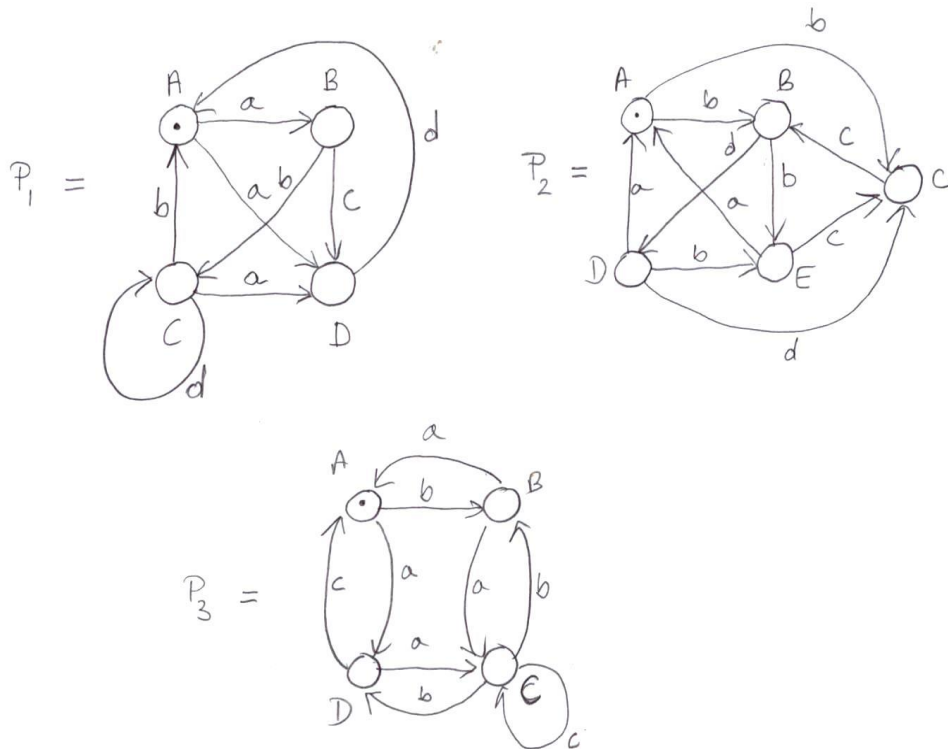


### CS 2SD3. Sample solutions to the assignment 1.

Total of this assignment is 166 pts. 100% = 160 pts. Each assignment is worth 10% of total. Most of solutions are not unique.

If you think your solution has been marked wrongly, write a short memo stating where marking in wrong and what you think is right, and resubmit to me during class, office hours, or just slip under the door to my office. The deadline for a complaint is 2 weeks after the assignment is marked and returned.

- 1.[15] a.[9] For each one of the following three processes, give the Finite State Processes (FSP) description of the labelled transition graph. Dots indicate initial states.
- b.[6] Use LTSA to transform the solutions to 1.a back into labelled transition systems. Compare the results and discuss differences (if any).



Solutions (not unique!):

a.

[3]  $P_1 = A$

$A = (a \rightarrow B \mid a \rightarrow D)$

$B = (b \rightarrow C \mid c \rightarrow D)$

$C = (d \rightarrow C \mid a \rightarrow D \mid b \rightarrow A)$

$D = (d \rightarrow A)$

P1 = A

A = (a → B | a → (d → A))

B = (b → C | c → (d → A))

C = (d → C | a → (d → A) | b → A)

[3] P2 = A

A = (b → B | b → C)

B = (b → E | d → D)

C = (c → B)

D = (a → A | b → E | d → C)

E = (a → A | c → C)

P2 = A

A = (b → B | b → (c → B))

B = (b → E | d → D)

D = (b → E | d → (c → B))

E = (a → A | c → (c → B))

[3] P3 = A

A = (a → D | b → B)

B = (a → A | a → C)

C = (b → B | b → D | c → C)

D = (a → C | c → A)

P3 = A

A = (a → (a → C | c → A) | b → B)

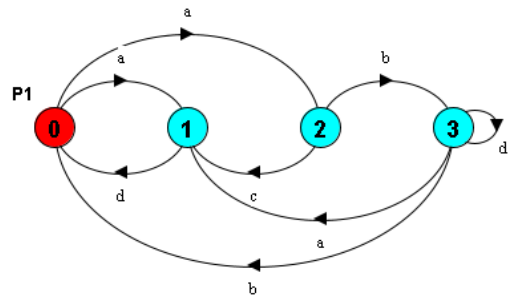
B = (a → A | a → C)

C = (b → B | b → (a → C | c → A) | c → C)

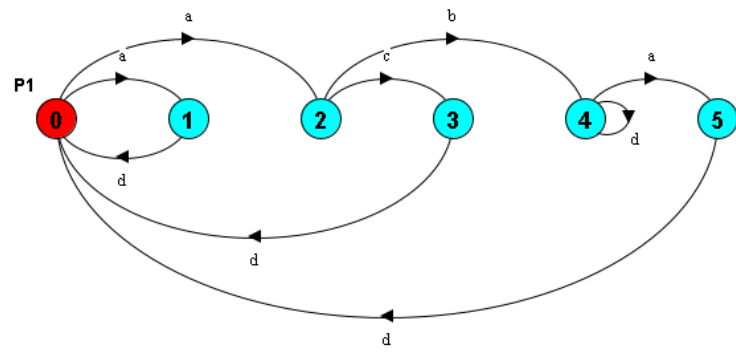
b.

[2]

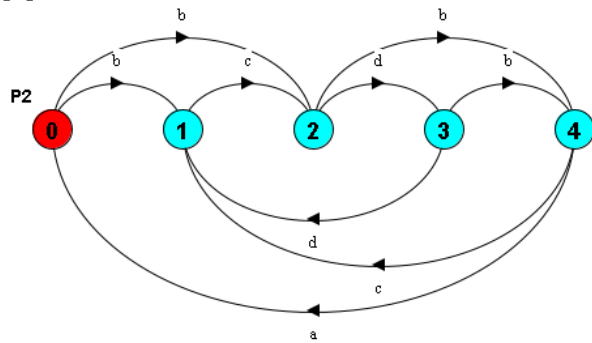
P1 = A,  
A = (a → B | a → D),  
B = (b → C | c → D),  
C = (d → C | a → D | b → A),  
D = (d → A).



