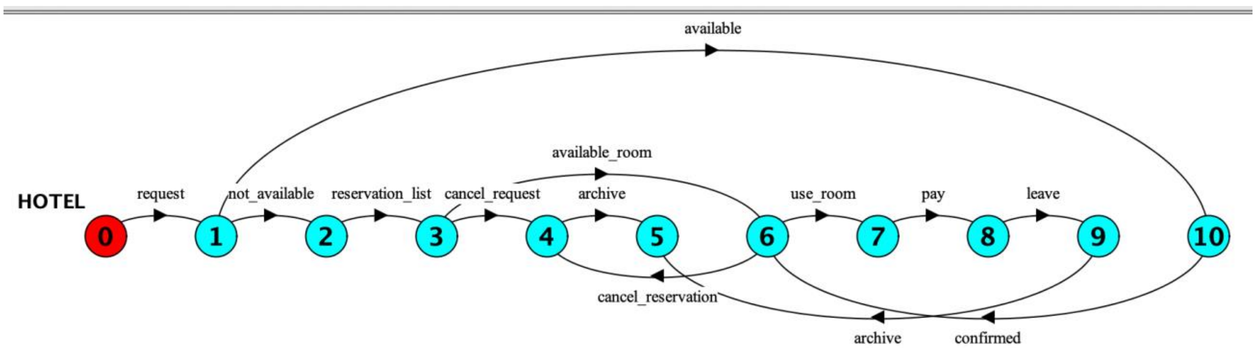
2SD3 Assignment 1

Rochan Muralitharan – muralr3

1.
```
HOTEL = (request -> REQUEST),
REQUEST = (available -> confirmed -> CONFIRMED
    | not_available -> reservation_list -> RESERVATION),
CONFIRMED = (use_room -> pay -> leave -> FINISHED
    | cancel_reservation -> CANCELLED),
RESERVATION = (available_room -> CONFIRMED
    | cancel_request -> CANCELLED),
FINISHED = (archive -> STOP),
CANCELLED = (archive -> STOP).
```



2. a)

P1 = A

A = (a -> B | a -> D)
B = (b -> C | c -> D)
C = (a -> D | b -> A | d -> C)
D = (d -> A)

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)

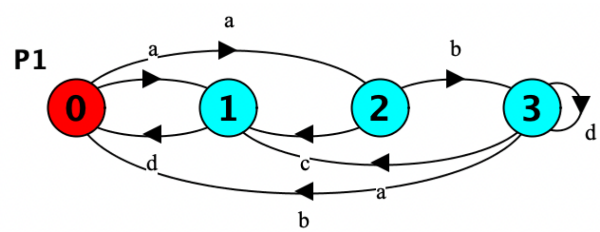
P3 = A

A = (a -> D| b -> B)

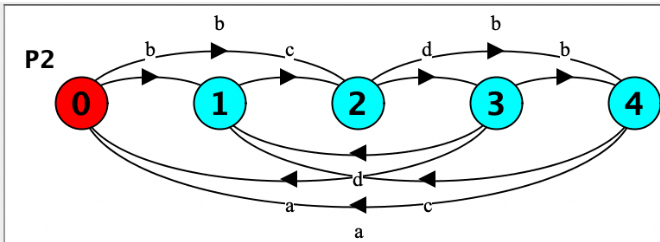
```
B = (a -> A | a -> C)
C = (b -> B | b -> D | c -> C)
D =  (a -> C | c -> A)
```

b)

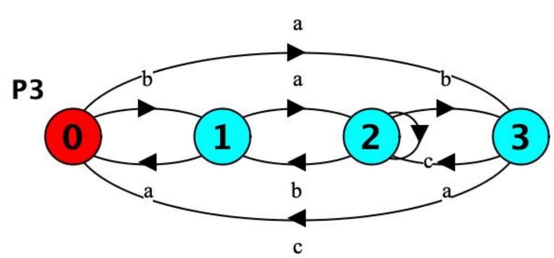
```
P1 = A,
A = (a -> B | a -> D),
B = (b -> C | c -> D),
C = (a -> D | b -> A | d -> C),
D =  (d -> A).
```




```
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).
```

