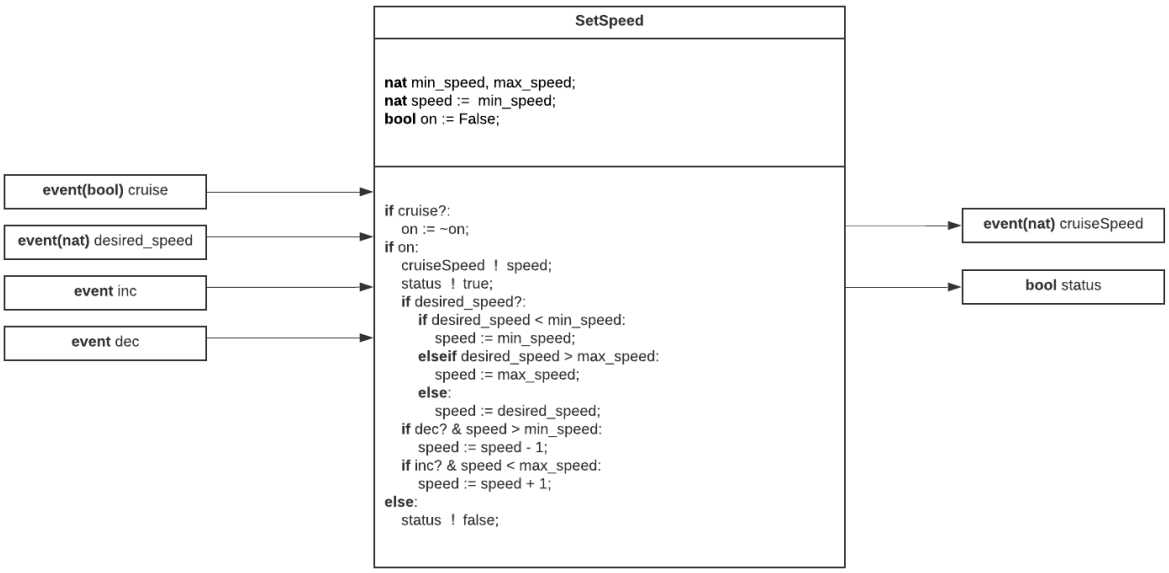
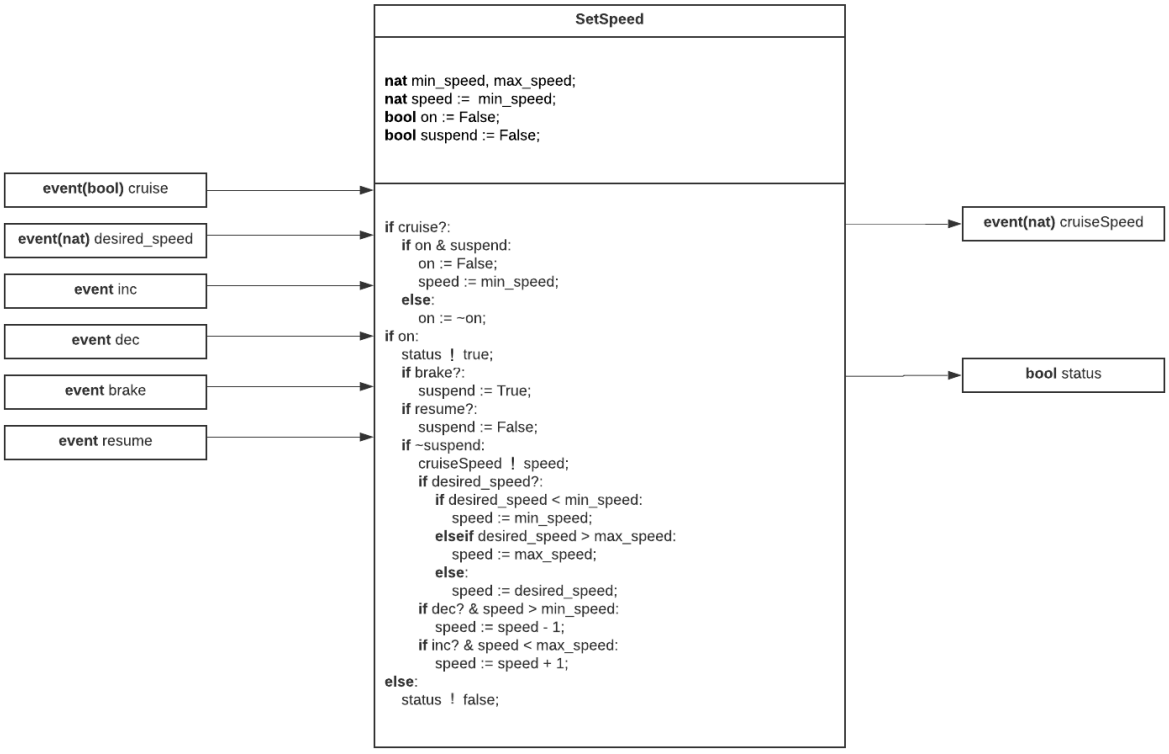
Problem 1 Event-based synchronous components

