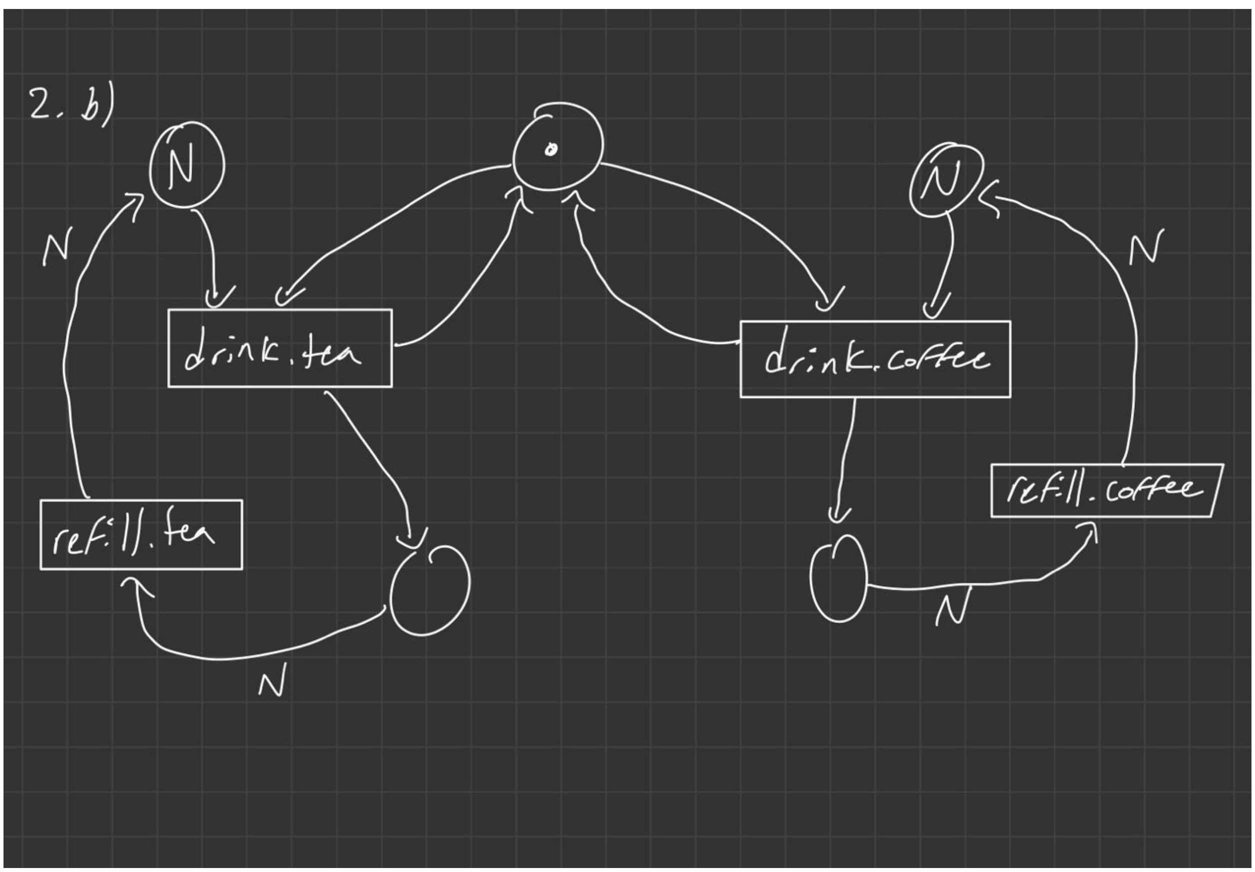
2SD3 Midterm

Rochan Muralitharan
muralr3 – 400396431

1. DOOR_CONTROLLER = CLOSED,
   OPEN = (rear -> CLOSED | front -> OPEN | both -> OPEN | neither -> CLOSED),
   CLOSED = (rear -> CLOSED | front -> OPEN | both -> OPEN | neither -> CLOSED).

