truth values of \(\{a\}\) indicate whether action \(m\) or action \(n\) is used to make the transition.

Answer:

\[
\Phi_N \triangleq \text{if } x \text{ then if } y \text{ then if } a \text{ then } x' \land \neg y' \\
&\quad\text{else } x' \land y' \\
&\quad\text{else if } a \text{ then } \neg x' \land y' \\
&\quad\text{else } \neg x' \land \neg y' \\
&\quad\text{else if } y \text{ then if } a \text{ then } x' \land y' \\
&\quad\text{else } \neg x' \land \neg y' \\
&\quad\text{else if } a \text{ then } \neg x' \land \neg y' \\
&\quad\text{else } \neg x' \land \neg y'
\]

where the last column has to be expanded further using \textbf{if-then-else}. For example, \(x' \land \neg y'\) should be expanded as:

\[
\text{if } x' \text{ then (if } y' \text{ then } \bot \text{ else } \top) \text{ else } \bot
\]

and similarly for the remaining entries in the last column, using the definitions of \(\neg\) and \(\land\) in part (11) in terms of \textbf{if-then-else}.

(13) This is a refinement of part (12). Show that program \(P\) can be translated into a single propositional WFF \(\Phi'_N\) such that:

\begin{itemize}
  \item the only logical connective in \(\Phi'_N\) is \textbf{if-then-else},
  \item there are no more than 4 propositional variables in \(\Phi'_N\).
\end{itemize}
Answer:

$$\Phi'_N \triangleq \text{if } x \text{ then if } y \text{ then } (x' \land \neg y') \lor (x' \land y')$$

$$\text{else if } y \text{ then } (x' \land y') \lor (\neg x' \land \neg y')$$

$$\text{else if } y \text{ then } (x' \land y') \lor (\neg x' \land \neg y')$$

Again here, the “\lor” in the last column has to be expanded further using if-then-else, based on the definitions in part (11).

(14) Given your representation of the 4 states of $N$ with two propositional variables, as in part (12) or in part (13), write a propositional WFF $\varphi_{i,j}$ representing the set of states $\{s_i, s_j\}$ for each of the six possible combinations of two states:

- $\{s_0, s_1\}$
- $\{s_0, s_2\}$
- $\{s_0, s_3\}$
- $\{s_1, s_2\}$
- $\{s_1, s_3\}$
- $\{s_2, s_3\}$

We want to use such a WFF $\varphi_{i,j}$ to indicate that execution of $N$ is in state $s_i$ or in state $s_j$, i.e., for any assignment $I$ of truth values to the propositional variables, we want:

$I \models \varphi$ iff “execution of $N$ is in state $s_i$ or in state $s_j$”

Hint for (14): Use any of the logical connectives, no need to restrict yourself to if-then-else.

Answer:

- $\{s_0, s_1\}$ represented by $\varphi_{0.1} \triangleq (x \land \neg y) \lor (\neg x \land y)$
- $\{s_0, s_2\}$ represented by $\varphi_{0.2} \triangleq (x \land y)$
- $\{s_0, s_3\}$ represented by $\varphi_{0.3} \triangleq (x \land y)$
- $\{s_1, s_2\}$ represented by $\varphi_{1.2} \triangleq (\neg x \land y)$
- $\{s_1, s_3\}$ represented by $\varphi_{1.3} \triangleq (\neg x \land y)$
- $\{s_2, s_3\}$ represented by $\varphi_{2.3} \triangleq (x \land y)$

