Chapter 7

(Sent by Simin Nadjm-Tehrani)

In the presentation of the Lamport mutual exclusion protocol (chapter 7, p107), the program will terminate immediately as all guards are false to begin with.

I think the code misses two important lines that include the code for the CS-operations and the code for non-CS-operations in each process.

Chapter 8

In page 135, Fig. 8.7, on the line showing the events of process 2, the second letter should be j instead of a.

Chapter 9

In page 143 Section 9.3.2 line 5, N should be n.
In page 147 Section 9.4.2 line 8, the correct line is “A knot is a subgraph of a directed graph …”)

(Sent by Simin Nadjm-Tehrani)

In **page 139**, the sentence "Process i will execute statement 1 for the (r+1)st time when (V^r.i \superset W^r.i)” is not true. According to definition of X^r it denotes the value of X after process i executing statement 1. At this point W.i is always = V.1! So if the intention is to refer to W.i and V.i *before* executing statement 1 then V^r.i and W^r.i cannot be used for that purpose.

Equation (9.1) holds under a hidden assumption. The assumption is that the when process k receives and delivers the increment from i, process i is still in its r+1 round (i.e. it has not received messages from other nodes and acted upon statement 2 - once or several times - thus changing its V.i during the interval that k is receiving and acting on the increment from i). This is a hidden synchrony assumption.

I think Theorem 9.2 is not correctly formulated. Or rather the presented proof is not proving that theorem. I think your intention might be to say that:

- If the algorithm terminates then it is correct in the sense that every process has received the state from every other process.

Whether the algorithm terminates or not must be subject to the condition that delivery of messages can be done within a finite time bound (otherwise channels do not get empty). I think that a fixed-point type of argument might be needed to show that the increments will become empty after a finite number of message exchanges.

Chapter 10

(Sent by Simin Nadjm-Tehrani)

1) The code for the initiator on **page 153** will not allow the neighbors of the initiator to ever reach deficit=0. Since the initiator never acks, the neighbors will forever remain at

deficit=1 once they reach that stage.
□
The correction could be to add the following rule to the initiator:
[] message = (S,k) and S>=D□→ send ack to sender
□
Chapter 11
□*(Sent by Simin Nadjm-Tehrani)*

In the bully algorithm on **page 174**, the processes with id<i will never be informed of the election of the new leader. This is an unclear question. Perhaps the formulation intends that each lower id j should in turn discover the failed leader and initiate an election (thereby repeating the process for election N-1 times in the worst case). However, this is highly unlikely to have been meant as it is extremely inefficient.
□
Another interpretation is that the second line from bottom on page 174 is incorrect and it should say: send leader to all j (instead of: to all j >i). If we check the proof on page 175 it seems that your proof in the bold font part assumes the latter.
□