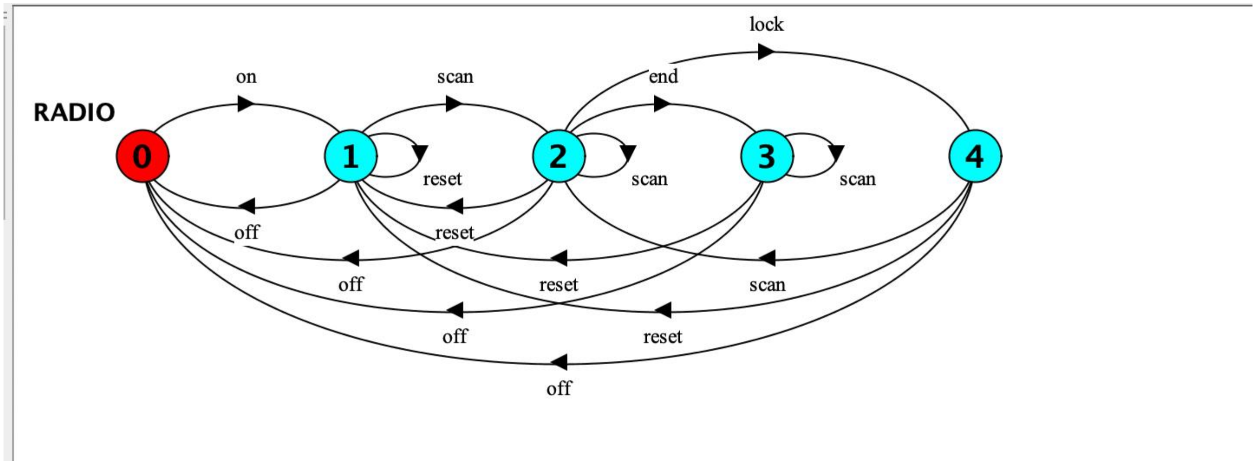



```
P3 = A,
A = (a -> D| b -> B),
B = (a -> A | a -> C),
C = (b -> B | b -> D | c -> C),
D =  (a -> C | c -> A).
```



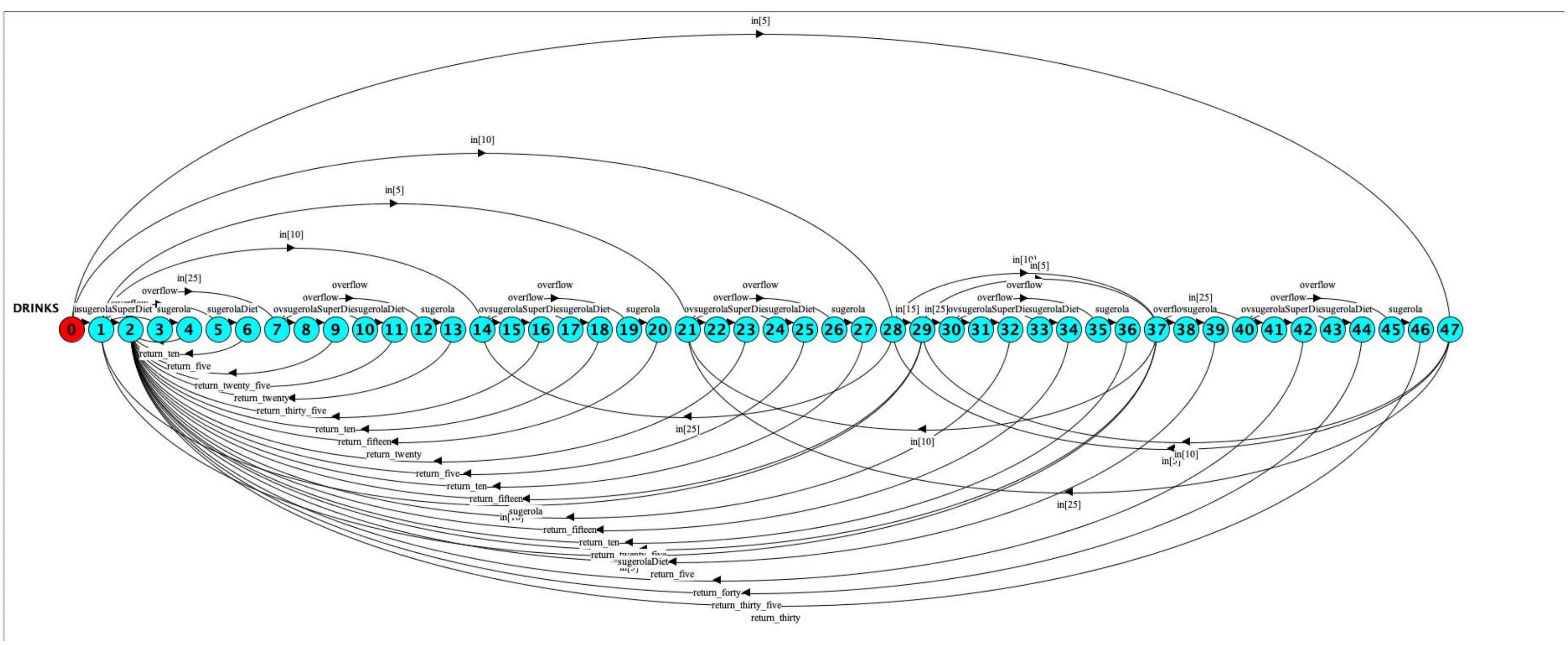
3.

```
RADIO = OFF,
OFF = (on -> TOP_FREQ),
TOP_FREQ = (scan -> SCANNING | reset -> TOP_FREQ | off -> OFF),
SCANNING = (reset -> TOP_FREQ | lock -> MIDDLE_FREQ | scan -> SCANNING |
    end -> BOTTOM_FREQ | off -> OFF),
MIDDLE_FREQ = (reset -> TOP_FREQ | scan -> SCANNING | off -> OFF),
BOTTOM_FREQ = (reset -> TOP_FREQ | scan -> BOTTOM_FREQ | off -> OFF).
```

