Some Modifications of Lockout-Free Mutual Exclusion Algorithms

Hironobu Kurumazaki, Minako Kawa, Yasuaki Nishitani, and Yoshihide Igarashi

Department of Computer Science, Gunma University, Kiryu, Japan 376-8515
E-mail:igarashi@comp.cs.gunma-u.ac.jp

Abstract

In this paper we propose two mutual exclusion algorithms in the asynchronous multi-writer/reader shared memory model. One is a modification of the $N$-process algorithm by Peterson, and the other is a modification of the tournament algorithm by Peterson and Fischer. For these modified algorithms, some processes have the advantage of access to the resource over other processes. We show the lockout-freedom of these modified algorithms by analyzing the time bounds for the trying region.

1 Introduction

Mutual exclusion is a problem of managing access to a single indivisible resource that can only support one user at a time. An early algorithm for the mutual exclusion problem was proposed by Dijkstra [5]. His algorithm guarantees mutual exclusion, but it does not guarantee the high-level fairness. Subsequent algorithms improve on the Dijkstra's algorithm by guaranteeing fairness to the different users [12, 13] and by weakening the type of shared memory [1, 2, 3, 4, 6, 7, 9, 10]. Books by Raynal [15] and by Lynch [11] contain a number of mutual exclusion algorithms and their variations.

In this paper we propose two mutual exclusion algorithms in the asynchronous multi-writer/reader shared memory model. Our algorithms are modifications of the $N$-process algorithm by Peterson [13] and the tournament algorithm by Peterson and Fischer [14] so that we allow priority of some users to access the resource. These modified algorithms guarantee the lockout-freedom. The lockout-freedom of these algorithms are proved by showing time bounds for spending in the trial region.

2 Preliminary

A user with access to the resource is modeled as being a critical region. When a user is not involved in any way with the resource, it is said to be in the remainder region. In order to gain admittance to its critical region, a user executes a trying protocol. The duration from the start of executing the trying protocol to the
entrance of the critical region is called the trying region. After the end of using the resource by a user, it executes an exit protocol. The duration of executing the exit protocol is called the exit region. Each user follows a cycle, moving from its remainder region to its trying region, then to its critical region, then to its exit region, and then back again to its remainder region. This cycle can be repeated.

The inputs to process $i$ from user $U_i$ are the $try_i$ action which means a request by $U_i$ for access to the resource, and the $exit_i$ action which means an announcement by $U_i$ that it is done with the resource. The outputs of process $i$ are $crit_i$, which means the granting of the resource to $U_i$, and $rem_i$ which tells $U_i$ that it can continue with the rest of its work.

The system to solve the mutual exclusion problem should satisfy the following conditions.

(1) There is no reachable system state in which more than one users are in the critical region.
(2) If at least one user is in the trying region and no user is in the critical region, then at some later point some user enters the critical region.
(3) If a user is in the exit region, then at some later point the user enters the remainder region.

Conditions (1), (2) and (3) above are called mutual exclusion, progress for the trying region, and progress for the exit region, respectively. The following conditions are called the lockout-freedom.

(1) If all users always return the resource, then any user that reaches the trying region eventually enters the critical region.
(2) Any user that reaches the exit region eventually enters the remainder region.

## 3 Modification of $N$-process algorithm

The $N$-process algorithm by Peterson is a lockout-free mutual exclusion algorithm using multi-writer/reader shared variables [13]. We modify this algorithm so that some users have advantage of easier access to the resource than other users.

The set of processes $\{1, 2, ..., n\}$ is divided into two disjoint groups, a low priority group $G_1$ with $i_1$ processes and a high priority group $G_2$ with $n - i_1$ processes. Without loss of generality we may assume that $G_1 = \{1, ..., i_1\}$ and $G_2 = \{i_1 + 1, ..., n\}$. We choose an appropriate level $l_1$ where $l_1$ should be between $0$ and $i_1 - 1$.

\[
\text{procedure } 2\text{priorityME}(G_1 = \{1, ..., i_1\}, G_2 = \{i_1 + 1, ..., n\}, l_1) \quad \{ \ 0 \leq l_1 \leq i_1 - 1 \ \}\]

\[
\text{shared variables}\]

\[
\text{for every } k \in \{1, \ldots, n - 1\}: \quad \text{turn}(k) \in \{1, \ldots, n\}, \text{initially arbitrary, writable and readable by all processes};
\]

\[
\text{for every } i \in \{1, \ldots, i_1\}: \quad \text{flag}(i) \in \{0, \ldots, n - 1\}, \text{initially 0, writable by } i \text{ and readable by all } j \neq i;
\]

\[
\text{for every } i \in \{i_1 + 1, \ldots, n\}: \quad \text{flag}(i) \in \{l_1, \ldots, n - 1\}, \text{initially } l_1, \text{ writable by } i \text{ and readable by all } j \neq i;
\]
process $i \{ 1 \leq i \leq i_1 \}$
input actions $\{ \text{inputs from user } U_i \text{ to process } i \} \text{: } try_i, \, exit_i;$
output actions $\{ \text{outputs to user } U_i \} \text{: } crit_i, \, rem_i;$

