Question 3

Part A:

Collaboration	Starting Roles	Terminating Roles	Participating Roles
C1	R2	R1	{R1,R2,R3}
C2	R2	R3	{R2,R3}
C3	R2	R3	{R2,R3}
C4	R1	{R2,R3}	{R1,R2,R3}

✓

Part B:

Collaboration	Starting Roles	Terminating Roles	Participating Roles
C1 ;w C3 = C5	SR(C1) U (SR(C3)-PR(C1))= R2 U (R2 - {R1,R2,R3})= R2	TR(C3) U (TR(C1)-PR(C3))= R3 U (R1- {R2,R3})= R3 U R1= {R3,R1}	{R1,R2,R3}
C2 ;s C3 = C6	SR(C2) = R2	TR(C3) = R3	{R2,R3}

✓

Part C:

Collaboration	Starting Roles	Terminating Roles	Participating Roles
C5 [] C6 = C7	SR(C5) U SR(C6) = R2 U R2 = R2	TR(C5) U TR(C6)= {R3,R1} U {R3}= {R3,R1}	{R1,R2,R3}
C7 ;w C4 = C	SR(C7) U (SR(C4) - PR(C7)) = R2 U (R1 – {R1,R2,R3}) = R2	TR(C4) U (TR(C7) - PR(C4))= {R2,R3} U ({R3,R1}- {R1,R2,R3})={R2,R3}	{R1,R2,R3}

Part D:

- Looking at the syntax tree for the equation, one can see that the equation is properly formed. As such, the derivation algorithm should be applicable.

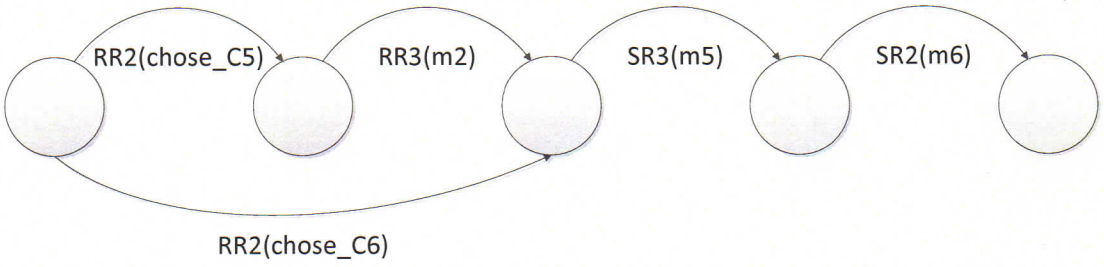
One needs local choice (satisfied)

Part E:

- It is assumed here that there is one component per role and that, for example, alloc(R2) would give simply R2 back, for notation simplicity.
- Applying the derivation algorithm:
  - (C1 ;w C3) = C5
    - Weak sequencing, no messaging required.
  - (C2 ;s C3) = C6
    - Strong sequencing
    - All roles in TR(C2) should send a message to SR(C3)
    - Message needs to be sent from R3 to R2
      - Message:
        - do_m4
  - C5 [] C6 = C7
    - Here the choice is local since the starting role for both C5 and C6 is R2.
    - Alloc(C5) = {R1,R2,R3} and alloc(C6) = {R2,R3} which means that R1 should be receiving a message informing it of the choice that was made.
      - Messages:
        - ~~chose_C5~~ not required since a message is included in C1
        - chose_C6
  - C7 ;w C4 = C
    - Weak sequencing, no messaging required.

