Szymanski’s protocol

This problem consists in transferring the pseudocode of a complicated $N$-way mutual exclusion algorithm into a model of its implementation. The algorithm in question is the algorithm by Szymanski. It is intended for an arbitrary number of identical processes, ordered in a linear array. The process behaviours are defined through a finite set of actions. An action represents a change of local state of a process. An action may be conditioned on both the local state of the process, and the context in which it may take place. The context represents a global condition on the local states of the rest of processes inside the system. The processes are distinguished by unique indexes (e.g., from 0 to $N - 1$, where $N$ is the number of processes.

An idealized pseudo-code for Szymanski’s mutual exclusion algorithm can be given as follows. In the algorithm, an arbitrary number of processes compete for a critical section. The local state of each process consists of a control state ranging over the integers from 1 to 7 and of two boolean flags, $w$ and $s$. A pseudo-code version of the actions of any process $i$ could look as follows:

1: \texttt{await } \forall j : j \neq i : \neg s_j  \\
2: w_i, s_i := \texttt{true, true}  \\
3: \texttt{if } \exists j : j \neq i : (pc_j \neq 1) \land \neg w_j \texttt{ then } s_i := \texttt{false, goto 4}  \\
\quad \texttt{else } w_i := \texttt{false, goto 5}  \\
4: \texttt{await } \exists j : j \neq i : s_j \land \neg w_j \texttt{ then } w_i, s_i := \texttt{false, true}  \\
5: \texttt{await } \forall j : j \neq i : \neg w_j  \\
6: \texttt{await } \forall j : j < i : \neg s_j  \\
7: s_i := \texttt{false, goto 1}

For instance, according to the code at line 6, if the control state of a process $i$ is 6, and if the context is that the value of $s$ is $\texttt{false}$ in all processes to the left, then the control state of $i$ may be changed to 7. In a similar manner, according to the code at line 4, if the control state of a process $i$ is 4, and if the context is that there is at least another process (either to the right or to the left of $i$) where the value of $s$ is $\texttt{true}$ and the value of $w$ is $\texttt{false}$, then the control state and the values of $w$ and $s$ in $i$ may be changed to 5, $\texttt{false}$, and $\texttt{true}$, respectively.

Your problem is to realize this pseudocode by a model of implementation, where the atomic actions are realistic. I.e., an atomic action may only read or write a single local variable of one process. Note that the algorithm may work or not work, depending on how this is done. To check that you have a correct translation, use SPIN to check that the algorithm enforces mutual exclusion at line 7. Also investigate which guarantees of non-starvation are given by the algorithm. Of course, SPIN can do this only for a configuration of a bounded number of processes.
Exercise on Temporal Operators.

Since. Consider the new binary temporal operator $S$, pronounced “since”. Intuitively, since is the “backwards” analogue of (strong) until. That is, $\phi_1 S \phi_2$ means that the last occurrence of $\phi_2$ was followed by a period of $\phi_1$ up to the present from the state after that where $\phi_2$ held. The formal semantics can be described as

$$\bullet \quad (\sigma, i) \models \phi_1 S \phi_2 \quad \text{iff} \quad \exists j \leq i : (\sigma, j) \models \phi_2 \quad \text{and} \quad \forall k : j < k \leq i \quad (\sigma, k) \models \phi_1$$

Your problem is the following:

a) Express $\Box (p \implies p S q)$ as a formula containing $p$, $q$, and the other temporal operators that we have used ($\circ$, $\Box$, $\Diamond$, $U$, $W$).

b) Draw a Büchi automaton that accepts the language that satisfies $\Box (p \implies p S q)$.

c) Make a never-claim in PROMELA that will check whether a program satisfies $\Box (p \implies p S q)$.

Validity of Temporal Formulas

Which of the following temporal logic properties are valid (i.e., holds for any possible computation)? Here $p$ and $q$ are arbitrary state formulas.

a) $\Diamond p \land \Box q \quad \Leftrightarrow \quad \Diamond (p \land \Box q)$

b) $(\Box p \lor \Diamond q) \quad \Leftrightarrow \quad p W (\Diamond q)$

c) $\Diamond \Box (p \implies \Box q) \quad \Leftrightarrow \quad (\Diamond \Box q \lor \Diamond \Box (\lnot p))$

d) $\Diamond \Box p \land \Diamond \Box q \quad \Leftrightarrow \quad \Diamond (\Box p \land \Box q)$

e) $\Diamond p \land \Box q \quad \Leftrightarrow \quad \Box (\Diamond p \land q)$

Note that you can use SPIN to do this problem for you (so no risk for mistakes!). Describe how to do that.