(15) Consider the following WFF of LTL:

$$\theta \triangleq G(\varphi_{0.1} \rightarrow X(\varphi_{1.3} \lor \varphi_{2.3}))$$

where each $\varphi_{i,j}$ is defined in part (14). By inspection, the assertion $N \models_{\text{LTL}} \theta$ holds, according to the formal semantics of LTL. Your task is to translate the assertion “$N \models_{\text{LTL}} \theta$” into a WFF $\Psi$ of propositional logic so that the truth of the assertion (at the meta-level) in LTL is equivalent to the satisfiability of $\Psi$ in propositional logic.
Answer: (Details added on April 4, 2016.) Write $\varphi'_{i,j}$ for the WFF $\varphi_{i,j}$ after substituting $x'$ for $x$ and $y'$ for $y$. The statement “$\mathcal{N} \models_{\text{LTL}} \theta$” is equivalent to the statement:

$$\Phi_N' \models_{\text{PL}} (\varphi_{0,1} \rightarrow (\varphi'_{1,3} \lor \varphi'_{2,3}))$$

because $\Phi_N'$ from part (13) is a complete description of $\mathcal{N}$, and $(\varphi_{0,1} \rightarrow (\varphi'_{1,3} \lor \varphi'_{2,3}))$ is a translation of $\theta$ in propositional logic (PL). Hence, by soundness of completeness of PL, we can write the sequent:

$$\Phi_N' \vdash_{\text{PL}} (\varphi_{0,1} \rightarrow (\varphi'_{1,3} \lor \varphi'_{2,3}))$$

which is derivable iff the following propositional WFF is valid:

$$\Phi_N' \rightarrow (\varphi_{0,1} \rightarrow (\varphi'_{1,3} \lor \varphi'_{2,3}))$$

Hence, the desired $\Psi$ can be written as:

$$\Psi \triangleq \Phi_N' \rightarrow (\varphi_{0,1} \rightarrow (\varphi'_{1,3} \lor \varphi'_{2,3}))$$

which is valid iff the satisfaction statement “$\mathcal{N} \models_{\text{LTL}} \theta$” holds.

Problem 4. LTL Definability and CTL Definability.

(16) Write a LTL formula which enforces the following requirement in a transition system: At every state $s$, if $p$ is true, then at every state $s'$ reachable from $s$ in one transition or more, if $q$ is true, then $r$ is false until $t$ becomes true (for all continuations of the execution path starting at $s'$).

Answer: Although the translation of informal requirements into formal specifications is generally fraught with ambiguities, in this case the informal requirement can be directly translated to:

$$G \left( p \rightarrow XG (q \rightarrow \neg r U t) \right)$$

(17) Write a LTL formula which enforces the following requirement in a a transition system: For every state $s_i$ along an execution path $s_1 s_2 s_3 \cdots$, unless $s_i$ is the first state $s_1$, if $p$ is true in $s_i$, then $p$ must be true in at least one of the two states just before $s_i$, i.e., in the states $s_{i-1}$ and $s_{i-2}$.

Answer: The informal requirement in English can be directly translated to:

$$(Xp \rightarrow p) \land G (XXp \rightarrow p \lor Xp)$$

(18) Write a LTL formula which enforces the following requirement in a a transition system: In every odd state along an execution path $\pi = s_1 s_2 s_3 \cdots$ the atom $p$ is true, and in every even state of the same path $\pi$ the atom $p$ is false.

Answer: The informal requirement in English can be directly translated to:

$$p \land G (p \leftrightarrow X\neg p)$$
Write a CTL formula (not LTL formula) which enforces the following requirement in a transition system: At every state \( s \), if \( p \) is true, then at every state \( s' \) reachable from \( S \) in one transition or more, if \( q \) is true, then \( r \) is false until \( t \) becomes true (for all continuations of the execution path starting at \( s' \)).

\textbf{Answer}: In this case, it suffices to insert the appropriate quantifiers into LTL formula in (16):

\[
AG \left( p \rightarrow AXAG (q \rightarrow A[\neg r U t]) \right)
\]

Write a CTL formula (not LTL formula) which enforces the following requirement in a transition system: There exists a path \( \pi \) such that, for every state \( s \) on \( \pi \), there exists a path \( \pi' \) starting at \( s \), which eventually enters a state \( s' \) where \( p \) is true and which, immediately after \( s' \), enters another state \( s'' \) where \( \neg p \) is true.

\textbf{Answer}: The informal requirement in English can be directly translated to:

\[
EG EF (p \land EX \neg p)
\]