Homework 7

Process algebra

whether the following rules should be accepted

a) $\boldsymbol{P|P = P}$

No. For example, $P = tick$. In this case, the $P|P$ should have the same effect as $tick.tick$, which is apparently not P.

b) $\boldsymbol{P \setminus b \setminus b = P \setminus b}$

Yes, because $P\setminus b$ suppresses the occurrence of b in process P. For a process Q that doesn't consist of any b, we have $Q\setminus b = Q$. Therefore, we can conclude that $(P\setminus b)\setminus b = P\setminus b$.

c) $\boldsymbol{(a.P)|(a.Q) = a.(P|Q)}$

No, because $(a.P)|(a.Q) = a.\big(P|(a.Q)\big) + a.\big((a.P)|Q\big)$.

Apparently, $\big(P|(a.Q)\big) + \big((a.P)|Q\big) \neq (P|Q)$. So, the above rule should not be accepted.

Ricart-Agrawala Algorithm

a) ***Provide an example that illustrates why the modified algorithm is no longer correct.***

Suppose that there are two processes and one is waiting in the deferred list of the process that is currently in the for loop. If the loop is executed before setting requestCS false, it would be possible that after executing line $Main{:}p11$. The other process then may enter the critical section and come back to $Main{:}p5$ and re-send the request. Because we suppose that **Receive** can preempt **Main**, the **Receive** function of process that is waiting at line $Main{:}p11$ may execute to line $Receive{:}p3$ and then finds that "*not requestCS*" is ***false*** so it adds the other process to deferred list again.

The procedure stated above can happen indefinitely and block one of the process forever.

b) ***Can the statement be replaced?***

Because it is possible that several processes with different requestNum sending request together. If the request with lower requestNum arrives after the one with higher requestNum, the resulting highestNum would not be actually the highest one.

Token-Passing

a) In node i, can the value of $requested[j]$ be less than the value of $granted[j]$ for $j \neq i$? Why? Justify your answer.

No, because $granted$ of other nodes other than i can only be updated by $sendToken$. However, the precondition of sending $granted$ is $requested[N] > granted[N]$. While the initial state is $requested[j] = granted[j]$ for all j, we can conclude that there is no such j that satisfies $j \neq i$ and $requested[j] < granted[j]$ by induction.

b) In node i, can the value of $\mathbf{requested}[\boldsymbol{j}]$ be greater than the value of $\mathbf{granted}[\boldsymbol{j}]$ for $j \neq i$? Why? Justify your answer.

Yes, because other nodes may request for new tokens after being granted one. This requires those nodes to increase myNum by one and update $requested[j]$ to the newer myNum. Node i then knows the newer $request[j]$ by code in line $p13 - p14$.