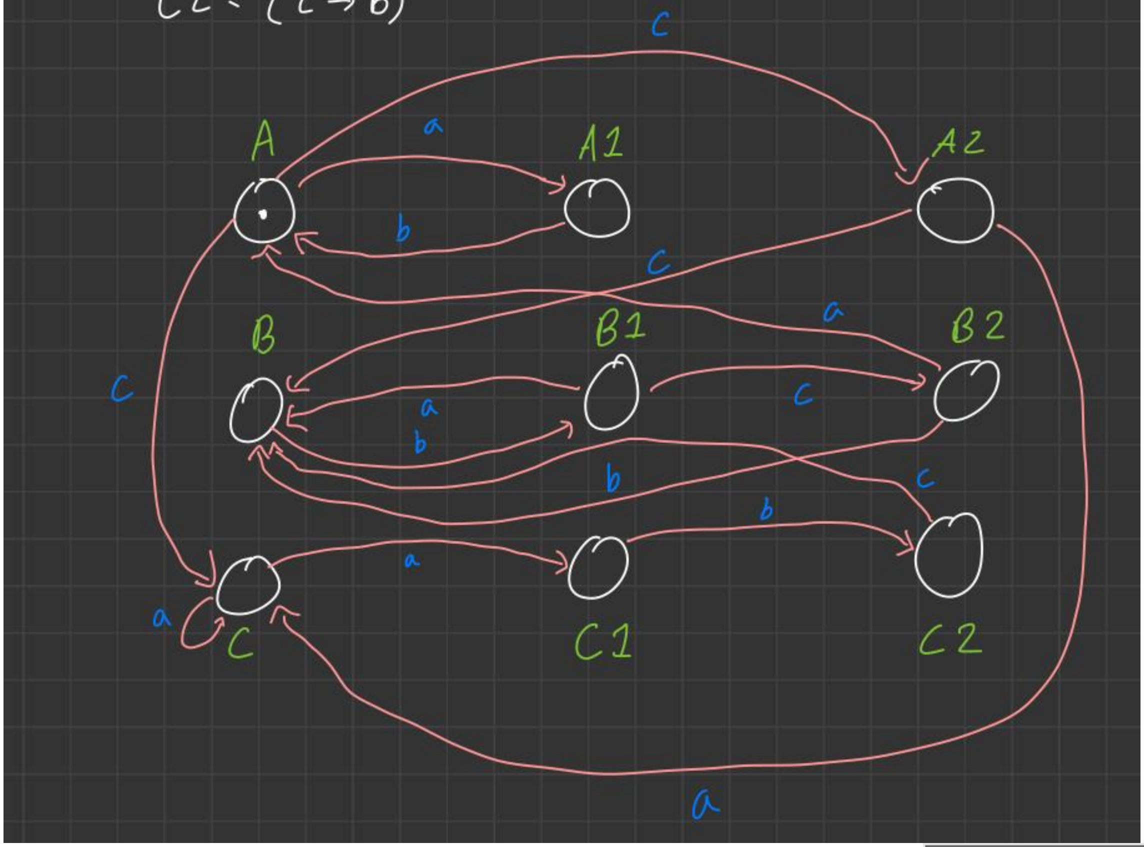


4. In Java File
5.

```
DRINKS = ZERO,
ZERO = (in[5] -> FIVE| in[10] -> TEN | in[25] -> TWENTY_FIVE),
FIVE = (in[5] -> TEN | in[10] -> FIFTEEN | in[25] -> THIRTY),
TEN = (in[15] -> FIFTEEN | in[10] -> TWENTY | in[25] -> THIRTY_FIVE),
FIFTEEN = (sugerola -> STOP | in[5] -> TWENTY |  in[10] -> TWENTY_FIVE | in[25] -> FORTY),
TWENTY = (sugerolaDiet -> STOP | overflow -> sugerola -> return_five -> STOP| in[5] -> TWENTY_FIVE | in[10] -> THIRTY | in[25] -> FORTY_FIVE),
TWENTY_FIVE = (sugerolaSuperDiet -> STOP | overflow -> sugerolaDiet -> return_five -> STOP | overflow -> sugerola -> return_ten -> STOP
| in[5] -> THIRTY |  in[10] -> THIRTY_FIVE | in[25] -> FIFTY),
THIRTY = (overflow -> sugerola -> return_fifteen -> STOP
                | overflow -> sugerolaDiet -> return_ten -> STOP
                | overflow -> sugerolaSuperDiet -> return_five -> STOP),
THIRTY_FIVE = (overflow -> sugerola -> return_twenty -> STOP
                | overflow -> sugerolaDiet -> return_fifteen -> STOP
                | overflow -> sugerolaSuperDiet -> return_ten -> STOP),
FORTY = (overflow -> sugerola -> return_twenty_five -> STOP
