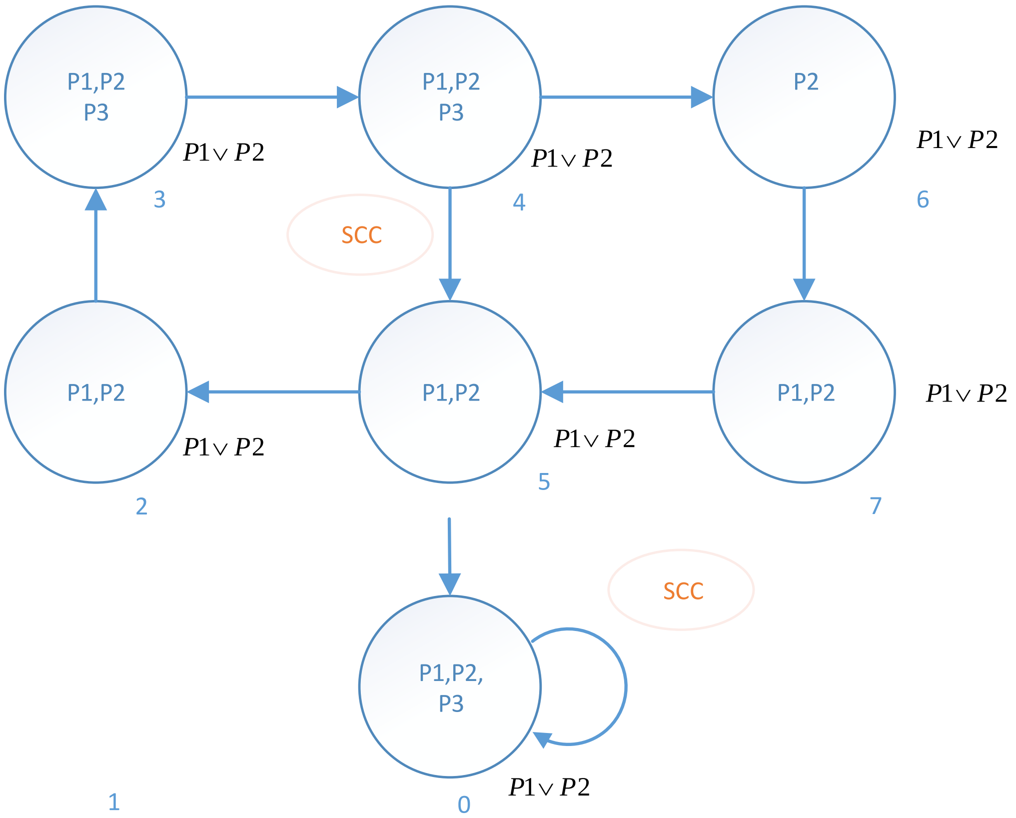
Q1:

1. Answer:

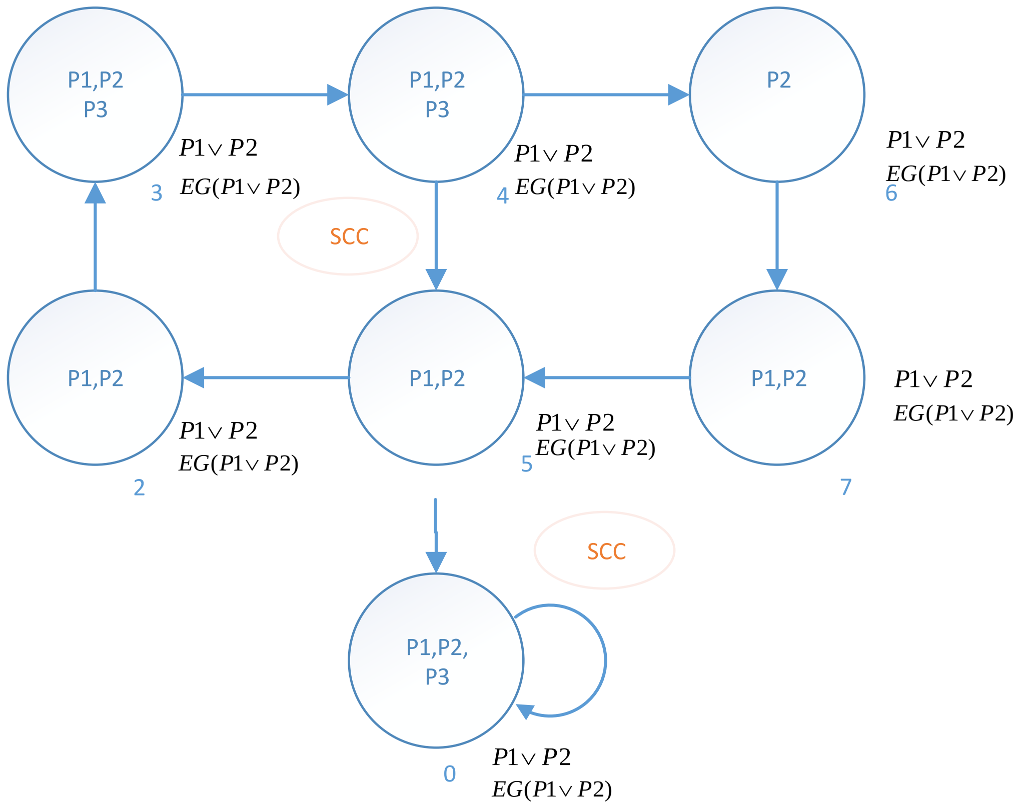
AF (f) = ¬EG (¬f) $\Rightarrow$ AF (¬P1 ∧ ¬P2) = ¬EG (¬ (¬P1 ∧ ¬P2)) = ¬EG (P1 ∧ P2)

The following diagrams are the steps to check ¬EG (P1 ∧ P2)

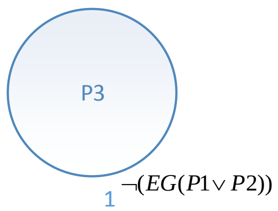
Step1: check P1 ∧ P2



Step 2:check EG(P1 ∧ P2)



Step 3: check ¬EG(P1∧P2)



Therefore, the initial state does not satisfy the requirement AF(¬P1∧¬P2)(¬EG(P1∧P2)). It means that this model cannot meet requirement AF(¬P1∧¬P2).

2.

(1).To any to different processes, the state of them will not be “critical” at the same time.

$$\forall i, j : n \bullet i != j \ \ imply\ AG(\neg(critical(i) \wedge critical(j)))$$

(2)To any process, if request happens, critical will always eventually happen

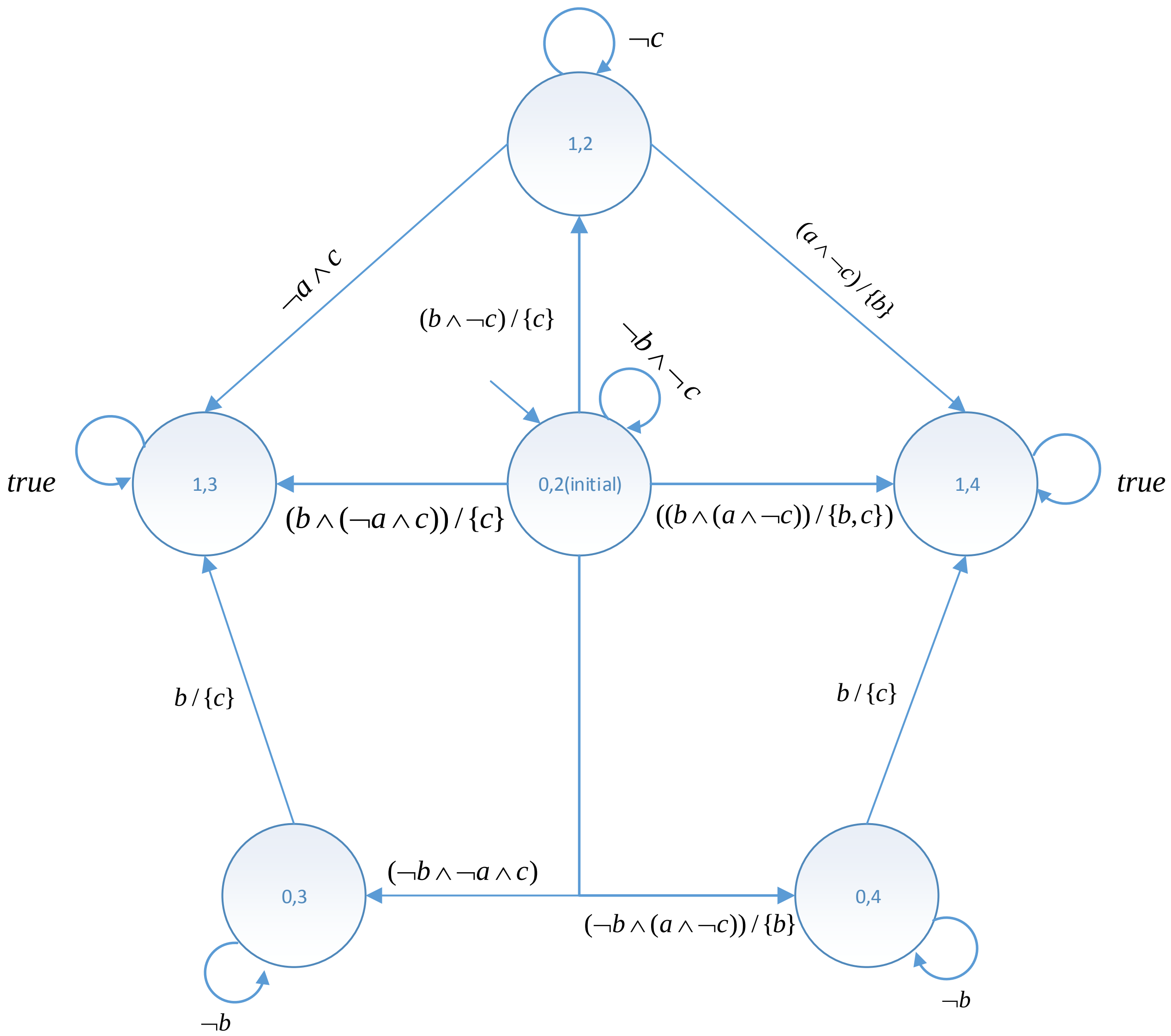
$$\forall i : n \bullet AG(request(i) \ \Rightarrow AF(critical(i))))$$

(3) The system is deadlock free

$$\forall i : n \bullet \text{AG EX true}$$

Q2

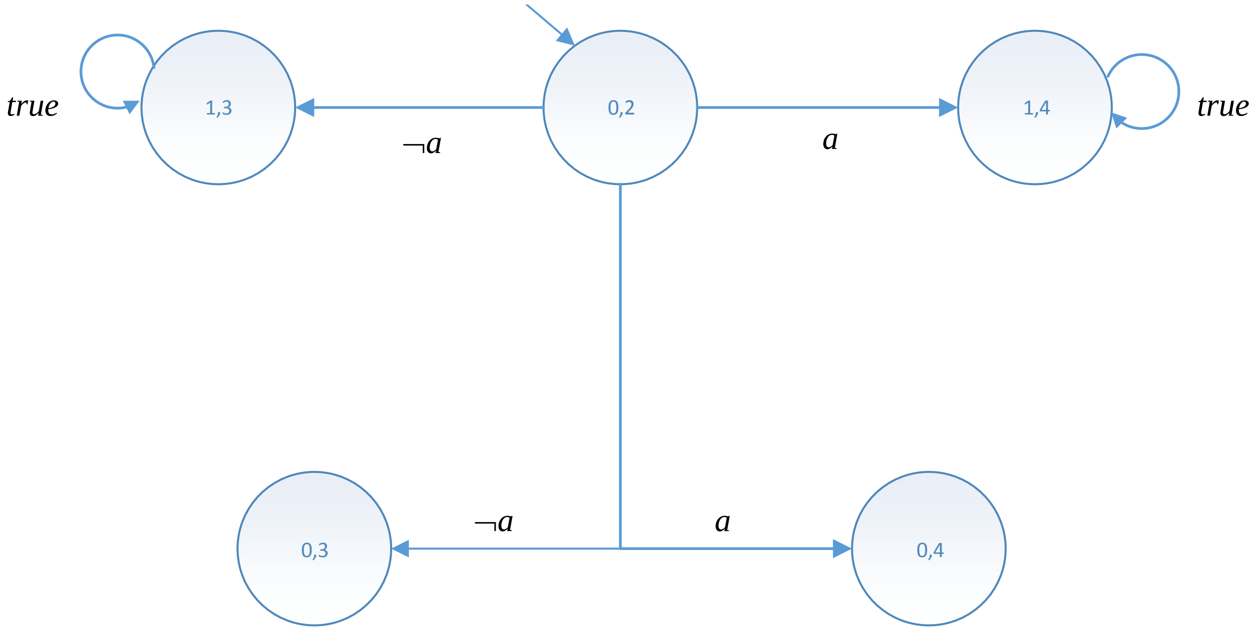
1. $A_1 \| A_2$



Deterministic: No, considering the input ($a \wedge \neg b \wedge \neg c$) to the initial state, states (0,4) and (0,2) are all possible next state. So it is not deterministic.

Reactive: No, considering the input ($a \wedge b \wedge c$) to the initial state, it will not react. So it is not reactive.

2. $A_1 \,||\, A_2 \setminus \{b,c\}$



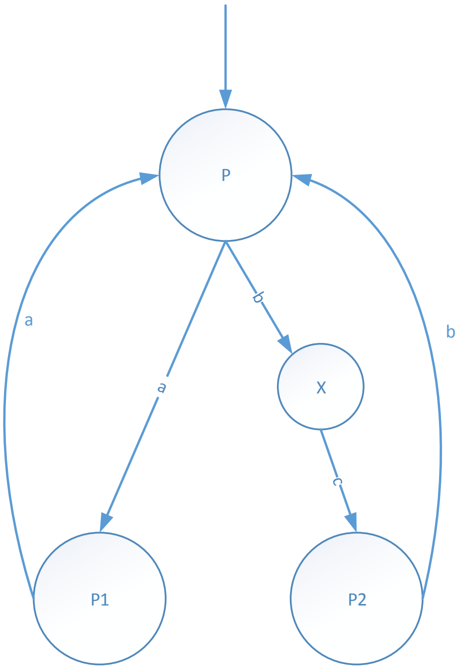
Deterministic: No, considering the input ($\neg a$) to the initial state, states (1,3) and (0,3) are all possible next state. So it is not deterministic.

Reactive: No, considering state (0, 4), it will not react any action. So it is not reactive.

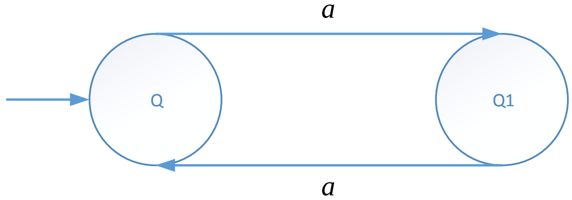
Q3

P and Q can be drawn like below(X is a middle state):

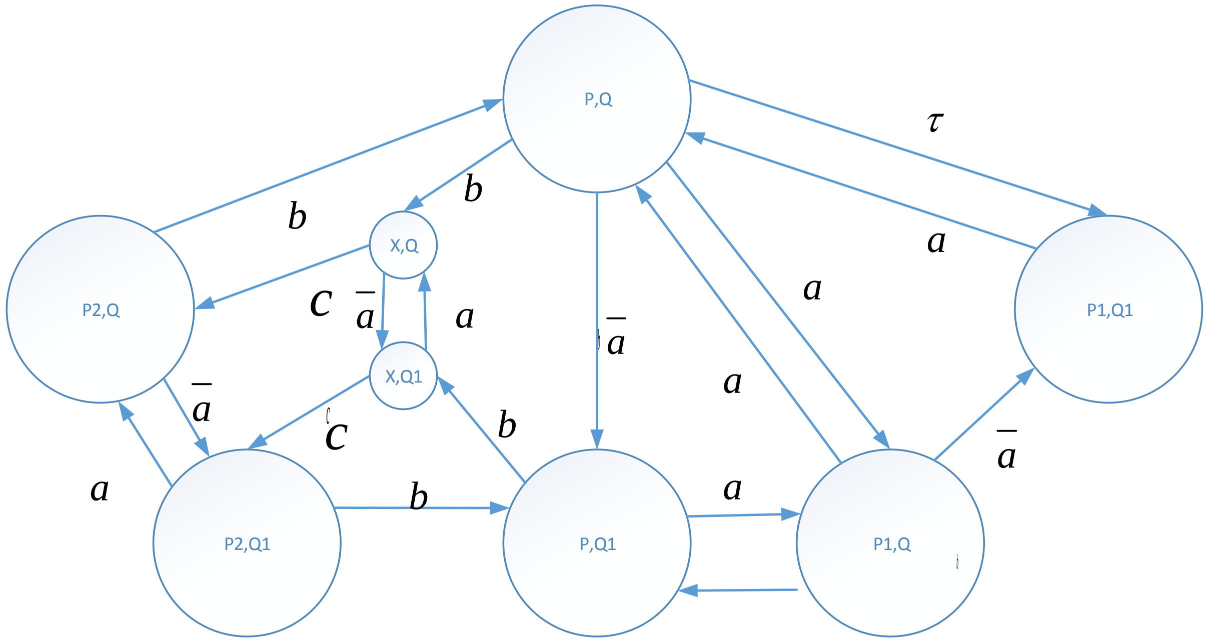
P.



Q.



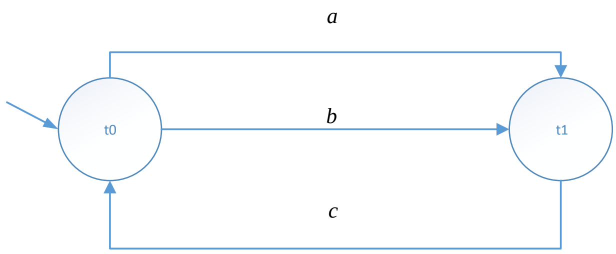
P || Q can be drawn like below:



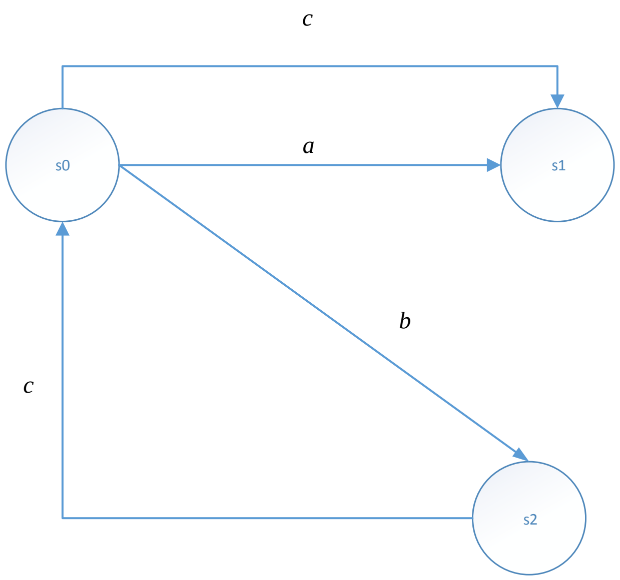
Q4

We can define 2 processes T,S as below:

T



S



(1) Bisimulation

We can use the algorithm to check Bisimulation.

We need to compute the greatest fixpoint (gfp): we do it by computing approximates starting from the Cartesian product:

- R0={(t0,s0), (t0,s1), (t0,s2), (t1,s0), (t1,s1),(t1,s2)} – Cartesian product
- R1={(t0,s0),(t1,s0),(t1,s1),(t1,s2)} – removed those pairs that violate local condition on final (if T final then S final)
- R2={(t0,s0),(t1,s1),(t1,s2)} – removed (t1,s0), since t1 can do c and s0 cannot
- R3={(t0,s0),(t1,s1),(t1,s2)} – nothing is further removed
- R2=R3

greatest fixpoint reached, S and T are bisimilar

(2) Simulation

We can use the algorithm to check Simulation.

We need to compute the greatest fixpoint (gfp): we do it by computing approximates starting from the Cartesian product:

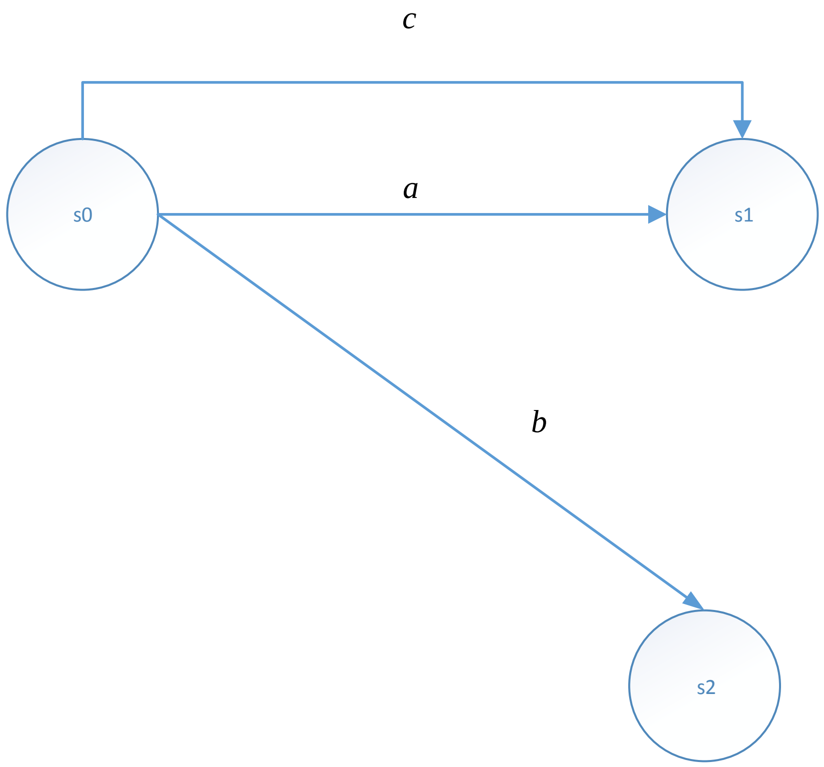
- R0={(t0,s0), (t0,s1), (t0,s2), (t1,s0), (t1,s1),(t1,s2)} – Cartesian product
- R1={(t0,s0),(t1,s0),(t1,s1),(t1,s2)} – removed those pairs that violate local condition on final (if T final then S final)
- R2={(t0,s0),(t1,s1),(t1,s2)} – removed (t1,s0), since t1 can do c and s0 cannot
- R3={(t0,s0),(t1,s1),(t1,s2)} – nothing is further removed

• R2=R3

Fixed point reached!

S simulates T and similarly vice versa

However if S performs like below:



(1). Bisimulation

As the algorithm, we can see that S and T are not Bisimilar.

(2). Simulation

We can use the algorithm to check Simulation.

We need to compute the greatest fixpoint: we do it by computing approximates starting from the Cartesian product:

• R0= {(s0,t0), (s0,t1), (s1,t0), (s1,t1), (s2,t0),(s2,t1)} – Cartesian product

• R1= {(s0,t0), (s1,t0), (s1,t1), (s2,t0),(s2,t1)} – removed those pairs that violate local condition on final (if S final then T final)

• R2= {(s0,t0), (s1,t1), (s2,t0), (s2,t1)} – removed (s1,t0), since s1 can do c but t0 cannot; removed (t1,s2) since t1 can do c but s2 cannot

• R3= {(s0,t0), (s1,t1), (s2,t0), (s2,t1)} – nothing is removed R2=R3

greatest fixpoint reached, so T simulates S.

However, as we can see S2 cannot accept C action, so S does not simulates T.