(a)



(b)



The additional state *suspend* is necessary in this case since *brake* and *resume* need to operate on states. If this SRC doesn't have the state *suspend*, the effects of *brake* and *resume* will not be able to persist

beyond the iterations during which their respective events are present, and since iteration executions are considered instantaneous in SRC, they will simply not have any observable effects.

Problem 2

Note: I'm making the following assumptions when approaching this question:

- I'm ignoring transitions to self such as $recv_3 \to recv_3$ since it's triggered whenever there is no input and there can be an infinite number of it if $P3$ makes an infinite amount of self-check in between inputs.
- I'm assuming if an input channel is shared by two processes, anything written to it will get consumed simultaneously by both processes in a fashion similar to shared input channels in composite processes. So for example, $sendReq_3!req$ is consumed simultaneously by $idle_1 \to send_{13}$ and $idle_2 \to send_{23}$. I'm making this assumption because without it, $idle_1 \to send_{13}$ might consume $sendReq_3!req$ first, and thus $idle_2 \to send_{23}$ cannot be triggered until another $sendReq_3!req$ is sent. I don't see the resend action so I'm assuming this is intended by the question. This assumption also eliminates the issue where for example, $P1$ responds twice to $P3$ before $P2$ can respond; there'd be too many possibilities if we consider cases like this.
- Although shared input channels are assumed to be consumed simultaneously, shared output channels are not. So for example, although $idle_1 \to send_{13}$ and $idle_2 \to send_{23}$ are triggered simultaneously, one of $send_{13} \to idle_1$ and $send_{23} \to idle_2$ will happen before the other and trigger $recv_3 \to recv_3$. Otherwise outputs will just overwrite one another.

(a) The first 9 transitions. Order may differ depending on the timing. 4th transition will have different states if $P1$ responds before $P2$:

1 $P3 : (idle_3, x_3 = 18, nRcv_3 = ?, sum_3 = ?) \xrightarrow[o:sendReq_3!req]{i:\epsilon} (recv_3, x_3 = 18, nRcv_3 = 0, sum_3 = 18)$

2 or 5 $P2 : (idle_2, x_2 = 15, nRcv_2 = ?, sum_2 = ?) \xrightarrow[o:\epsilon]{i:sendReq_3?req} (send_{23}, x_2 = 15, nRcv_2 = ?, sum_2 = ?)$

3 or 6 $P2 : (send_{23}, x_2 = 15, nRcv_2 = ?, sum_2 = ?) \xrightarrow[o:in_3!15]{i:\epsilon} (idle_2, x_2 = 15, nRcv_2 = ?, sum_2 = ?)$

4 $P3 : (recv_3, x_3 = 18, nRcv_3 = 0, sum_3 = 18) \xrightarrow[o:\epsilon]{i:in_3?15} (recv_3, x_3 = 18, nRcv_3 = 1, sum_3 = 33)$

5 or 2 $P1 : (idle_1, x_1 = 12, nRcv_1 = ?, sum_1 = ?) \xrightarrow[o:\epsilon]{i:sendReq_3?req} (send_{13}, x_1 = 12, nRcv_1 = ?, sum_1 = ?)$

6 or 3 $P1 : (send_{13}, x_1 = 12, nRcv_1 = ?, sum_1 = ?) \xrightarrow[o:in_3!12]{i:\epsilon} (idle_1, x_1 = 12, nRcv_1 = ?, sum_1 = ?)$

7 $P3 : (recv_3, x_3 = 18, nRcv_3 = 1, sum_3 = 33) \xrightarrow[o:\epsilon]{i:in_3?12} (recv_3, x_3 = 18, nRcv_3 = 2, sum_3 = 45)$

8 $P3 : (recv_3, x_3 = 18, nRcv_3 = 2, sum_3 = 45) \xrightarrow[o:\epsilon]{i:\epsilon} (idle_3, x_3 = 15, nRcv_3 = 2, sum_3 = 45)$ ¡¿?