                | overflow -> sugerolaDiet -> return_ten -> STOP
                | overflow -> sugerolaSuperDiet -> return_fifteen -> STOP),
FORTY_FIVE = (overflow -> sugerola -> return_thirty -> STOP
                | overflow -> sugerolaDiet -> return_thirty_five -> STOP
                | overflow -> sugerolaSuperDiet -> return_forty -> STOP),
FIFTY = (overflow -> sugerola -> return_thirty_five -> STOP
                | overflow -> sugerolaDiet -> return_twenty -> STOP
                | overflow -> sugerolaSuperDiet -> return_twenty_five -> STOP).
```

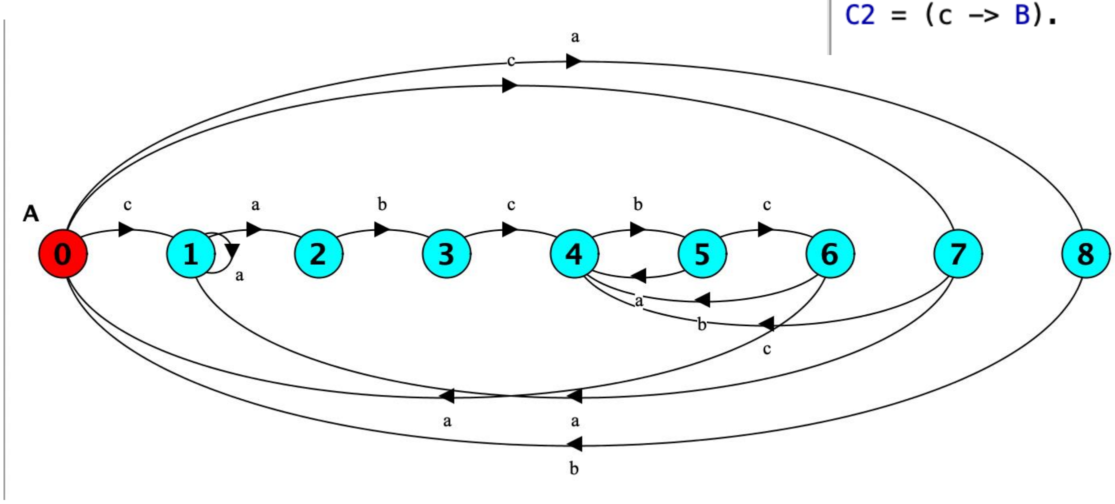


6. a) A = (a → A1 | c → A2 | c → C)
A1 = (b → A)
A2 = (a → C | c → B)
B = (b → B1)
B1 = (a → B | c → B2)
B2 = (a → A | b → B)
C = (a → C1 | a → C)
C1 = (b → C2)
C2 = (c → B)



6. B)

```
A = (a -> A1 | c -> A2 | c -> C),
A1 = (b -> A),
A2 = (a -> C | c -> B),
B = (b -> B1),
B1 = (a -> B | c -> B2),
B2 = (a -> A | b -> B),
C = (a -> C1 | a -> C),
C1 = (b -> C2),
C2 = (c -> B).
```

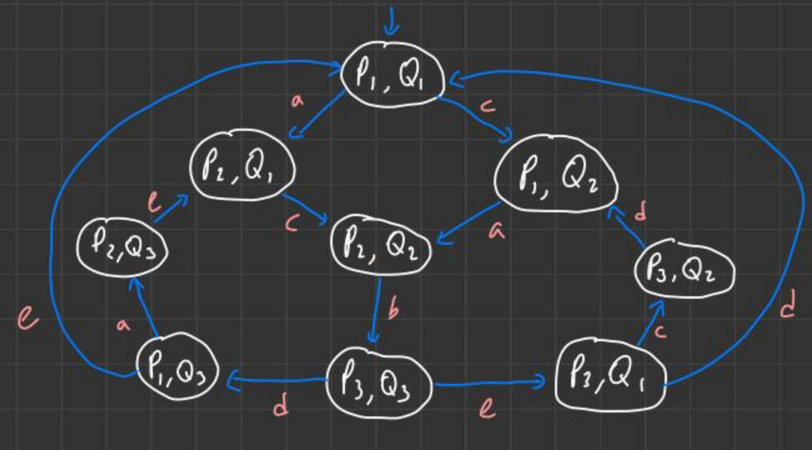


7.a) $P = (a \rightarrow b \rightarrow d \rightarrow P)$ $\quad S_1 = (P_1, P_2, P_3)$