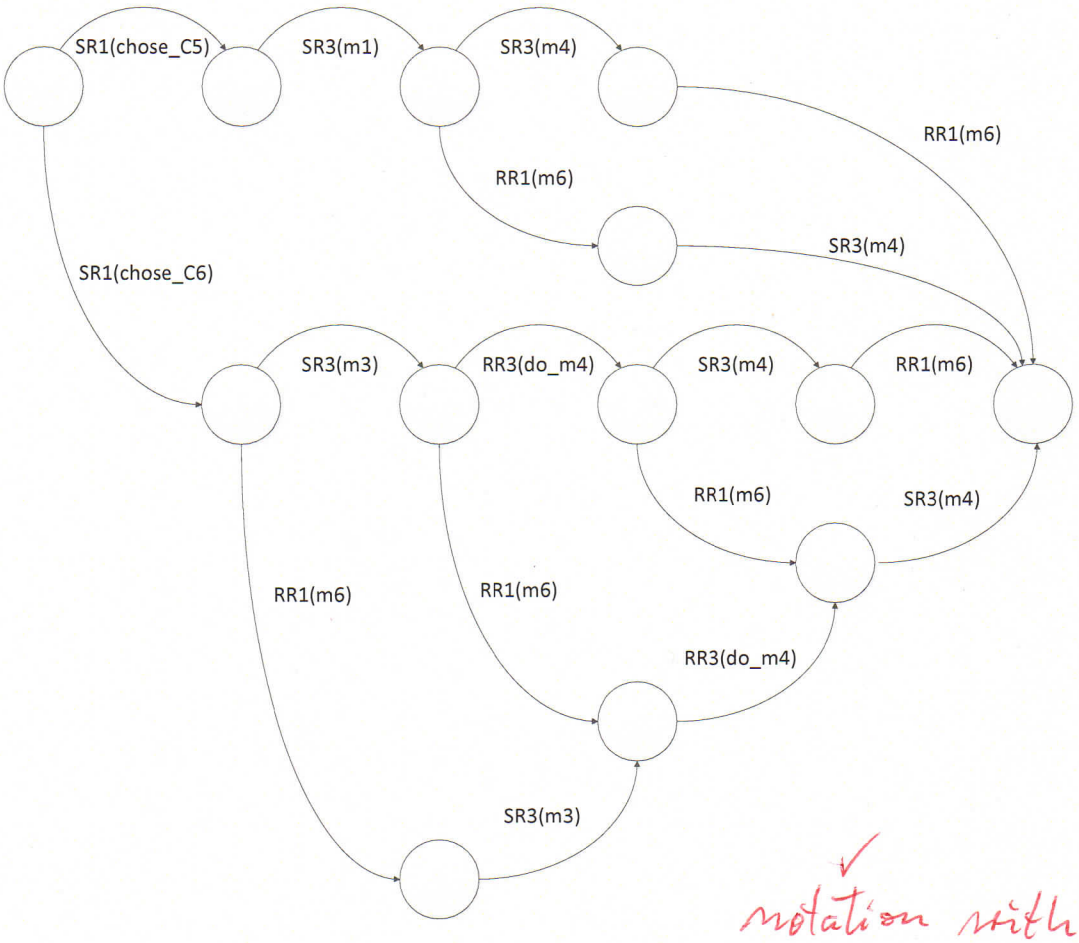
Part F:

- Role 1 has the most simple behavior:
  - If R2 decide to execute C1;wC3, then it informs R1 who will then wait for m2 before sending m5 and m6.
  - If R2 decide to execute C2;sC3, then it informs R1 who will then know it does not have to wait for m2 and will start sending m5 and m6 right away (since C4 is in weak sequence with the block before and R1 as no role in C2;sC3)