9 $P2 : (idle_2, x_2 = 15, nRcv_2 = ?, sum_2 = ?) \xrightarrow[o:sendReq_2!req]{i:\epsilon} (recv_2, x_2 = 15, nRcv_2 = 0, sum_2 = 15)$

(b) After $P2$ sends a request and averages $x_1 = 12, x_2 = 15, x_3 = 18$, x_2 remains 15; then $P3$ sends a request and averages $x_1 = 12, x_2 = 15, x_3 = 18$, and x_3 also becomes 15; then $P1$ sends a request and averages $x_1 = 12, x_2 = 15, x_3 = 15$, and x_1 becomes 14. Note that all of the aforementioned updates will happen regardless of asynchrony since the order in which processes responds to a request doesn't matter in the case of averaging, and requests are send sequentially so that the next process cannot update until the previous process finishes updating. So the update of x_1, x_2, x_3 proceeds as follows:

1 $(12, 15, 18) \to$

2 $(14, 15, 15) \rightarrow$

3 $(14.4074, 14.6667, 14.5556) \rightarrow$

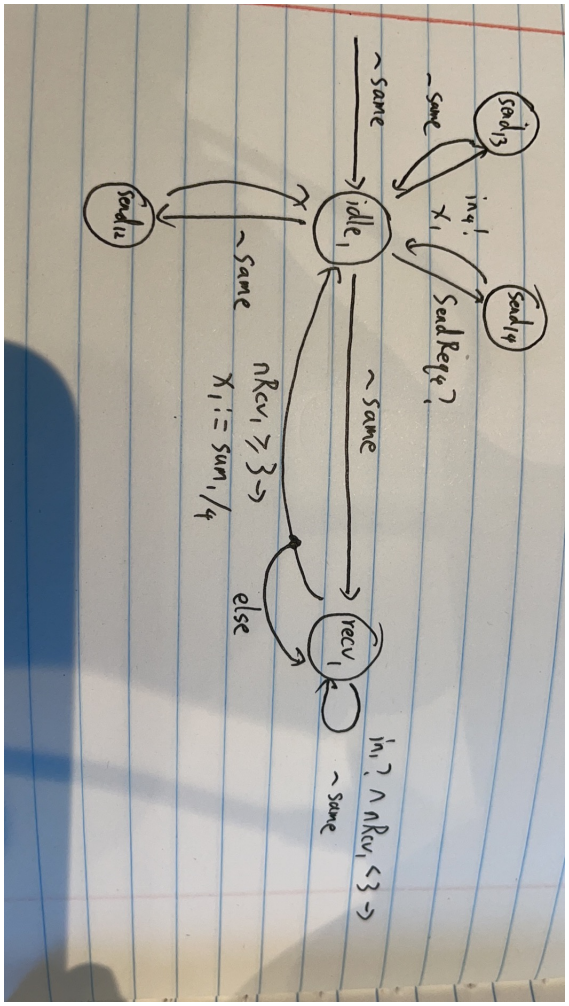
4 $(14.4842, 14.5432, 14.5021) \rightarrow$

5 $(14.4976, 14.5098, 14.4987) \rightarrow$

6 $(14.4997, 14.5020, 14.4994)$

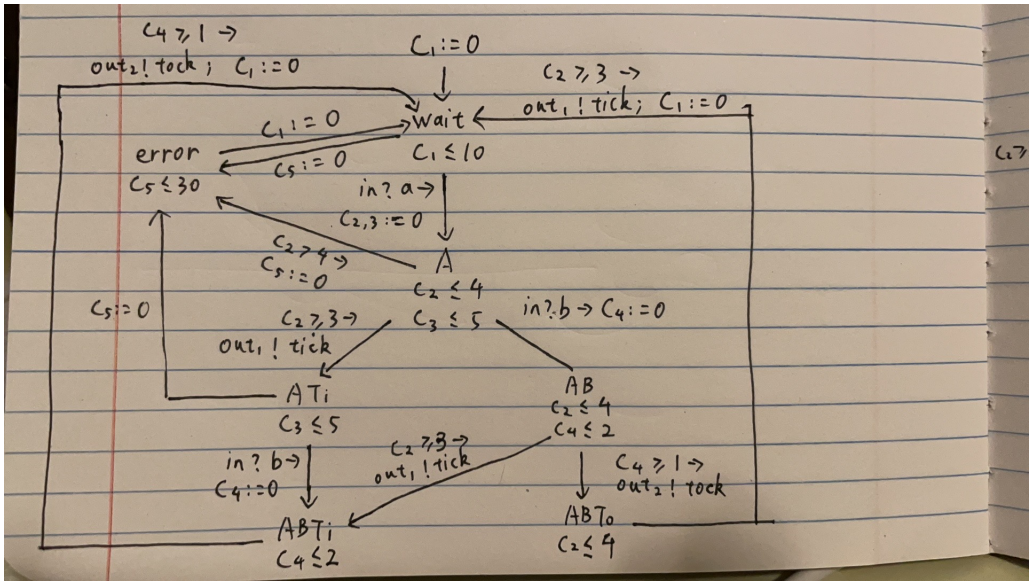
(c) Yes; for example if the schedule were $(P2, P1, P3)$ then the first iteration would become $(x_1 = 15, x_2 = 15, x_3 = 18)$, and all subsequent iterations would yield values above 15 rather than below.