$Q = (c \rightarrow b \rightarrow e \rightarrow Q)$ $\quad S_2 = (Q_1, Q_2, Q_3)$

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

$S1 \times S2 = ((P_1,Q_1), (P_2,Q_1), (P_3,Q_1), (P_1,Q_2), (P_2,Q_2), (P_3,Q_2), (P_1,Q_3), (P_2,Q_3), (P_3,Q_3))$



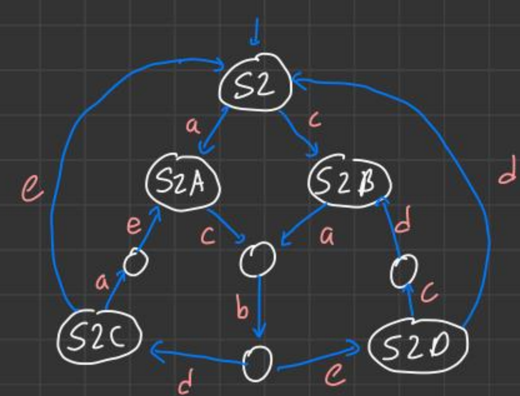
$S2 = (a \rightarrow S2A \mid c \rightarrow S2B)$

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

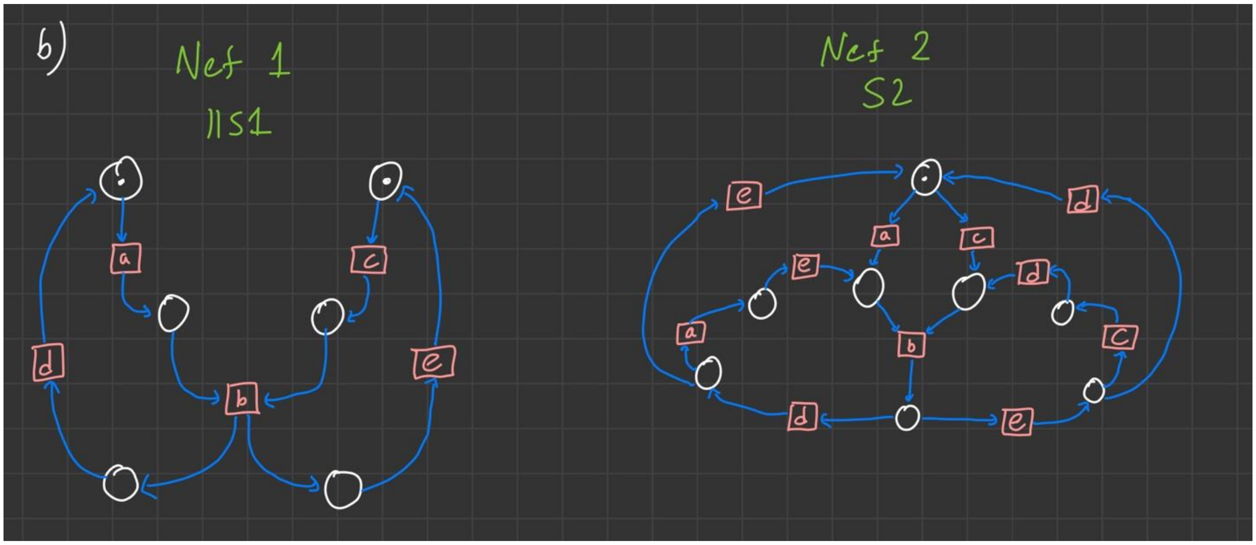
$S2B = (a \rightarrow b \rightarrow d \rightarrow S2C \mid a \rightarrow b \rightarrow e \rightarrow S2D)$