2. A) const N = 2

```
CUSTOMER = (drink.tea -> CUSTOMER | drink.coffee -> CUSTOMER).
STAFF_MEMBER = (refill.tea -> STAFF_MEMBER | refill.coffee -> STAFF_MEMBER).
TEA_MACHINE = TEA_MACHINE[N],
TEA_MACHINE[i:0..N] = (when  (i>0)  drink.tea -> TEA_MACHINE[i-1]
          | when (i==0) refill.tea -> TEA_MACHINE[N]).
COFFEE_MACHINE = COFFEE_MACHINE[N],
COFFEE_MACHINE[i:0..N] = (when  (i>0)  drink.coffee -> COFFEE_MACHINE[i-1]
          | when (i==0) refill.coffee -> COFFEE_MACHINE[N]).
const C = 2
||CUSTOMERS = (forall [c:0..C] customer [c] : CUSTOMER).
||HOTEL_CAF = (CUSTOMERS || STAFF_MEMBER || TEA_MACHINE || COFFEE_MACHINE).
```

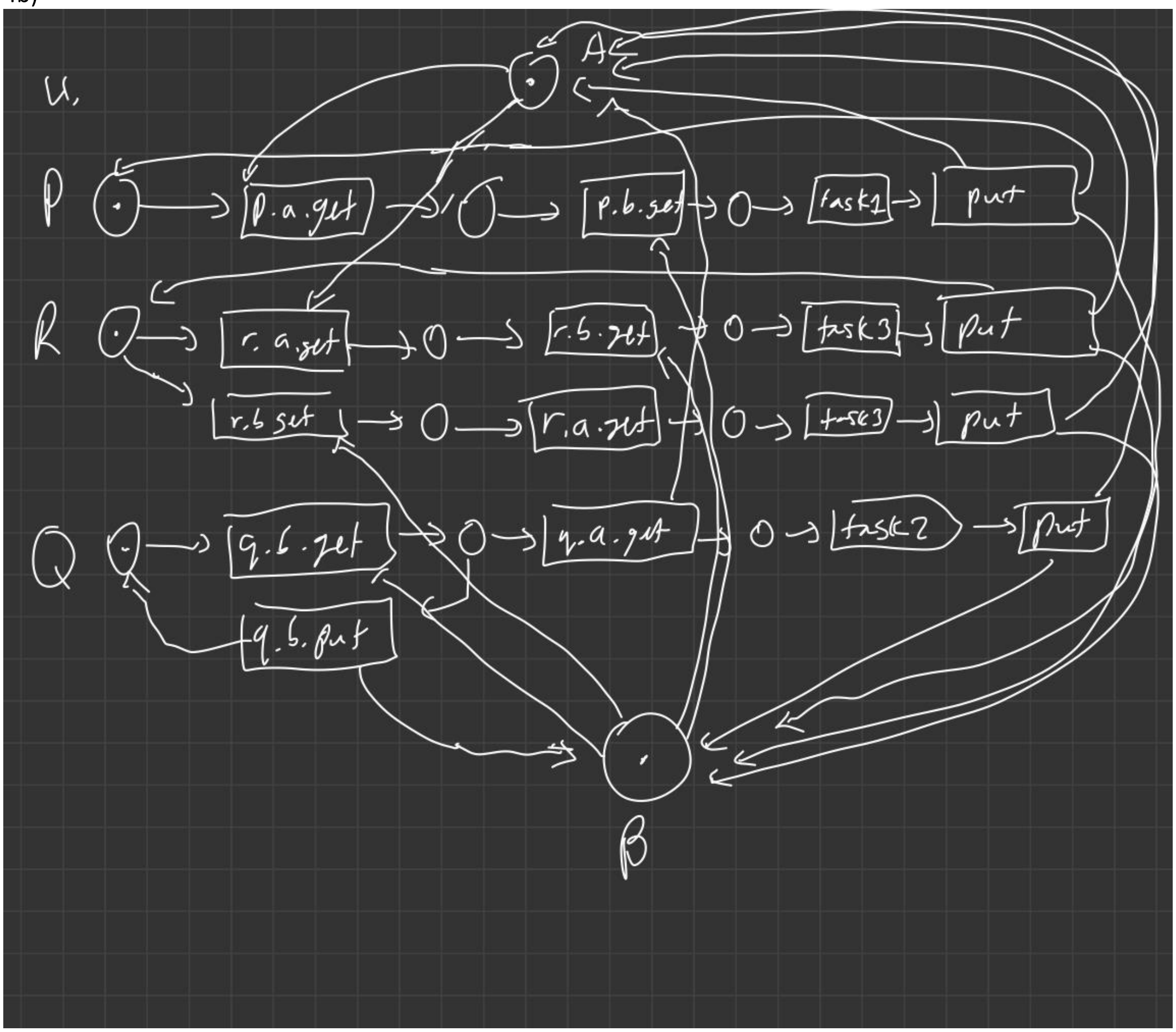


3. A = (b -> C | c -> b -> B),

B = (b -> B | a -> A),
C = (a -> b -> B | a -> B | b -> A).

4. A) RESOURCE = (get -> put -> RESOURCE).
P = (p.a.get -> p.b.get -> task1 -> p.a.put -> p.b.put -> P).
Q = (q.b.get -> (q.b.put -> Q | q.a.get -> task2 -> q.a.put -> q.b.put -> Q)).
R = (r.a.get -> r.b.get -> task3 -> r.a.put -> r.b.put -> R | r.b.get -> r.a.get -> task3 -> r.a.put -> r.b.put -> R).
||SYSTEM = (P || Q || R || {p,q,r} :: {a,b} : RESOURCE).

4b)



5. const N = 5
CLERK = (show_seats -> CLERK | select_seat -> print_ticket -> CLERK).
SEATS = SEATS[N],
SEATS[i:0..N] = (when (i>0) select_seat -> SEATS[i-1] | when (i==0) select_seat -> ERROR).
CLIENT = (select_seat -> CLIENT).
||COMPUTER = (CLERK || SEATS || CLIENT).

We set the value N to be the number of seats that are available in the system. The SEATS_AVAILABLE process keeps track of the seats that are still available that the client can select. The CLERK process will show the available seats and the customers can select an available seat. When the seat is selected, the seat will be removed from the SEATS_AVAILABLE process and lower