$P1 = A,$   
 $A = (a \rightarrow B \mid a \rightarrow (d \rightarrow A)),$   
 $B = (b \rightarrow C \mid c \rightarrow (d \rightarrow A)),$   
 $C = (d \rightarrow C \mid a \rightarrow (d \rightarrow A)).$

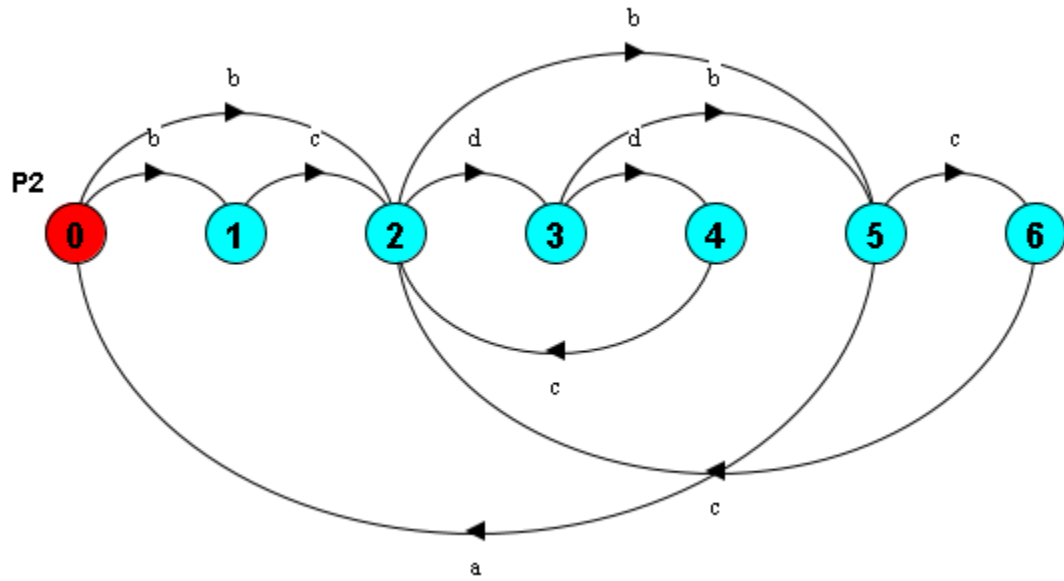


[2]



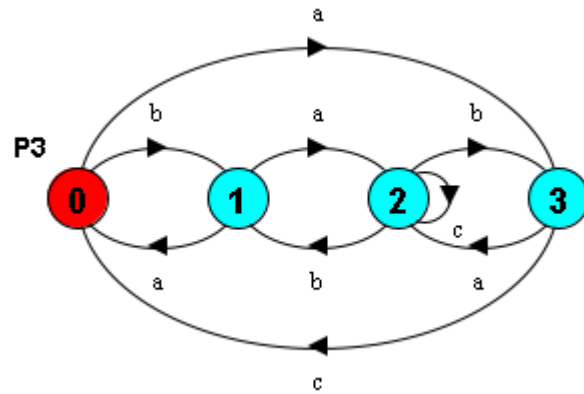
$P2 = A,$   
 $A = (b \rightarrow B \mid b \rightarrow C),$   
 $B = (b \rightarrow E \mid d \rightarrow D),$   
 $C = (c \rightarrow B),$   
 $D = (b \rightarrow E \mid d \rightarrow C),$   
 $E = (a \rightarrow A \mid c \rightarrow C).$

$P2 = A,$   
 $A = (b \rightarrow B \mid b \rightarrow (c \rightarrow B)),$   
 $B = (b \rightarrow E \mid d \rightarrow D),$   
 $D = (b \rightarrow E \mid d \rightarrow (c \rightarrow B)),$   
 $E = (a \rightarrow A \mid c \rightarrow (c \rightarrow B)).$



[2]

$P3 = A,$   
 $A = (a \rightarrow D \mid b \rightarrow B),$   
 $B = (a \rightarrow A \mid a \rightarrow C),$   
 $C = (b \rightarrow B \mid b \rightarrow D \mid c \rightarrow C),$   
 $D = (a \rightarrow C \mid c \rightarrow A).$

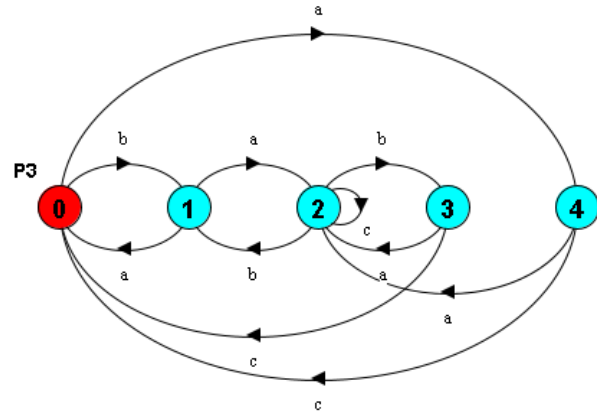


$P3 = A,$

$A = (a \rightarrow (a \rightarrow C \mid c \rightarrow A) \mid b \rightarrow B),$

$B = (a \rightarrow A \mid a \rightarrow C),$

$C = (b \rightarrow B \mid b \rightarrow (a \rightarrow C \mid c \rightarrow A) \mid c \rightarrow C).$



2.[10] Consider the following simple hotel reservation system.

A customer makes a room request.

If room is available, a confirmation is sent to the customer,

otherwise the customer is put on a reservation list. If a room is confirmed, the customer may either use it, pay for the room, leave and the whole transaction is archived. However the customer may also cancel his/her reservation. When the customer is on waiting list, a room may become available, and then a confirmation is sent to a customer. The customer may also give up waiting and cancel his/her request.

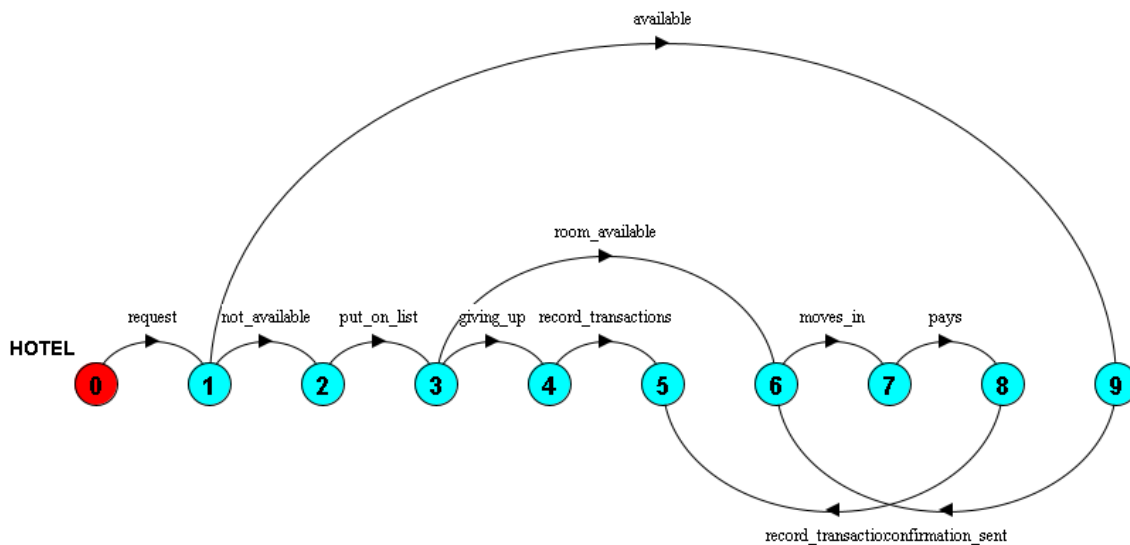
Model this reservation system as a FSP process reservation. Note that this process always stops, so you have to use the process STOP. Also provide appropriate labelled transition system (use LTSA)

Solution:

[8]

```
HOTEL = (request -> REQUESTED),
REQUESTED = (available -> confirmation_sent -> CONFIRMED
| not_available -> put_on_list -> ON_WAITING_LIST),
CONFIRMED = (moves_in -> pays -> ARCHIVED),
ON_WAITING_LIST = (room_available -> CONFIRMED | giving_up -> CANCELED),
ARCHIVED = (record_transactions -> STOP),
CANCELED = (record_transactions -> STOP).
```

[2]



- 3.[10] A miniature portable FM radio has three controls. An on/off switch turns the device on and off. Tuning is controlled by two buttons **scan** and **reset** which operate as follows. When the radio is turned on or **reset** is pressed, the radio is tuned to the top frequency of the FM band (108 MHz). When **scan** is pressed, the radio scans towards the bottom of the band (88 MHz). It stop scanning when it **locks** onto a station or it reaches the bottom (**end**). If the radio is currently tuned to a station and scan is pressed then it start to **scan** from the frequency of that station towards the bottom. Similarly, when reset is pressed the receiver tunes to the top. Model the radio as a *FSP* process **RADIO**. Also provide an appropriate labelled transition system.  
Hint: The alphabet of **RADIO** is {on, off, scan, reset, lock, end}.

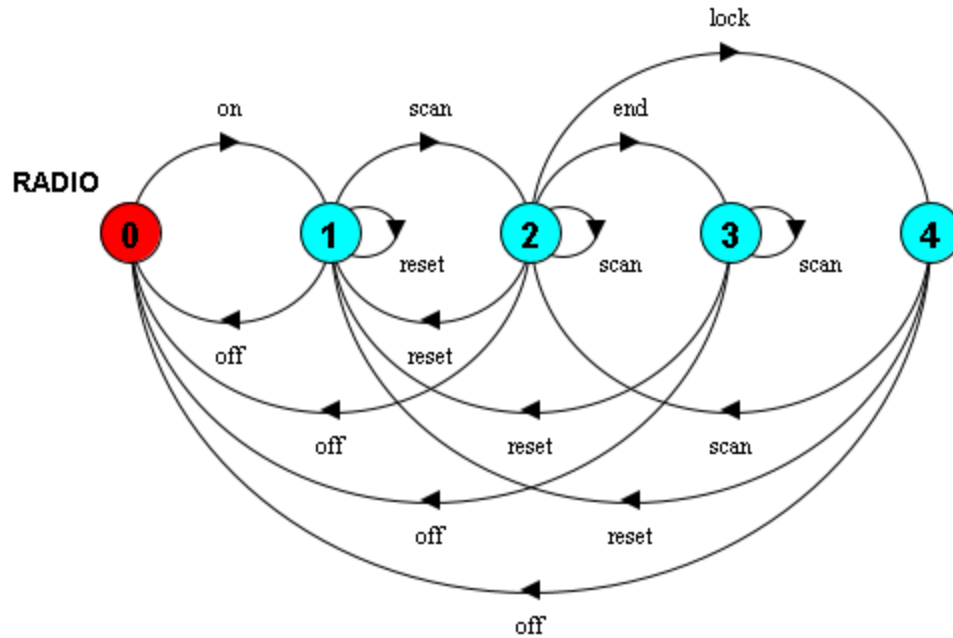
Solution:

[8]

FSP:

```
/* FM radio */  
  
RADIO      = OFF,  
OFF        = (on -> TOP),  
TOP        = (scan -> SCANNING | reset -> TOP | off -> OFF),  
SCANNING   = (scan -> SCANNING | reset -> TOP | off -> OFF | lock -> TUNED | end ->  
BOTTOM),  
TUNED      = (scan -> SCANNING | reset -> TOP | off -> OFF),  
BOTTOM     = (scan -> BOTTOM | reset -> TOP | off -> OFF).
```

[2]



4.[15] Program the radio of Question 3 in Java, complete with graphic display (if you can).

Java solutions will not be posted.

5.[15] A drinks dispensing machine charges 15c for can of Sugerola, 20c for a can of SugerolaDiet and 25c for a can of SugerolaSuperDiet . The machine accepts coins with denominations 5c, 10c and 25c and gives changes. Model the machine as an *FSP* process, DRINKS.

Solution (not unique):

[12]

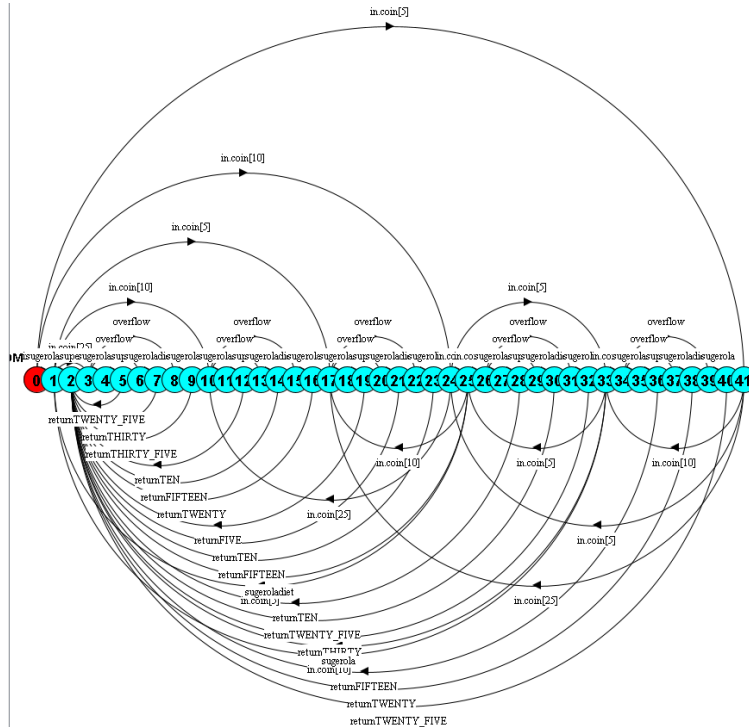


```

DM = (in.coin [5] -> FIVE | in.coin [10] -> TEN | in.coin [25] -> TWENTY_FIVE),
FIVE = (in.coin [5] -> TEN | in.coin [10] -> FIFTEEN | in.coin [25] -> THIRTY),
TEN = (in.coin [5] -> FIFTEEN | in.coin [10] -> TWENTY | in.coin [25] -> THIRTY_FIVE),
FIFTEEN = (sugerola -> STOP | in.coin [5] -> TWENTY | in.coin [10] -> TWENTY_FIVE | in.coin [25] -> FORTY),
TWENTY = (sugeroladiet -> STOP | in.coin [5] -> TWENTY_FIVE | in.coin [10] -> THIRTY | in.coin [25] -> FORTY_FIVE),
TWENTY_FIVE = (sugerolasuperdiet -> STOP | in.coin [5] -> THIRTY | in.coin [10] -> THIRTY_FIVE | in.coin [25] -> FIFTY),
THIRTY = (overflow -> sugerola -> returnFIFTEEN -> STOP
          | overflow -> sugeroladiet -> returnTEN -> STOP
          | overflow -> sugerolasuperdiet -> returnFIVE -> STOP),
THIRTY_FIVE = (overflow -> sugerola -> returnTWENTY -> STOP
               | overflow -> sugeroladiet -> returnFIFTEEN -> STOP
               | overflow -> sugerolasuperdiet -> returnTEN -> STOP),
FORTY = (overflow -> sugerola -> returnTWENTY_FIVE -> STOP
         | overflow -> sugeroladiet -> returnTWENTY -> STOP
         | overflow -> sugerolasuperdiet -> returnFIFTEEN -> STOP),
FORTY_FIVE = (overflow -> sugerola -> returnTHIRTY -> STOP
              | overflow -> sugeroladiet -> returnTWENTY_FIVE -> STOP
              | overflow -> sugerolasuperdiet -> returnTEN -> STOP),
FIFTY = (overflow -> sugerola -> returnTHIRTY_FIVE -> STOP
         | overflow -> sugeroladiet -> returnTHIRTY -> STOP
         | overflow -> sugerolasuperdiet -> returnTWENTY_FIVE -> STOP).

```

[3]



6.[15] Consider the following set of FSPs:

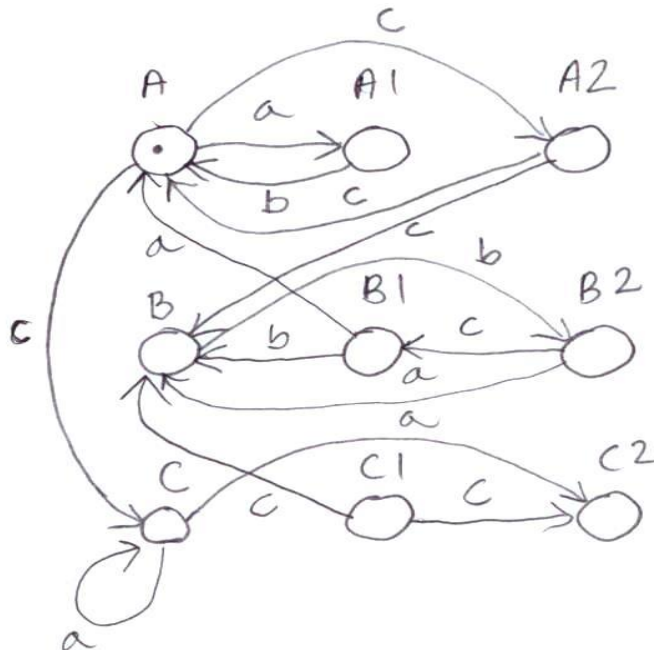
$A = ((a \rightarrow (b \rightarrow A)) \mid (c \rightarrow (a \rightarrow C \mid c \rightarrow B)) \mid c \rightarrow C)$   
 $B = (b \rightarrow (a \rightarrow B \mid c \rightarrow (a \rightarrow A \mid b \rightarrow B)))$   
 $C = ((a \rightarrow (b \rightarrow (c \rightarrow B))) \mid (a \rightarrow C))$

- a.[12] Construct an equivalent Labelled Transition System using the rules from page 16 of Lecture Notes 2.
- b.[3] Use LTSA to derive appropriate LTS, and, if different than yours, analyse and explain differences.

Solution:

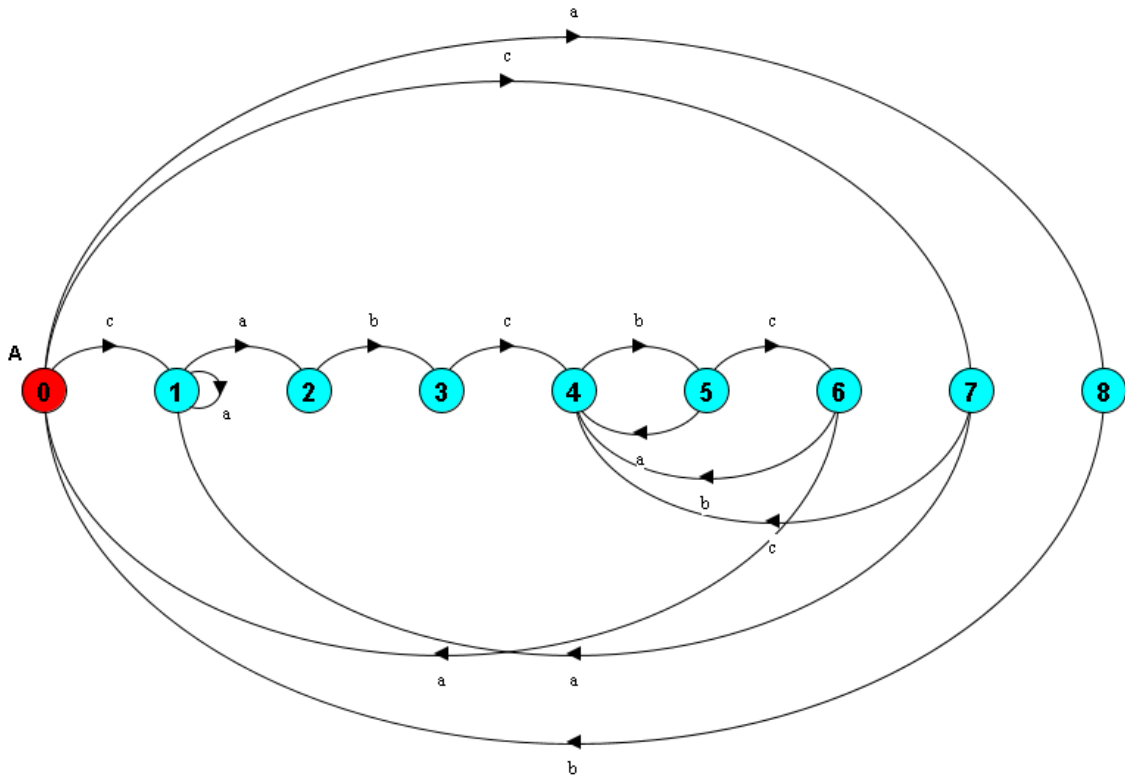
a.

$A = (a \rightarrow A1 \mid c \rightarrow A2 \mid c \rightarrow C)$   
 $A1 = (b \rightarrow A)$   
 $A2 = (a \rightarrow C \mid c \rightarrow B)$   
 $B = (b \rightarrow B2)$   
 $B1 = (a \rightarrow A \mid b \rightarrow B)$   
 $B2 = (a \rightarrow B \mid c \rightarrow B1)$   
 $C = (a \rightarrow C2 \mid a \rightarrow C)$   
 $C1 = (c \rightarrow B)$   
 $C2 = (b \rightarrow C1)$



b.