$S2C = (e \rightarrow S2 \mid a \rightarrow e \rightarrow S2A)$

$S2D = (d \rightarrow S2 \mid c \rightarrow d \rightarrow S2B)$

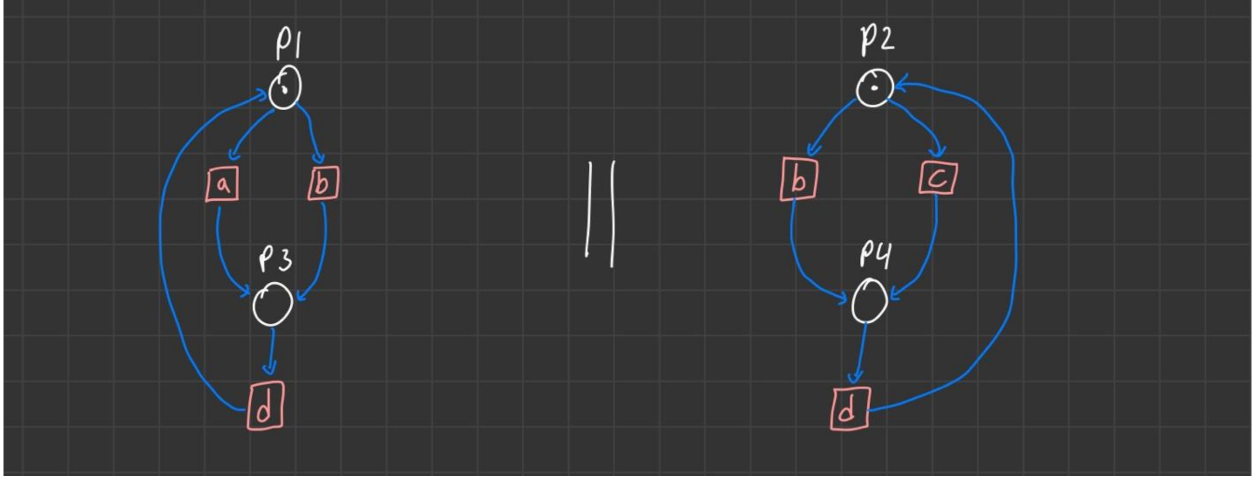


As we can see, the processes ||S1 and S2 generated isomorphic Labelled Transition Systems. We can say that LTS(||S1) = LTS(S2) as the systems are bisimilar since each possible trace that can be executed from the initial state of ||S1 can also be executed from the initial state of S2. Since the Labelled Transitions Systems of ||S1 and S2 are bisimilar or equivalent, we can then say that the Finite State Processes of ||S1 and S2 are equivalent or bisimilar.

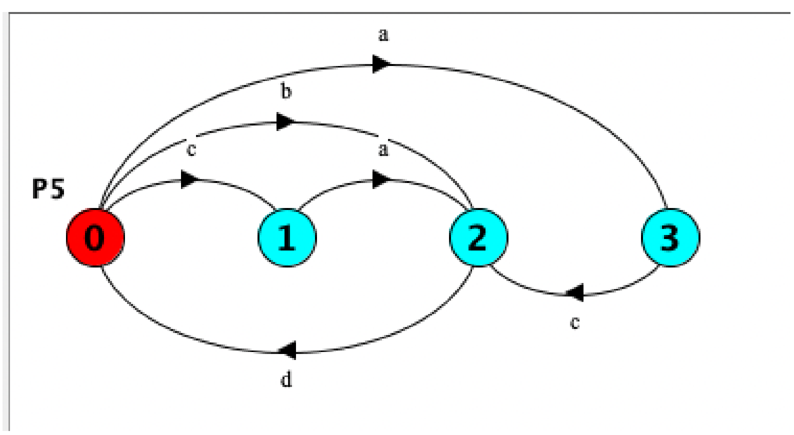


These nets are not identical. N1 allows for {a,c} and {d,e} to be executed simultaneously, while N2 does not allow for simultaneity. N2 is similar/isomorphic to the LTS diagrams of ||S1 and S2.

8. We can split the Petri Net into the following composition of Petri Nets. We can then make a composition of FSP processes from the Petri Nets.

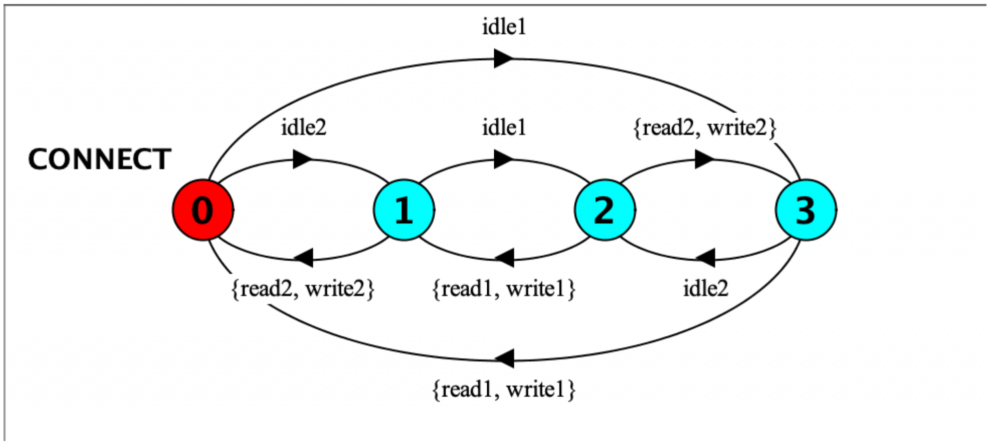



```
P1 = (a -> P3 | b -> P3),
P3 = (d -> P1).
P2 = (b -> P4 | c -> P4),
P4 = (d -> P2).
||P5 = (P1 || P2).
```



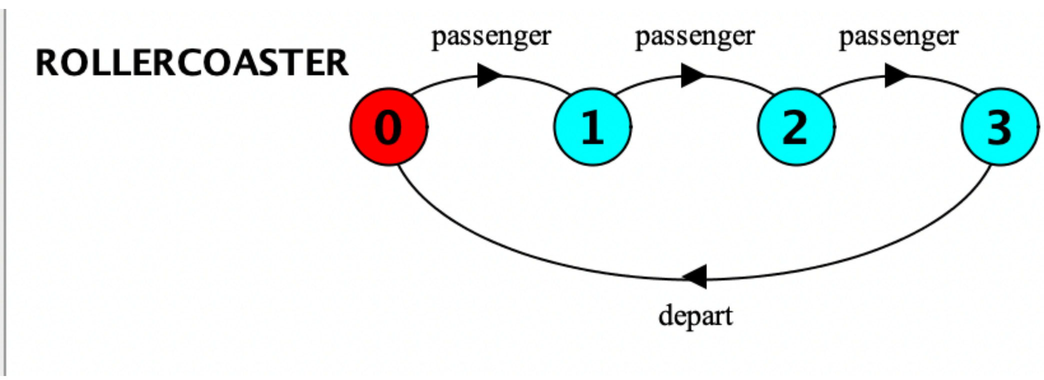
9.

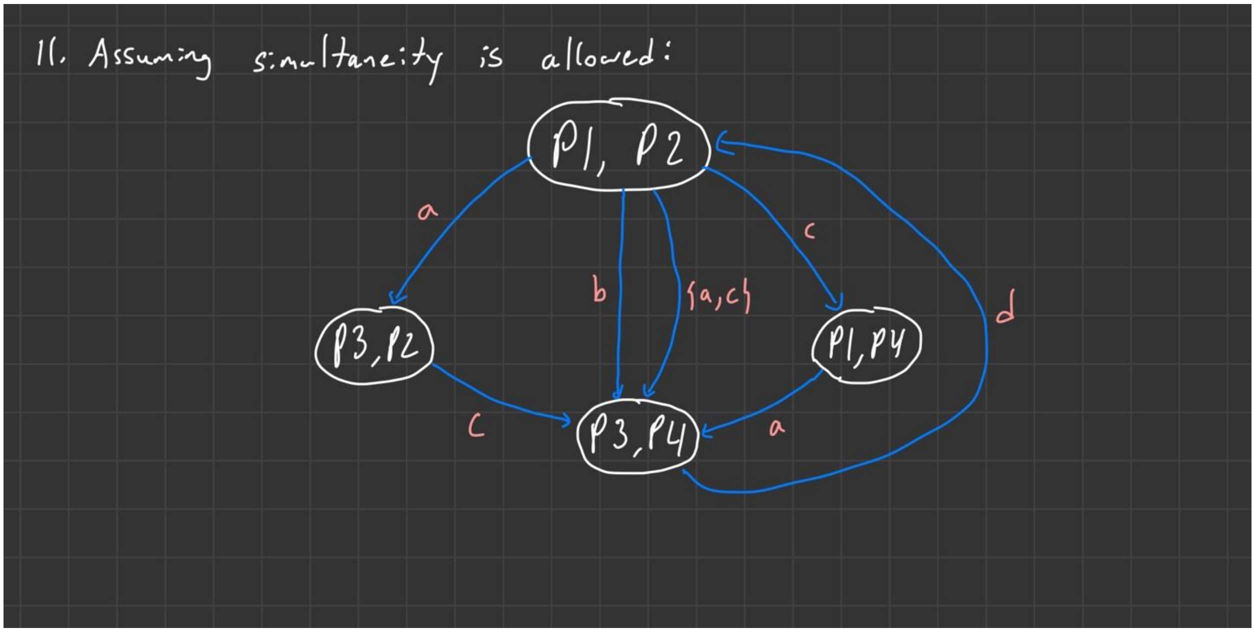
```
COMP1 = (idle1 -> (read1 -> COMP1 | write1 -> COMP1)).
COMP2 = (idle2 -> (read2 -> COMP2 | write2 -> COMP2)).
LOCK = (write1 -> LOCK | write2 -> LOCK).
||CONNECT = (COMP1 || COMP2 || LOCK).
```



