ARJYA DAS
Email: arjyadas@iisc.ac.in

Assignment 2
SR No. : 04-04-00-10-51-21-1-19516

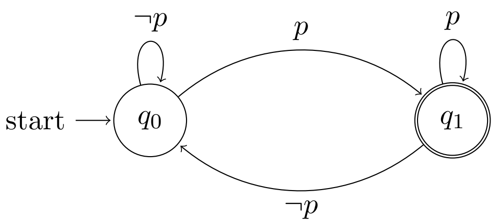
Course: E0 272: Formal Methods in Software Engineering
Term: January-April 2022

Problem 1:

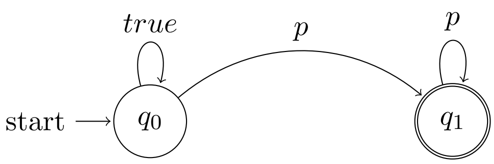
Equivalence of LTL assertions:

(a) $G(F_p)$ and $F(G_p)$:

Büchi automata for $G(F_p)$:



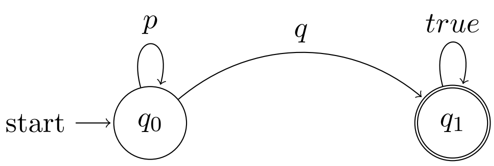
Büchi automata for $F(G_p)$:



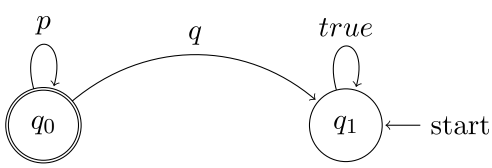
Notice, $G(F_p)$ accepts $\neg p(p\neg p)^\omega$. But, $F(G_p)$ does not accept $\neg p(p\neg p)^\omega$. So, the counter example shows, the pair of LTL assertions are not equivalent.

(b) $\neg(pUq)$ and $(\neg q)U(\neg p)$:

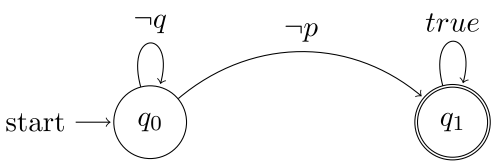
Büchi automata for (pUq):



Büchi automata for $\neg(pUq)$:



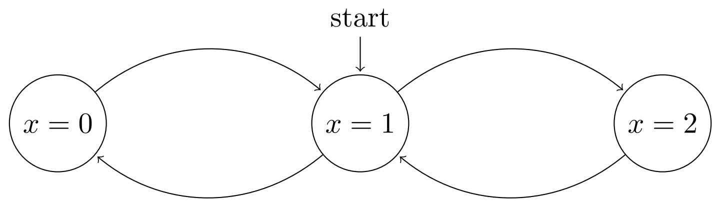
Büchi automata for $(\neg q)U(\neg p)$:



Notice, $\neg(pUq)$ does not accept $\neg q(\neg p)^\omega$. But, $(\neg q)U(\neg p)$ accepts $\neg q(\neg p)^\omega$. So, the pair of LTL assertions are not equivalent.

Problem 2:

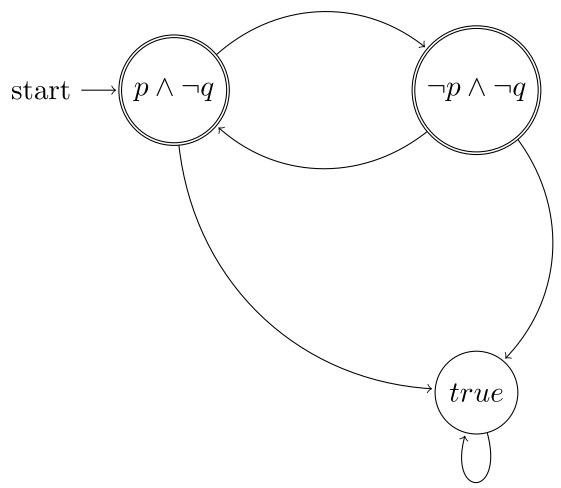
(a) *Transition System:*



(b) No, the system does not satisfy the property "$(x = 1) \implies F(x = 2)$". Initially, the system is in "$x = 1$" state. Then it non-deterministically goes to either "$x = 0$" state or "$x = 2$" state. Then it again comes back to "$x = 1$" state. For an infinite sequence of states of an execution it may happen that the system falls into the loop of "$x = 1$" state and "$x = 0$" state i.e. 10101010... . In this scenario, in future, "$x = 2$" state won't appear. So, the property "$(x = 1) \implies F(x = 2)$" is not satisfied.