the number of available seats by 1. This will not allow another customer from booking that seat. If there are no more seats available, the SEATS process will hit the ERROR state. Double booking is prevented by removing the seat from the available seats and not showing any taken seats in the show_seats action.

6. A) P is not bisimilar to Q as if we give in the trace “ba”, the LTS P would be in state p2, while LTS Q would be in either state q2 or q3. The state p2 allows for the actions b and c while state q2 allows for only the action b. State q3 similarly only allows the action c. Thus p2 is not bisimilar to q2 and p2 is not bisimilar to q3 and thus, P and Q are not bisimilar.

   B) To prove that P is bisimilar to S, we =have to show that there is a pair of bisimilar states for all states in P and S.
   State p0 and s0 both only have the action b, and are therefore bisimilar.
   State p1 and s1 both only have the action a when we follow the trace ”b ” and are therefore bisimilar.
   State p2 and s2 both have the actions b and c when we follow the trace ”ba ” and are therefore bisimilar.
   When we enter the trace “bab”, this case is the same as p0 and s0 which is already bisimilar.
   When we enter the trace “bac” the states p0 and s3 only have the action b and are therefore bisimilar.
   Since it has been shown all cases have bisimilar pairs, P is therefore bisimilar to S.

7. If we trace both the processes P1 and P2, we see that they have the same traces as Traces(P1) = Traces(P2) = Prefix((a(bc U cb))*).

   If we draw the Petri Nets of ||P1Q and ||P2Q, we will see that taking the right transition of “a” for ||P1Q, we will enter a deadlock state as there is no token to run the transition “c”. Thus, ||P1Q will deadlock while, ||P2Q will never deadlock.