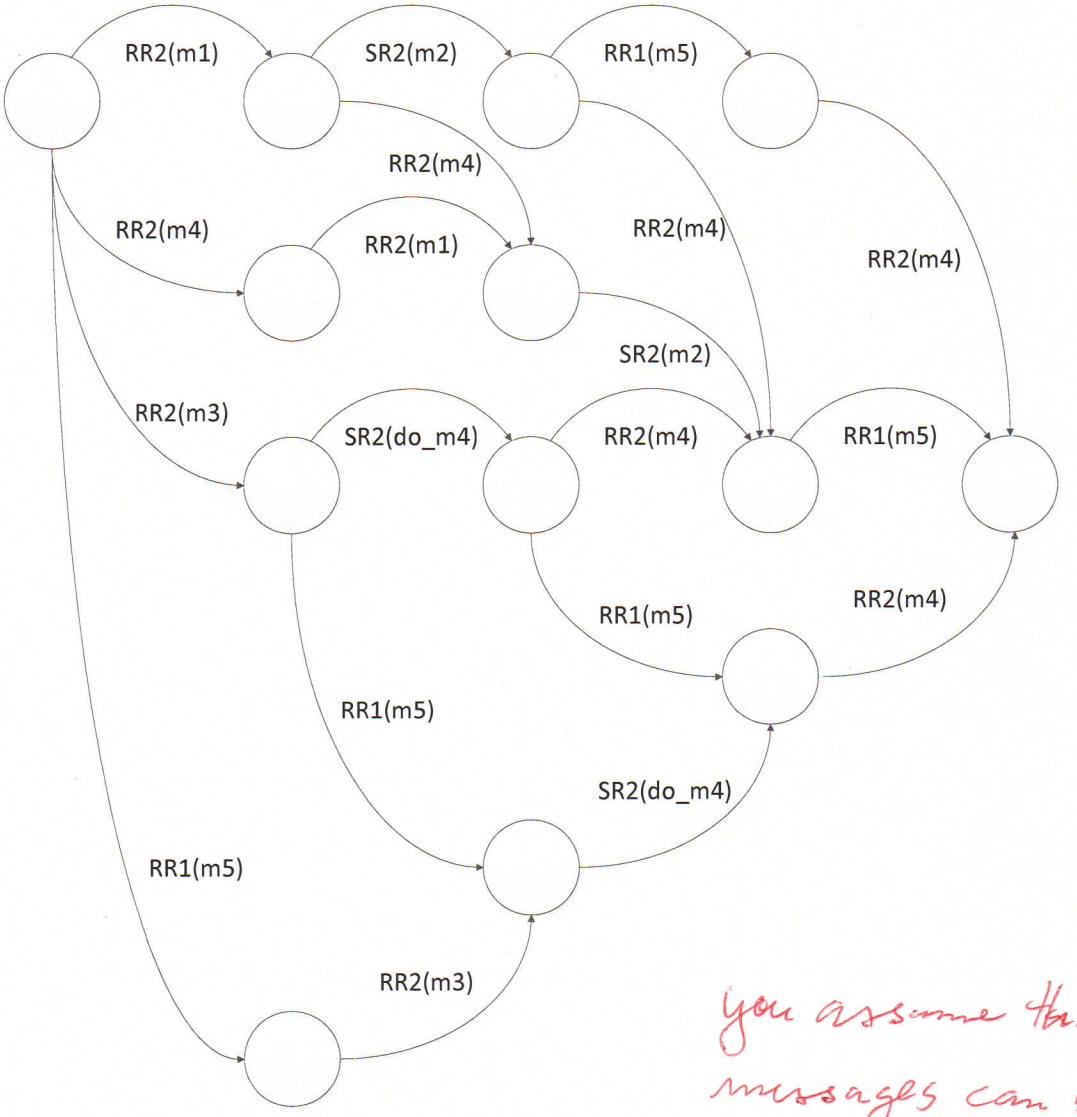
✓

- Role 2:
  - If R2 chose the C1;wC3 path, then it must start by sending m1 and cannot receive anything at that point since R1 must way for m2 to arrive which is triggered by the arrival of m1. Once m1 is sent, then m6 might arrive at any point in time and this results in 2 paths of execution: either R2 sends m4 before receiving m6 or m6 arrives before R2 has time to send m4.
  - If R2 chose the C2;sC3 path, then it R1 is not forced to wait for m2 before sending m6 and as such m6 can arrive at any point after the decision has been made. The rest of the actions must happen in order since strong sequencing is enforced.



notation with hierarchical states and concurrency would have been useful. Or you use a message pool.

- Role 3:
  - R3 does not receive any indication from R2 about the choice it did. It is not necessary as R3 can deduce the path to take depending on which message it receives.
  - If R3 receive m1 or m4, then the choice C1;wC3 was taken. Since on that path R1 must receive m2 from R3 before sending m5, then m5 can only arrive after R3 has sent m2.
  - If R4 receive m3 or m5, then the choice C2;sC3 was taken. Since on that path R1 can send m5 right away, then R3 must be ready to receive it at any time. The remaining actions must be done in order due to the strong sequencing enforced by the messages exchanged during C2;sC3.



you assume that messages can be received out of order?