$A = (a \rightarrow A1 \mid c \rightarrow A2 \mid c \rightarrow C),$   
 $A1 = (b \rightarrow A),$   
 $A2 = (a \rightarrow C \mid c \rightarrow B),$   
 $B = (b \rightarrow B2),$   
 $B1 = (a \rightarrow A \mid b \rightarrow B),$   
 $B2 = (a \rightarrow B \mid c \rightarrow B1),$   
 $C = (a \rightarrow C2 \mid a \rightarrow C),$   
 $C1 = (c \rightarrow B),$   
 $C2 = (b \rightarrow C1).$



7.[18] a.[8] Show that processes  $\parallel S1$  and  $S2$  generate the same Labelled Transition Systems, i.e.  $LTS(\parallel S1) = LTS(S2)$  (or equivalently, they generate the same behaviour)

$P = (a \rightarrow b \rightarrow d \rightarrow P)$

$Q = (c \rightarrow b \rightarrow e \rightarrow Q)$

$\parallel S1 = (P \parallel Q)$

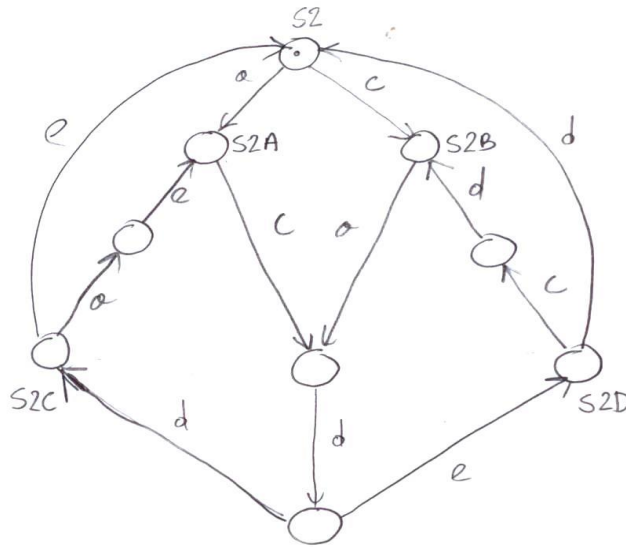
$S2 = (a \rightarrow S2A \mid c \rightarrow S2B)$

$S2A = (c \rightarrow b \rightarrow d \rightarrow S2C \mid c \rightarrow b \rightarrow e \rightarrow S2D)$   
 $S2B = (a \rightarrow b \rightarrow d \rightarrow S2C \mid a \rightarrow b \rightarrow e \rightarrow S2D)$   
 $S2C = (e \rightarrow S2 \mid a \rightarrow e \rightarrow S2A)$   
 $S2D = (d \rightarrow S2 \mid c \rightarrow d \rightarrow S2B)$

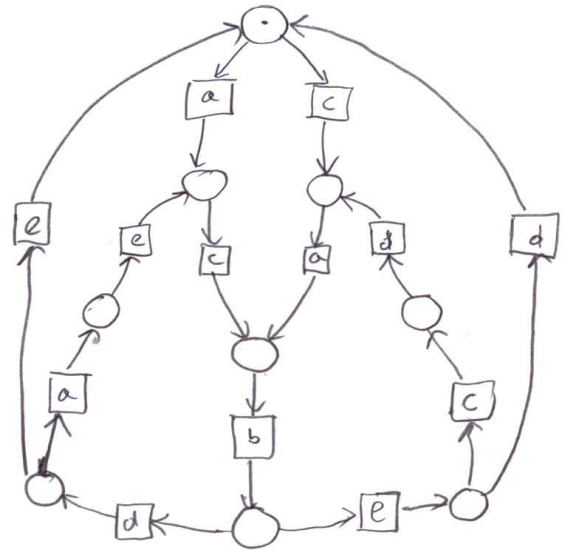
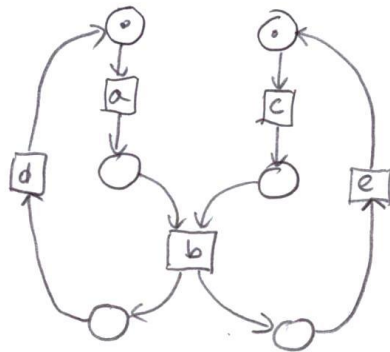
- b.[10] Using a method presented on page 17 of Lecture Notes 3 and pages 10-11 of Lecture Notes 4, transform the processes  $\parallel S1$  and  $S2$  into appropriate Petri nets. Are these nets identical? Explain the difference. Which one allows *simultaneity*?

Solution:

- a.  $LTS(\parallel S1) = LTS(S2)$  and is isomorphic to the below diagram:



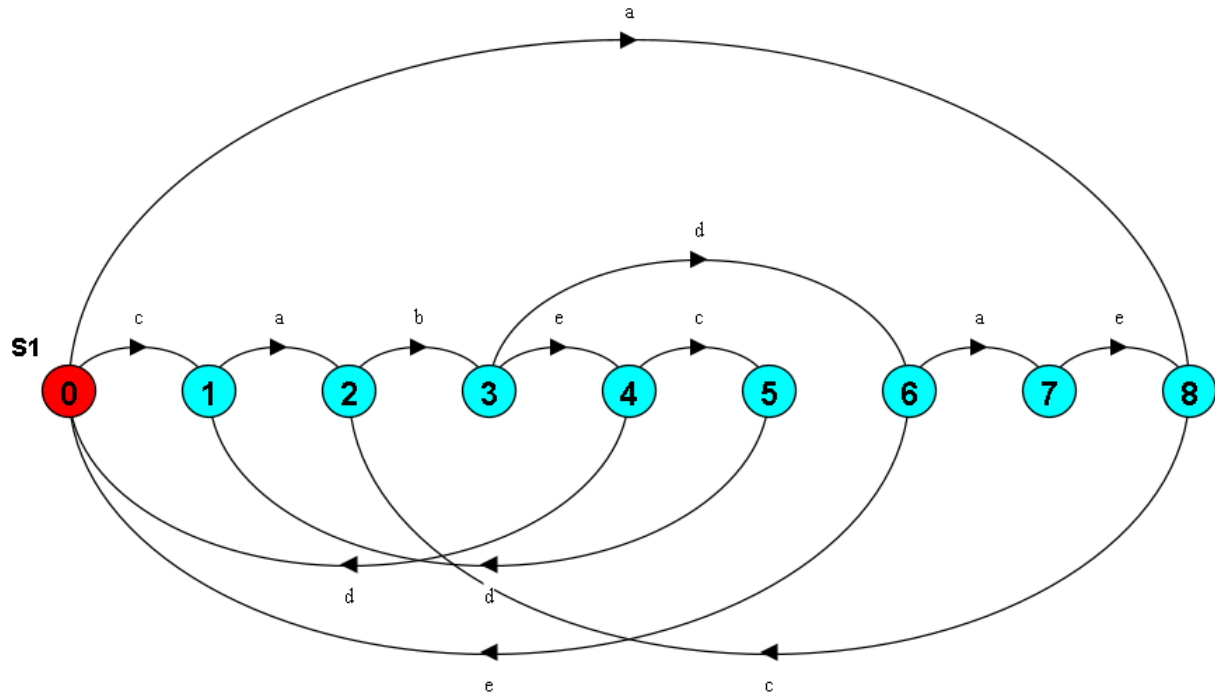
- b.  $\parallel S1$  corresponds to  $N1$  and  $S2$  corresponds to  $N2$ .



In N1, actions  $\{a, c\}$  and  $\{d, e\}$  can be executed simultaneously. N2 is isomorphic to  $LTS(\parallel S1) = LTS(S2)$ .