(c) p represents $(x = 1)$ and q represents $(x = 2)$. We have to draw a state based Büchi automata for $\neg((x = 1) \implies F(x = 2))$ i.e. $\neg(p \implies F_q)$. Using the fact $F_p = \neg G_{\neg p}$ we can write,
$\neg(p \implies F_q) = \neg(\neg p \vee F_q) = p \wedge \neg F_q = p \wedge \neg(\neg G_{\neg q}) = p \wedge G_{\neg q}$

State based Büchi automata:



Initially, $p \wedge \neg q$ must hold. After that $\neg q$ must hold and either p or $\neg p$ should hold. If q holds then automata will go to the $true$ state which is a non-accepting state and will remain on it for every future transitions.

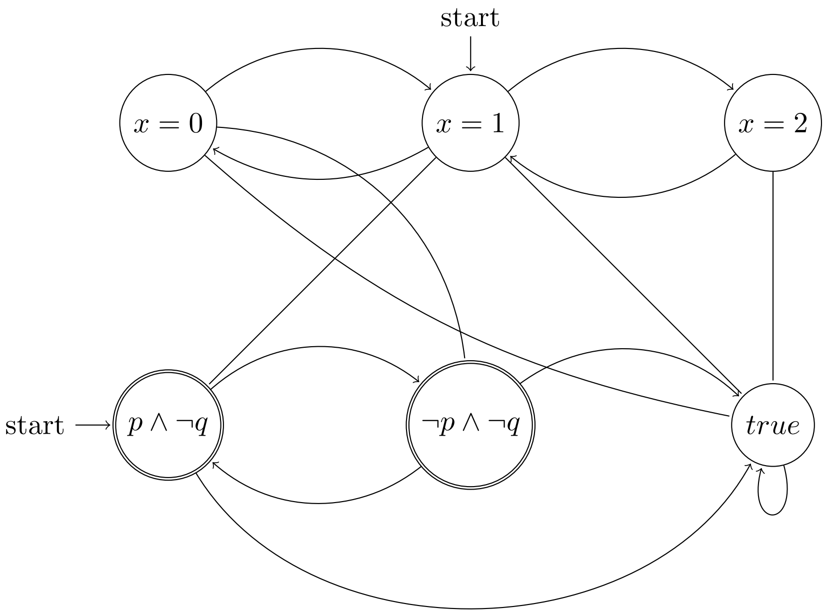
(d) *The product of the transition system of the model and the Büchi automaton for the formula:*

State 1: $p \wedge \neg q$ implies $(x = 1) \wedge \neg(x = 2)$ i.e. $(x = 1)$.

State 2: $\neg p \wedge \neg q$ implies $\neg(x = 1) \wedge \neg(x = 2)$ i.e. $(x = 0)$.

State 3: $true$ implies $(x = 0), (x = 1), (x = 2)$.

Check the automata of $T \wedge A_{\neg\text{formula}}$ in the next page:



Notice, that the above automata accepts the word described by the sequence "State 1 - State 2 - State 1 - State 2 - State 1 - State 2" (check the notation in previous page). This counter example shows that the formula is not satisfied by the transition system.

Problem 3:

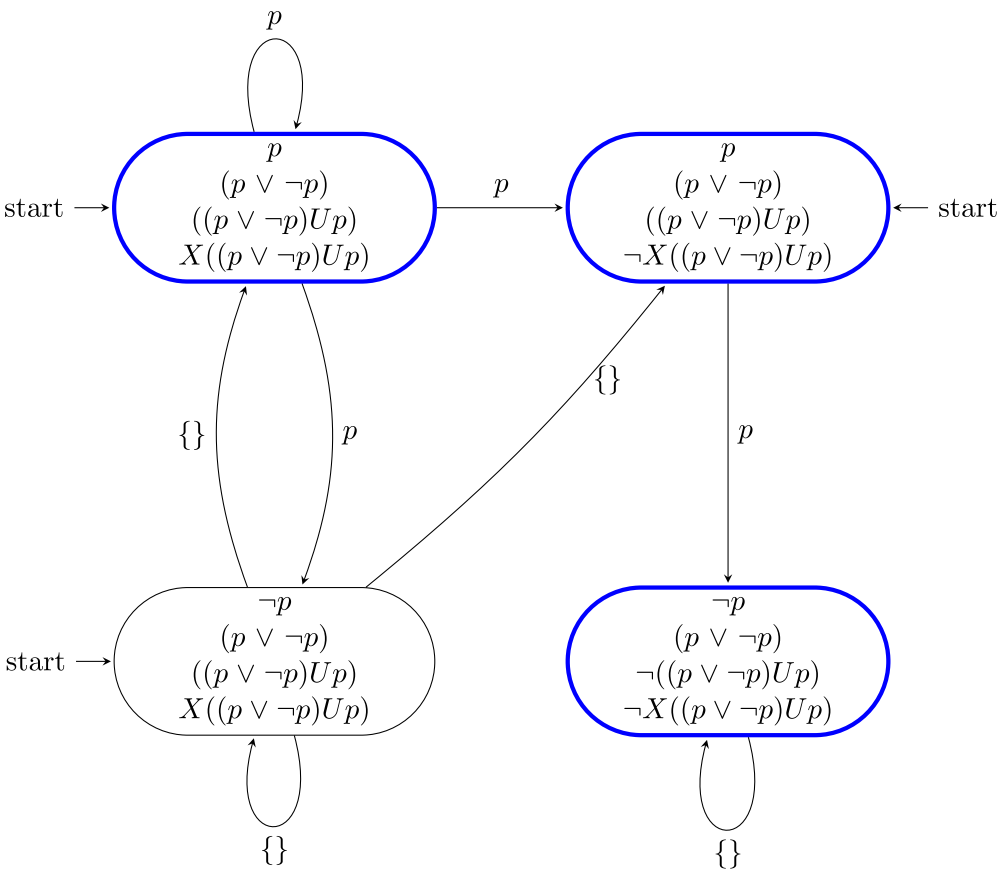
Formula automaton for LTL formula $trueUp$:

The LTL formula $trueUp$ can be written as: $(p \vee \neg p)Up$. We will construct the formula automaton for $(p \vee \neg p)Up$.

Closure of $(p \vee \neg p)Up$:

$cl((p \vee \neg p)Up) = \{p, \neg p, (p \vee \neg p), \neg(p \vee \neg p), ((p \vee \neg p)Up), \neg((p \vee \neg p)Up), X((p \vee \neg p)Up), \neg X((p \vee \neg p)Up)\}$

Automaton:



Final states are denoted by blue color.

Problem 4:

Traffic Light for Pedestrian Crossing:

Note: Only the final version i.e. *tlight-v3.pml* contains the comments that describes the meaning for each condition and when we apply them.

The first version - *tlight-v1.pml* satisfies the LTL properties: *safety*, *PedLiveness* and *VehLiveness*. But it doesn't satisfy the LTL property *VehGreen*. Main reason behind that is: we are changing the *vstatus* from GO to $GSCHANGE$ whenever *tick* is true and $vctr >= 3$. We should check the value of *pctr* and *cbutton* while changing the *vstatus*. When *vstatus* is GO, *tick* is true and $vctr >= 3$; if: $pctr = 0$ and *cbutton* is true, it is implied that a new pedestrian has come and no one is there at the crossing from previous. In this case *pctr* is changed to 1. Else if: *pctr* value is not zero then it is implied at least one pedestrian is waiting to cross. In this case we make *pstatus* $STOP$ and else guard takes care of increasing the *pctr* value.

The second version - *tlight-v2.pml* satisfies all the LTL properties. However, some states are unreachable. Reason behind this is: we have put some redundant conditions for the $vstatus = STOP$ that actually do not occur. We don't need to check the value of *pctr* and *cbutton* for this case as *vlight* can't remain red for more than 3 units of time at a stretch, so we should change the *vstatus* irrespective of variable status of *pctr* and *cbutton*.

After making the necessary changes mentioned above we get the final version - *tlight-v3.pml* satisfies all the LTL properties and contains less number of unreachable states for the corresponding LTL properties.