Bounded algorithms for mutual exclusion

Mutual exclusion algorithm for 2 processes

**Algorithm 12** A bounded mutual exclusion algorithm for two processors: with no lockout.

Initially $Want[0]$ and $Want[1]$ and $Priority$ are all 0

<table>
<thead>
<tr>
<th>Code for $p_0$</th>
<th>Code for $p_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(Entry):</strong></td>
<td><strong>(Entry):</strong></td>
</tr>
<tr>
<td>1: $Want[0] := 0$</td>
<td>1: $Want[1] := 0$</td>
</tr>
<tr>
<td>2: wait until $(Want[1] = 0$ or $Priority = 0)$</td>
<td>2: wait until $Want[0] = 0$ or $Priority = 1$</td>
</tr>
<tr>
<td>4: if $(Priority = 1)$ then</td>
<td>4: if $(Priority = 0)$ then</td>
</tr>
<tr>
<td>5: if $(Want[1] = 1)$ then</td>
<td>5: if $(Want[0] = 1)$ then</td>
</tr>
<tr>
<td>\hspace{2em} goto Line 1</td>
<td>\hspace{2em} goto Line 1</td>
</tr>
<tr>
<td><strong>(Critical Section)</strong></td>
<td><strong>(Critical Section)</strong></td>
</tr>
<tr>
<td><strong>(Exit):</strong></td>
<td><strong>(Exit):</strong></td>
</tr>
<tr>
<td>7: $Priority := 1$</td>
<td>7: $Priority := 0$</td>
</tr>
<tr>
<td>8: $Want[0] := 0$</td>
<td>8: $Want[1] := 0$</td>
</tr>
<tr>
<td><strong>(Remainder)</strong></td>
<td><strong>(Remainder)</strong></td>
</tr>
</tbody>
</table>

**Lemma 1** In every configuration of every execution, if $p_i$ is in the critical section then $want[i]=1$.

**Theorem 2** The algorithm satisfies mutual exclusion.

Proof: Assume by contradiction that both processes are in the critical section in some configuration.

By Lemma 1 $want[0] = want[1] = 1$. In the initial configuration $want[0] = want[1] = 0$, so both processes must have written 1 to their want variable.

Assume that the last such write of $p_0$ was after the last such write of $p_1$. $p_0$ enters the critical section after reading $want[1] = 0$ either in line 5 or in line 6. In both cases, this read happens after writing $want[0] = 1$ which we assumed that happened after writing $want[1] = 1$ for the last time, which is a contradiction.
Theorem 3  The algorithm satisfies no-deadlock.

Proof: Consider an admissible execution and assume that there is a configuration C in which a process, say \( p_0 \), is in the entry, and in all the following configurations no process is in the critical section. Analyze according to 2 possible cases - either both processes are in the entry in the following configurations, or \( p_1 \) is in the remainder. In both cases show that a process enters the critical section.

Theorem 4  The algorithm satisfies no-starvation.

Proof: Consider an admissible execution and assume that a process, say \( p_0 \), is starved, i.e., it is always in the entry, starting from some configuration C. Analyze according to 2 possible cases - either \( p_1 \) executes line 7 in some configuration after \( C \), or \( p_1 \) does not execute line 7 after \( C \). In both cases show that \( p_0 \) enters the critical section.
Tournament tree algorithm for $n$ processes

**Algorithm 13** The tournament tree algorithm:

A bounded mutual exclusion algorithm for $n$ processors.

```plaintext
procedure Node(v: integer; side: 0..1)
1:  Want$^v$[side] := 0
2:  wait until (Want$^v[1-side] = 0 or Priority$^v = side$)
3:  Want$^v[side] := 1$
4:  if (Priority$^v = 1 - side$) then
5:      if (Want$^v[1-side] = 1$) then goto Line 1
6:      else wait until (Want$^v[1-side] = 0$)
7:  if (v = 1) then // at the root
8:      (Critical Section)
9:  else Node([v/2], v mod 2)
10:   Priority$^v := 1 - side$
11:   Want$^v[side] := 0$
end procedure
```

Each node has its own variables $want^v[0], want^v[1], priority^v$.

We define a projection of an execution on node $v$:
given an execution $\alpha = C_0, \phi_1, C_1, \phi_2, C_2, ...$
we define $\alpha_v = D_0, \tau_1, D_1, \tau_2, D_2, ...$ by induction:

**Basis:** $D_0$ is the initial configuration of the 2-process algorithm, i.e., $want^v[0] = want^v[1] = priority^v = 0$ and $q_0, q_1$ are in their initial local states.

**Induction step:** Assumes $\alpha_v$ is defined up to configuration $D_{i-1}$, and let $\phi_j$ be the i-th event in $\alpha$ that occurs in node $v$, say in Node($v, 0$). Let $\phi_j = k$, i.e., process $p_k$ executes $\phi_j$ in $\alpha$. Then $\tau_i$ is defined to be $0$ and $D_i$ is defined as follows:

* the local state of $q_0$ is the same as the local state of $p_k$ in $C_j$.
* the local state of $q_1$ is the same as its local state in $D_{i-1}$.
* $want^v[0], want^v[1], priority^v$ are as in $C_j$. 

3
**Lemma 5** For each $v$, $\alpha_v$ is an execution of the algorithm for 2 processes (in every configuration only one process is executing $\text{Node}(v,0)$, and same for $\text{Node}(v,1)$).

Proof: By induction on the height of the node (from the leaves upwards).

**Lemma 6** For each $v$, if $\alpha$ is admissible then $\alpha_v$ is admissible.

Proof: By induction on the depth of the node (from the root downwards).

**Theorem 7** The tournament tree satisfies mutual exclusion and no-starvation.

Proof: Mutual exclusion follows from Lemmas 5,6 and since the 2-process algorithm satisfies mutual exclusion (Theorem 2). No-starvation follows from Lemmas 5,6 and since the 2-process algorithm satisfies no-starvation (Theorem 4).