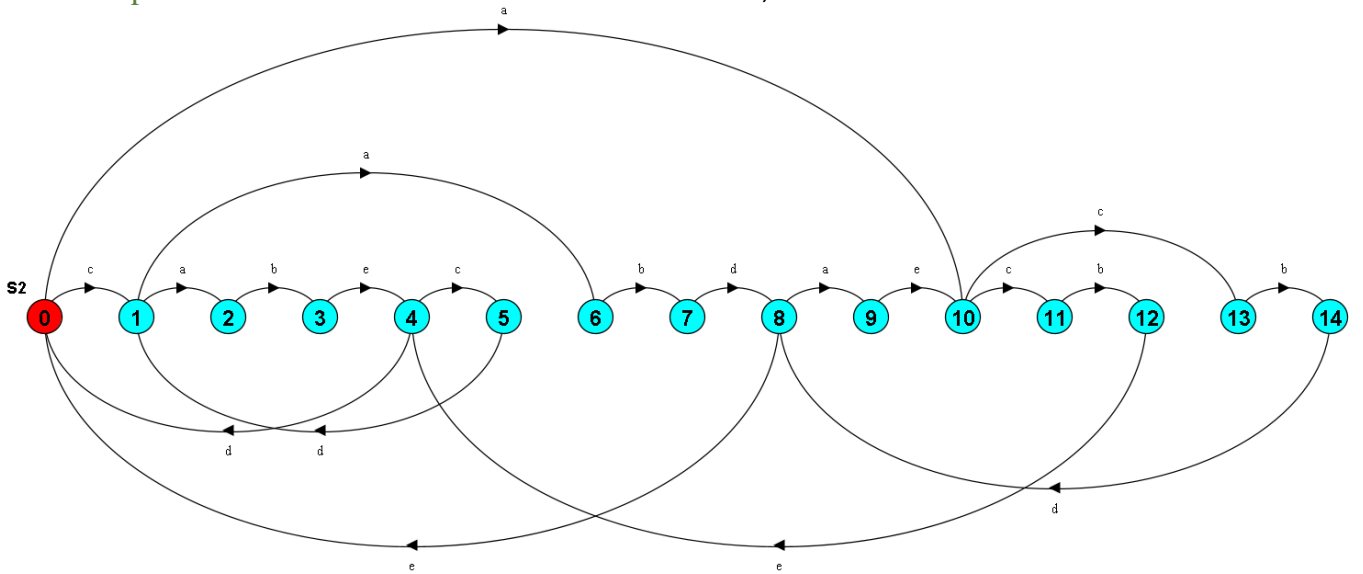
(Solution verification by LTSA tool)

$P = (a \rightarrow b \rightarrow d \rightarrow P).$   
 $Q = (c \rightarrow b \rightarrow e \rightarrow Q).$   
 $\parallel S1 = (P \parallel Q).$

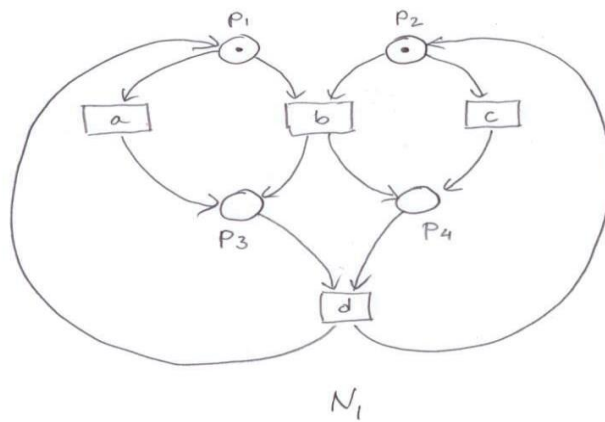


$S2 = (a \rightarrow S2A \mid c \rightarrow S2B),$   
 $S2A = (c \rightarrow b \rightarrow d \rightarrow S2C \mid c \rightarrow b \rightarrow e \rightarrow S2D),$   
 $S2B = (a \rightarrow b \rightarrow d \rightarrow S2C \mid a \rightarrow b \rightarrow e \rightarrow S2D),$   
 $S2C = (e \rightarrow S2 \mid a \rightarrow e \rightarrow S2A),$   
 $S2D = (d \rightarrow S2 \mid c \rightarrow d \rightarrow S2B).$

One example  $S1 = c \rightarrow a \rightarrow b \rightarrow e \rightarrow c \rightarrow d \rightarrow a \rightarrow b \rightarrow e \rightarrow d \rightarrow S1,$



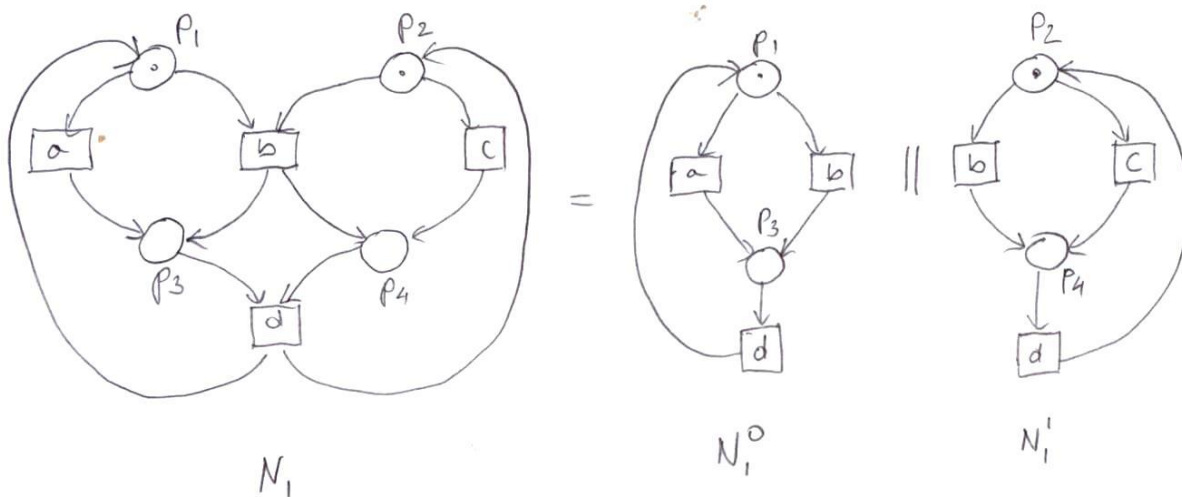
8.[10] Consider a Petri net below:



Model the net  $N_1$  as a composition of *FSP* processes.

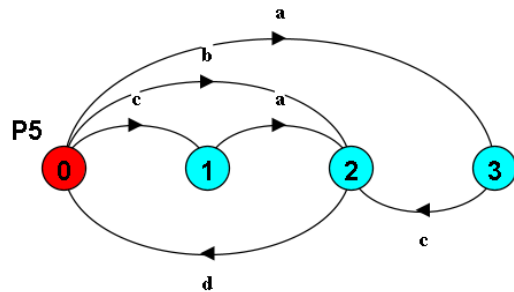
Solution:

Since we have:



a possible solution is:

$P1 = (a \rightarrow P3 \mid b \rightarrow P3),$   
 $P3 = (d \rightarrow P1).$   
 $P2 = (b \rightarrow P4 \mid c \rightarrow P4),$   
 $P4 = (d \rightarrow P2).$   
 $|| P5 = (P1 \mid P2).$

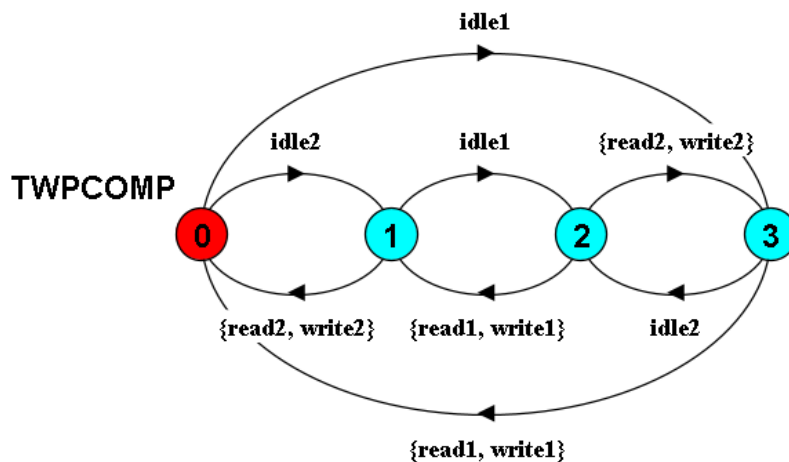


9.[10] Model the system from page 10 of Lecture Notes 3 as a composition of *FSP* processes. In this case, the entities that are represented by places in the Petri Nets model, must be represented by actions/transitions in *FSP* model.

Solution:

Solution: (a possible one, bonus for using labelling)

$COMP1 = (idle1 \rightarrow (read1 \rightarrow COMP1 \mid write1 \rightarrow COMP1)).$   
 $COMP2 = (idle2 \rightarrow (read2 \rightarrow COMP2 \mid write2 \rightarrow COMP2)).$   
 $MUT = (write1 \rightarrow MUT \mid write2 \rightarrow MUT).$   
 $|| TWPCOMP = (COMP1 \mid COMP2 \mid MUT).$

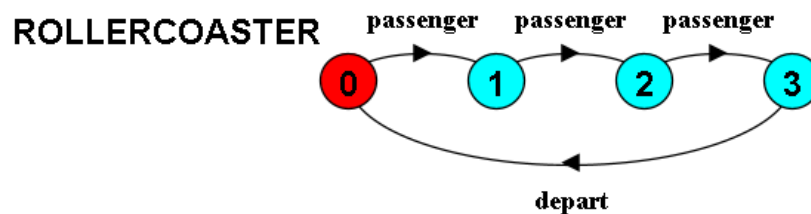




- 10.[10] A roller-coaster control system only permits its car to depart when it is full. Passengers arriving at the departure platform are registered with the roller-coaster controller by a turnstile. The controller signals the car to depart when there are enough passengers on the platform to fill the car to its maximum capacity of  $M$  passengers. Ignore the synchronization detail of passengers embarking from the platform and car departure. The roller-coaster consists of three processes: *TURNSTILE*, *CONTROL* and *CAR*. *TURNSTILE* and *CONTROL* interact by the shared action *passenger* indicating an arrival and *CONTROL* and *CAR* interact by the shared action *depart* signalling the car departure. Provide FSP description for each process and the overall composition.

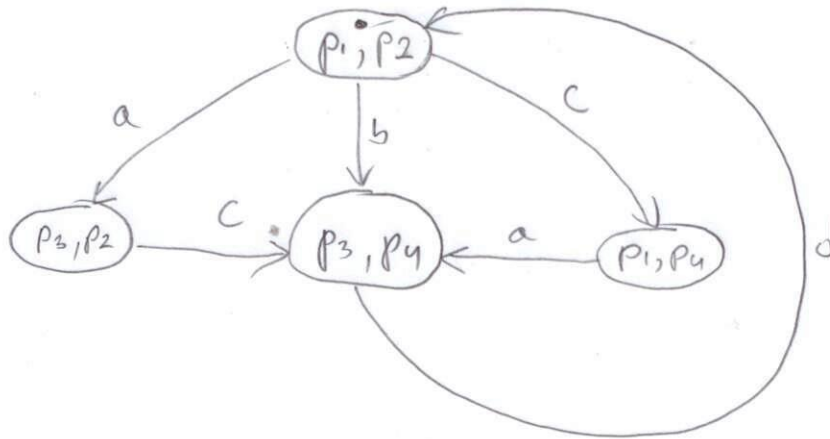
Solution (not unique):

```
const M = 3
TURNSTILE = (passenger -> TURNSTILE).
CONTROL   = CONTROL [0],
CONTROL [i:0..M] = (when (i<M) passenger -> CONTROL [i+1]
                    | when (i==M) depart -> CONTROL [0]).
CAR = (depart -> CAR).
|| ROLLERCOASTER = (TURNSTILE || CONTROL || CAR).
```

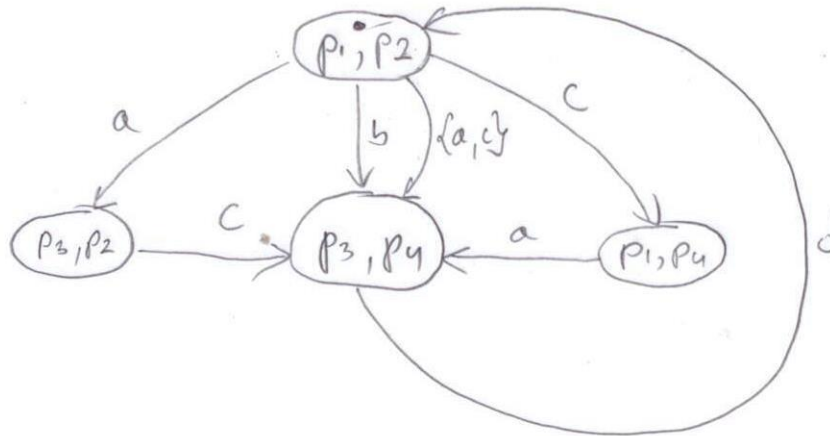


- 11.[10] Construct *reachability graph* (defined on page 18 of Lecture Notes 3) for the Petri net from Question 8.

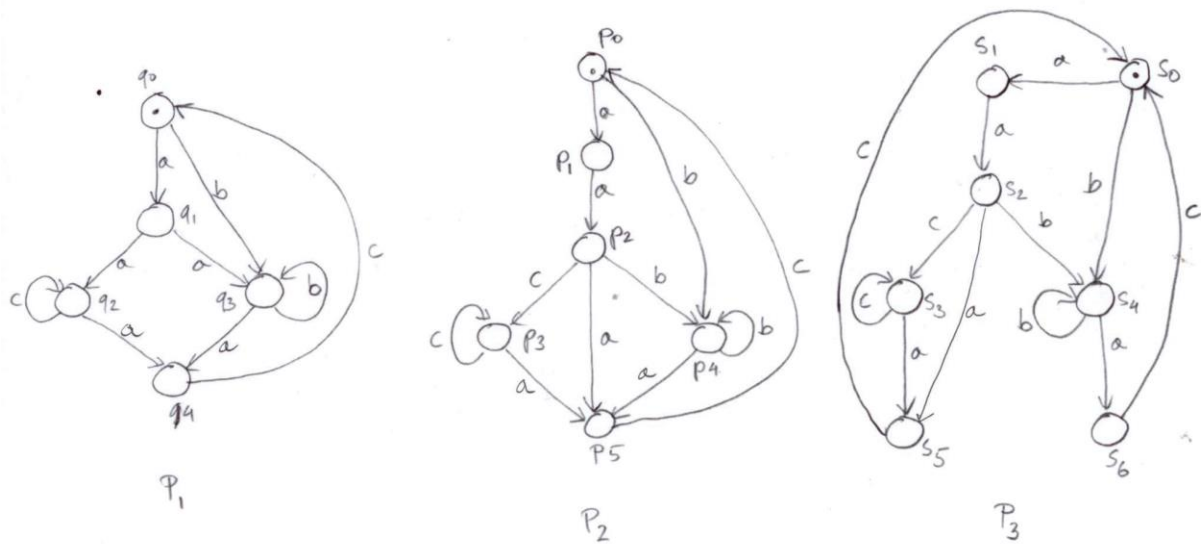
Solution: (7 if simultaneity is not mentioned)