(d)



Problem 3 Timed Components

(a)



(b) Assume a is received at time 0 for simplicity. For maximum $t_2 - t_1$, the smallest possible t_1 is 3, whereas the largest possible t_2 is 5 seconds waiting for b plus 2 seconds waiting for tock, so $max(t_2 - t_1) = 5 + 2 - 3 = 4$. For minimum $t_2 - t_1$, the largest possible t_1 is 4, whereas the smallest possible t_2 can be obtained with 0 second waiting for b plus 1 second waiting for tock, so $min(t_2 - t_1) = 1 - 4 = -3$.

(c) We can pad 4 seconds to the timing constraints in step 3, so tock happens at least 5 seconds and at most 6 seconds after b.

Problem 4 Dynamical Systems

(a)

$$A = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} = \begin{bmatrix} -\frac{0.1}{0.01} & \frac{0.1}{0.01} \\ -\frac{0.1}{0.5} & -\frac{1}{0.5} \end{bmatrix} = \begin{bmatrix} -10 & 10 \\ -0.2 & -2 \end{bmatrix}$$

$\lambda_1 = -6 + \sqrt{14} < 0, \lambda_2 = -6 - \sqrt{14} < 0$, so the system is not only Lyapunov stable, but also asymptotically stable, which means it converges to 0 with the region of attraction spanning the entire $Real^n$.[1]

(b) Since $v = 0$, this is an autonomous system characterized by the linear dynamical system $\dot{x} = Ax$. As explained in (a), the only equilibrium point for $\dot{x} = Ax$ with global field of attraction is the zero vector, so $[x_1(\infty), x_2(\infty)]^T$ approximates $[0, 0]^T$.[2]

(c) $v = [1, -1][x_1, x_2]^T = x_1 - x_2$, so $[0, \frac{1}{L}]^T v = [0, \frac{x_1 - x_2}{L}]^T$. Therefore $Ax + u$ becomes

$$\begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{x_1 - x_2}{L} \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K-1}{L} & -\frac{R+1}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

[1] Found this on Principles of Cyber-Physical Systems p260,261. We could have also manually checked equilibrium point by solving $Ax = 0$, which would have also given the only equilibrium point to be the zero vector.

[2] Alternatively, we may also get the same result by calculating $x(\infty) = e^{A\infty}[0.2, 1]^T$. $e^{A\infty} = Te^{D(\lambda_1, \lambda_2)\infty}T^{-1} = Te^{D(\lambda_1 \infty, \lambda_2 \infty)}T^{-1} = 0$, where T is the matrix of eigenvectors and D(a,b) represents diagonal matrix with entries a and b on the diagonal.

, which gives us the new A. The new eigenvalues are $\lambda_1 = -7 - 3\sqrt{3} < 0, \lambda_2 = -7 + 3\sqrt{3} < 0$, so still stable.

(d) $Ax + u$ now becomes

$$\begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{3x_1 - x_2}{L} \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K-3}{L} & -\frac{R+1}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

, and the new eigenvalues are $\lambda_1 = -7 - \sqrt{67} < 0, \lambda_2 = -7 + \sqrt{67} > 0$. One of the eigenvalues is positive, so no longer stable.

Problem 5 Asynchronous Processes

(a) Yes; since T_2 is always enabled we can just keep running T_2 and never touch x or z.

(b) No; the first 2 executions have to be T_2 and then T_3, since they are the only enabled tasks in the first and second executions; T_3 changes z.

(c) Yes; just run T_2, T_3, T_2, T_3 etc.

(d) No; the first 2 executions have to be T_2 and then T_3; in the third execution, only T_1 and T_4 are enabled and T_4 directly sets $x := 2$, so we have to run T_1 and set $x := (0+1)\%3$ while resetting y and z, which again returns us to the start of $T_2 \to T_3 \to T_1$ sequence, setting $x := (1+1)\%3 = 2$.