10.

```
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).
```



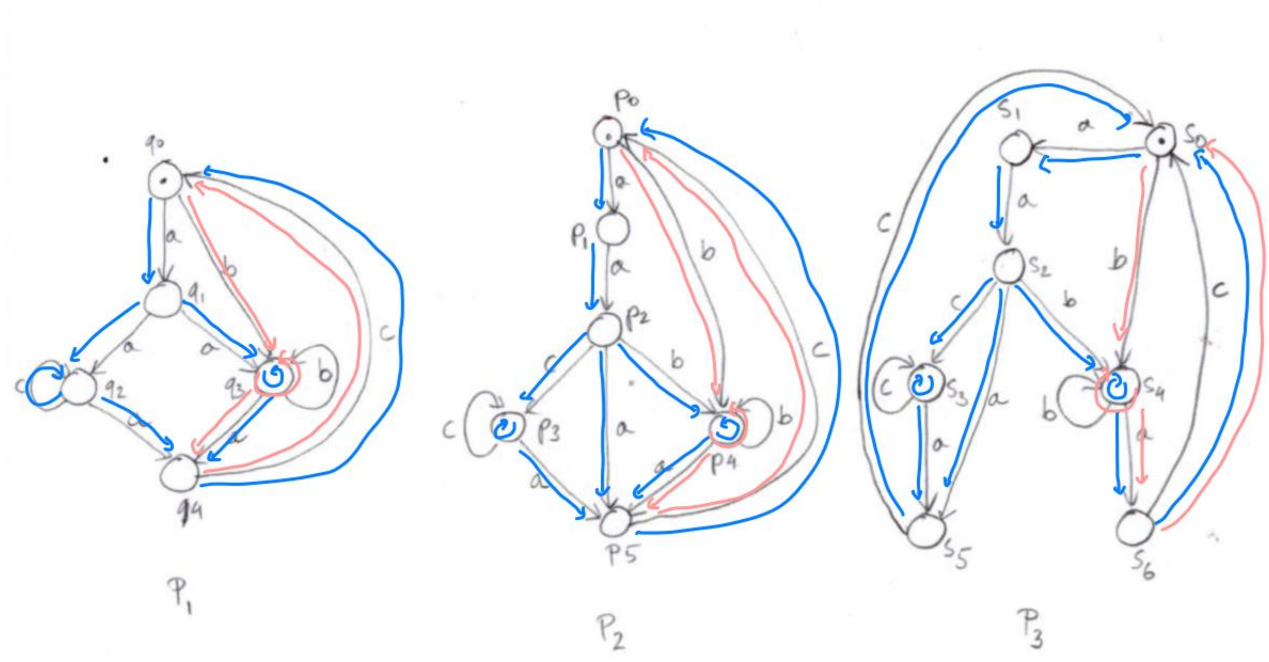


12. a)
P_0 and S_0 are bisimilar as they both allow the actions *a* and *b.*
P_1 and S_1 are bisimilar as they allow the action *a.*
P_2 and S_2 are bisimilar as they allow the actions *a*, *b,* and *c*.
P_3 and S_3 are bisimilar as they allow the actions *a* and *c.*
P_4 and S_4 are bisimilar as they allow the actions *a* and *b.*
P_5 and S_5 are bisimilar as they allow the action *c.*
P_5 and S_6 are bisimilar as they allow the action *c.*
We have proved all states and exhausted all possible cases.
Therefore, P_2 and P_3 are bisimilar.

b)
After tracing the string *aa*, P_1 would be at either state q_2 or q_3, and P_2 would be at state p_2. State q_2 allows for the actions *a* and *c* to be executed, and state q_3 allows for the actions *a* and *b* to be executed. However, state p_2 allows for the actions *a*, *b* and *c* to be executed. Thus, q_2 and p_2 are not bisimilar and q_3 and p_2 are not bisimilar, therefore making P_1 and P_2 not bisimilar.

c)
After tracing the string *aa*, P_1 would be at either state q_2 or q_3, and P_3 would be at state s_2. State q_2 allows for the actions *a* and *c* to be executed, and state q_3 allows for the actions *a* and *b* to be executed. However, state s_2 allows for the actions *a*, *b* and *c* to be executed. Thus, q_2 and s_2 are not bisimilar and q_3 and s_2 are not bisimilar, therefore making P_1 and P_3 not bisimilar.

d)



<u>Blue</u>

blue(P1) = $aa(c^* \cup b^*)ac$

blue(P2) = $aa(cc^*a \cup a \cup bb^*a)c = aa(c^* \cup b^*)ac$

blue(P3) = $aa((cc^*a \cup a)c \cup bb^*ac) = aa(c^* \cup b^*)ac$

blue = blue(P1) = blue(P2) = blue(P3) = $aa(c^* \cup b^*)ac$

<u>Pink</u>

pink = pink(P1) = pink(P2) = pink(P3) = bb^*ac

cycles = (blue $\cup$ pink)* -> (from q_0 to q_0, p_0 to p_0, s_0 to s_0)

Traces(P_1) = Traces(P_2) = Traces(P_3) = Pref((blue $\cup$ pink)*) = Pref($(aa(c^* \cup b^*)ac \cup bb^*ac)^*$)