or, if simultaneity is allowed:



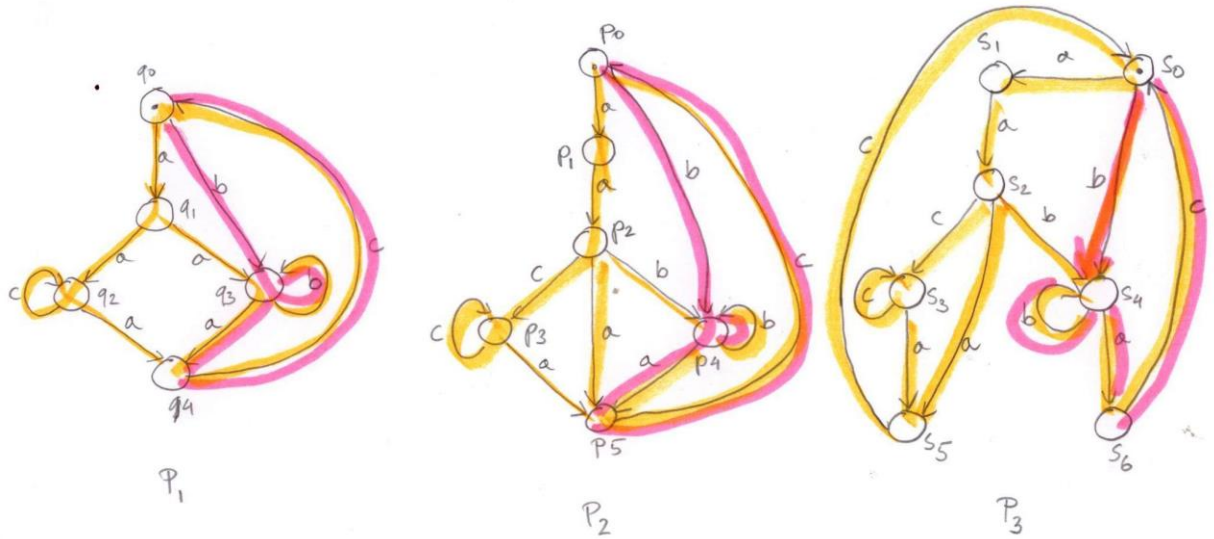
- 12.[28] Consider three Labelled Transition Systems (Finite State Machines, Finite Automata) given below:  $P_1$ ,  $P_2$  and  $P_3$ . Tokens represent initial states. Show that:
- a.[8]  $P_2 \approx P_3$ , i.e.  $P_2$  and  $P_3$  are *bisimilar*,
  - b.[6]  $P_1 \not\approx P_2$ , i.e.  $P_1$  and  $P_2$  are *not bisimilar*,
  - c.[6]  $P_1 \not\approx P_3$ , i.e.  $P_1$  and  $P_3$  are *not bisimilar*,
  - d.[8]  $\text{Traces}(P_1) = \text{Traces}(P_2) = \text{Traces}(P_3) = \text{Pref}(\text{give a proper regular expression})$ .



Solutions:

- a.  $p_0$  and  $s_0$  are bisimilar as in both cases actions  $a$  and  $b$  are allowed.  
 $p_1$  and  $s_1$  are bisimilar as they both allow only  $a$ .  
 $p_2$  and  $s_2$  allow  $a$ ,  $b$  and  $c$ , so they are bisimilar.  
 $p_3$  and  $s_3$  are bisimilar as they both allow  $a$  and  $c$ .  
 $p_4$  and  $s_4$  are bisimilar as they both allow  $a$  and  $b$ .  
 $p_5$  and  $s_5$  are bisimilar as they both allow only  $c$ , and  
 $p_5$  and  $s_6$  are also bisimilar as they also allow only  $c$ .  
We have exhausted all cases, so  $P_2$  and  $P_3$  are bisimilar, i.e.  $P_2 \approx P_3$ .
- b. After trace  $aa$  the labeled transition system  $P_1$  is either in the state  $q_2$  or the state  $q_3$ , while  $P_2$  is in the state  $p_2$ . In the state  $p_2$  the actions  $a$ ,  $b$  and  $c$  are allowed, in the state  $q_2$  the actions  $a$  and  $c$  are allowed, while in the state  $q_3$  the actions  $a$  and  $b$  are allowed. Hence both pairs  $(q_2, p_2)$  and  $(q_3, p_2)$  are *not* bisimilar, i.e.  $P_1 \not\approx P_2$ .
- c. After trace  $aa$  the labeled transition system  $P_1$  is either in the state  $q_2$  or the state  $q_3$ , while  $P_3$  is in the state  $s_2$ . In the state  $s_2$  the actions  $a$ ,  $b$  and  $c$  are allowed, in the state  $q_2$  the actions  $a$  and  $c$  are allowed, while in the state  $q_3$  the actions  $a$  and  $b$  are allowed. Hence both pairs  $(q_2, s_2)$  and  $(q_3, s_2)$  are *not* bisimilar, i.e.  $P_1 \not\approx P_3$ .

d.



Note that some parts are both yellow and red. In fact, only arrows  $q_0 \rightarrow q_3$ ,  $p_0 \rightarrow p_4$ ,  $s_0 \rightarrow s_4$ , labelled by  $b$ , are only red.

$\text{yellow}(P_1) = aa(c^* \cup b^*)ac$      $\text{yellow}(P_2) = aa(cc^*a \cup a \cup bb^*a)c = aa(c^* \cup b^*)ac$

$\text{yellow}(P_3) = aa((cc^*a \cup a)c \cup bb^*ac) = aa(c^* \cup b^*)ac$

$\text{yellow} = \text{yellow}(P_1) = \text{yellow}(P_2) = \text{yellow}(P_3)$

$\text{yellow} = aa(c^* \cup b^*)ac$

$\text{red} = bb^*ac$

$\text{cycle} = \text{from } q_0/p_0/s_0 \text{ to } q_0/p_0/s_0 = (\text{yellow} \cup \text{red})^*$

$\text{Traces} = \text{Pref}((\text{yellow} \cup \text{red})^*) = \text{Pref}((aa(c^* \cup b^*)ac) \cup bb^*ac)^*$