Distributed Algorithms 236358
Home Assignment 1

Submission date: 24/4/2013, 12:00, in pairs.
Submit on GR or by email to hagit@cs.technion.ac.il

1. A schedule is synchronous if for every pair of processes $p_i$ and $p_j$, there is an appearance of $p_j$ between every two consecutive appearances of $p_i$.

Present an algorithm that uses $o(n)$ distinct shared memory states, and provides mutual exclusion in every synchronous schedule, no deadlock and $k$-bounded waiting. The algorithm is allowed to use RMW operation. Note that this implies that the lower bound of $\Omega(n)$ distinct shared memory states does not hold for synchronous schedules.

2. The next page presents Peterson’s algorithm for two-process mutual exclusion. Prove that the algorithm provides mutual exclusion and no-starvation.

The next page also presents the Filter algorithm, generalizing Peterson’s algorithm for $n$ processes. The Filter algorithm creates $N$ ”waiting rooms”, called levels, which a process must pass through before entering the critical section. In each level of the filter, at least one process enters the level and at least one process is blocked if many try to enter the level. Prove that for every $\ell$, $0 \leq \ell \leq N - 1$, there are at most $N - \ell$ processes at level $\ell$.

3. An algorithm solves the $k$-mutual exclusion problem if at any time at most $k$ processors are in the critical section, and in addition satisfy the efficient resource utilization property: A process wishing to enter the critical section will do so if $k - 1$ processes (or less) are in the critical section.

Present a $k$-mutual exclusion algorithm using only read/write (possibly unbounded) registers, and prove that it provides $k$-mutual exclusion and efficient resource utilization.

4. We defined that a configuration $C$ is similar to (or indistinguishable from) a configuration $C'$, with respect to a set of processes $P$, denoted $C \overset{P}{\cong} C'$, if each processor in $P$ has the same state in $C$ as in $C'$ and $\text{mem}(C) = \text{mem}(C')$. For every finite schedule $\sigma$, we denote by $\sigma(C)$ the final configuration of the execution segment $\text{exec}(C, \sigma)$. Prove formally that for any finite $P$-only schedule $\sigma$, if $C \overset{P}{\cong} C'$ then $\sigma(C) \overset{P}{\cong} \sigma(C')$.

5. Prove that in any no-deadlock mutual exclusion algorithm, using only read and write operations, the entry code for process $p$ must include a write to some shared variable $A$, followed by a read to a different shared variable $B$, without $p$ writing to $B$ in between.
Algorithm 1 Peterson’s algorithm
Initially flag[0] and flag[1] are false and victim is 0

code for \( p_0 \)
\begin{align*}
1: & \text{flag}[0] := \text{true} \\
2: & \text{victim} := 0 \\
3: & \text{while} \ flag[1] \text{ and victim}==0 \text{ do wait} \\
4: & \text{flag}[0] := \text{false}
\end{align*}

\text{(Entry)}:
\text{(Critical Section)}:
\text{(Exit)}:
\text{(Remainder)}:

\begin{align*}
\text{code for } p_1
1: & \text{flag}[1] := \text{true} \\
2: & \text{victim} := 1 \\
3: & \text{while} \ flag[0] \text{ and victim}==1 \text{ do wait} \\
4: & \text{flag}[1] := \text{false}
\end{align*}

\text{(Entry)}:
\text{(Critical Section)}:
\text{(Exit)}:
\text{(Remainder)}:

Algorithm 2 Filter algorithm
level\([N]\), victim\([N-1]\) are arrays of integers. Initially level\([i]=0\) and and victim\([i]=0\).

code for \( p_i \)
\begin{align*}
1: & \text{for } (l=1; l<N; l++) \text{ do wait} \\
2: & \text{level}[i] := l \\
3: & \text{victim}[1] := i \\
4: & \text{while } (\exists k \neq i, \text{ level}[k] \geq l) \text{ and } \text{ (victim}[l]==i) \text{ do wait} \\
5: & \text{level}[i] := 0
\end{align*}

\text{(Entry)}:
\text{(Critical Section)}:
\text{(Exit)}:
\text{(Remainder)}: