COMP 6752 Homework 9, Question 3

This homework question centres around Peterson's famous mutual exclusion algorithm as running example. It is an improvement of the brilliant original algorithm of Dekker.

The algorithm deals with two concurrent processes A and B that want to alternate critical and noncritical sections. Each of these processes will stay only a finite amount of time in its critical section, although it is allowed to stay forever in its noncritical section. It starts in its noncritical section. The purpose of the algorithm is to ensure that they are never simultaneously in the critical section, and to guarantee that both processes keep making progress.

The processes use three variables. The Boolean variable $readyA$ can be written by process A and read by process B, whereas $readyB$ can be written by B and read by A. By setting $readyA$ to $true$, process A signals to process B that it wants to enter the critical section. The variable $turn$ is a shared variable: it can be written and read by both processes. Its use is the brilliant part of the algorithm. Initially $readyA$ and $readyB$ are both $false$ and $turn = A$.

Process A

repeat forever

$$\left\{\begin{array}{ll} \ell_1 & \textbf{leave noncritical section} \\ \ell_2 & readyA := true \\ \ell_3 & turn := B \\ \ell_4 & \textbf{await}\ (readyB = false \vee turn = A) \\ \ell_5 & \textbf{enter critical section} \\ \ell_6 & \textbf{leave critical section} \\ \ell_7 & readyA := false \\ \ell_8 & \textbf{enter noncritical section} \end{array}\right.$$

Process B

repeat forever

$$\left\{\begin{array}{ll} m_1 & \textbf{leave noncritical section} \\ m_2 & readyB := true \\ m_3 & turn := A \\ m_4 & \textbf{await}\ (readyA = false \vee turn = B) \\ m_5 & \textbf{enter critical section} \\ m_6 & \textbf{leave critical section} \\ m_7 & readyB := false \\ m_8 & \textbf{enter noncritical section} \end{array}\right.$$

a. What would be wrong with this protocol if we omitted the variable $turn$?

b. Express **Process A** as a CCS, CSP or ACP expression, featuring eight atomic actions $\ell_1, \ldots, \ell_8$ and a recursive equation with variable X. Feel free to use the simpler treatment of recursion, by seeing recursion variables as constants, and simply writing X instead of $\langle X|\mathcal{S}\rangle$.

c. Represent **Process A** as a process graph, by using $\ell_1, \ldots, \ell_8$ as transition labels. Also attach names to the states by calling each state after the transition that is enabled there. Thus transition ℓ_3 goes from state ℓ_3 to state ℓ_4.

d. Correct the answers to questions 1b and 1c by replacing the action ℓ_4 by the two actions "*B not ready*" and "$turn = A$". "*B not ready*" denotes the action of reading the value of $readyB$ and finding that it is $false$. Likewise, "$turn = A$" denotes the action of reading the value of $turn$ and finding that it is A. [You may skip the original answers to b and c.]

   It would be possible to model instruction ℓ_4 assuming *busy waiting*, by drawing self-loops in state ℓ_4 labelled "*B ready*" and "$turn = B$". However, I want to abstract from these unsuccessful read actions from the onset by not including them in our formal specification. Thus, the intuition of the **await** statement is that the process A patiently sits in state ℓ_4 until one of the transitions "*B not ready*" or "$turn = A$" is enabled.

e. Represent the behaviour of variable *readyA* as a process graph. Its transition labels are the (synchronisation partners $\overline{\ell_2}$ and $\overline{\ell_7}$ of the) write actions ℓ_2 and ℓ_7 that can be performed by processes **A**, and the (synchronisation partner $\overline{A\ not\ ready}$ of the) read action "*A not ready*" that can be performed by process B.

f. Give a process algebraic expression of this behaviour.

g. Represent the behaviour of variable *turn* as a process graph. Its transition labels are the (synchronisation partners of the) write actions ℓ_3 and m_3 that can be performed by processes **A** and **B**, and the read actions $turn = A$ and $turn = B$. Also give a process algebraic expression of this behaviour.

h. Now give a process algebraic expression of the entire protocol, involving the parallel composition of 5 processes. All actions except the entering and leaving of the critical and noncritical sections (ℓ_1, ℓ_5, ℓ_6, ℓ_8, m_1, m_5, m_6 and m_8) are internal (only needed to make the protocol work) and should be abstracted away. You may choose whether to use CCS, CSP or ACP, and feel free to rename for instance $\overline{\ell_2}$ into ℓ_2 if this suits you.

i. On the next page you see the potential states of a process graph representation of the entire algorithm. A state of the algorithm is completely determined by a state of process A, a state of process B and a state of *turn*. For the states of *readyA* and *readyB* are completely determined by the states of A and B. This observation yields $8 \times 8 \times 2 = 128$ potential states.

   Complete the given drawing into a process graph by supplying the transitions. Don't bother labelling them. Also don't draw loops backwards to the left column or top row; instead use the grey shadows, which represent copies of the states at the opposite end of the diagram. How many states are reachable from the initial state?

In the future there will be follow-up homework questions on this running example.

Process Graph of Peterson's Mutual Exclusion Algorithm