** Remainder region **

$try_i:$
if $1 \leq i \leq i_1$ then
for $k = 1$ to $l_1$ do begin
flag($i$) := $k$;
turn($k$) := $i$;
waitfor $[\forall j \neq i \ (1 \leq j \leq i_1) : \text{flag}(j) < k]$ or $[\text{turn}(k) \neq i]$ end;
for $k = l_1 + 1$ to $n - 1$ do begin
flag($i$) := $k$;
turn($k$) := $i$;
waitfor $[\forall j \neq i : \text{flag}(j) < k]$ or $[\text{turn}(k) \neq i]$ end;
crit$_i$;

** Critical region **

$exit_i:$
if $1 \leq i \leq i_1$ then flag($i$) := 0
else flag($i$) := $l_1$;
rem$_i$;

Assertion 1 In any execution by 2priorityME, for any $k, 1 \leq k \leq l_1$, there are at most $i_1 - k$ winners from $G_1$ at level $k$.

From Assertion 1 there are at most $(n - i_1) + (i_1 - l_1) = n - l_1$ processes can be at level $l_1$ in the trying region. Then we have the next assertion.

Assertion 2 In any execution by 2priorityME, for any $k, l_1 + 1 \leq k \leq n - 1$ there are at most $n - k$ winners at level $k$.

From Assertion 1 and Assertion 2 we have the next theorem.

Theorem 1 2priorityME satisfies mutual exclusion.

Let $l$ be an upper bound on the time between successive steps of each process, and let $c$ be an upper bound on the maximum time that a user spends in the critical region.

We can prove the following two lemmas.

Lemma 2 In 2priorityME, the time from when a process enters the level $l_1$ of the trying region until it enters the critical region is at most $2^{n-l_1-1}c + O(2^{n-l_1}nl)$.

Lemma 3 In 2priorityME, the time from when a process of $G_1$ enters of the trying region until it enters the critical region is at most $(2^{n-1} + 2^{l_1-1})c + O(2^{n-1}nl)$.

From the two lemmas above the following theorem is immediate.

Theorem 4 2priorityME is lockout-free.
We can generalize 2priorityME. We partition the set of processes into \( r \) disjoint sets. Without loss of generality we may assume that these groups are \( G_1 = \{1, \ldots, i_1\}, G_2 = \{i_1 + 1, \ldots, i_2\}, \ldots, G_r = \{i_{r-1} + 1, \ldots, n\} \). Each group \( G_j \) is associated with a level bound \( l_j \ (1 \leq j \leq r) \), where \( 0 \leq l_1 \leq i_1 - 1, l_1 \leq l_2 \leq i_2 - 1, \ldots, l_{r-1} \leq l_r = n - 1 \). For convenience, we let \( l_0 = 0 \).

\[
\text{procedure } r\text{priorityME }((G_1 = \{1, \ldots, i_1\}, l_1), (G_2 = \{i_1 + 1, \ldots, i_2\}, l_2), \ldots, (G_r = \{i_{r-1} + 1, \ldots, n\}, l_r))
\]

\[\text{shared variables}\]
\[\text{for every } k \in \{1, \ldots, n - 1\}:\]
\[\text{turn}(k) \in \{1, \ldots, n\}, \text{initially arbitrary, writable and readable by all processes};\]

\[\text{for every } i \in G_j:\]
\[\text{flag}(i) \in G_j, \text{initially } l_j - 1, \text{writable by } i \text{ and readable by all } j \neq i;\]

\[\text{process } i \{ i \in G_i \}\]
\[\text{input actions } \{ \text{inputs from user } U_i \text{ to process } i \}: \text{try}_i, \text{exit}_i;\]
\[\text{output actions } \{ \text{outputs to user } U_i \}: \text{crit}_i, \text{rem}_i;\]

**Remainder region**

\[\text{try}_i;\]
\[\text{for } s := t \text{ to } r \text{ do}\]
\[\text{for } k := l_{s-1} \text{ to } l_s \text{ do begin}\]
\[\text{flag}(i) := k;\]
\[\text{turn}(k) := i;\]
\[\text{waitfor } [\forall j \neq i \ (1 \leq j \leq i_s) : \text{flag}(j) < k ] \text{ or } [ \text{turn}(k) \neq i ];\]
\[\text{end};\]

**Critical region**

\[\text{crit}_i;\]

\[\text{exit}_i;\]

\[\text{for } j = 1 \text{ to } r \text{ do}\]
\[\text{if } i \in G_j \text{ then } \text{flag}(i) := l_j - 1;\]

\[\text{rem}_i;\]

**Assertion 3** In any execution by \( r\text{priorityME} \), for any \( j, 1 \leq j \leq r \) and any \( k, l_{j-1} + 1 \leq k \leq l_j \), there are at most \( i_j - k \) winners from \( G_1 \cup \ldots \cup G_j \) at level \( k \).

**Lemma 5** In \( r\text{priorityME} \), for any \( j \ (1 \leq j \leq r) \) the time from when a process in \( G_j \) enters the trying region until it enters the critical region is at most \( 2^{n-l_j-1} + 2^{l_{r-1}-l_j-1} + \ldots + 2^{l_{j-1}-l_j-1} + O(2^{n-l_j-1}n) \).

From Assertion 3 and Lemma 5 we have the following theorem.

**Theorem 6** \( r\text{priorityME} \) solves the mutual exclusion problem and is lockout-free.

4 **Tournaments on priority trees**

We modify the tournament algorithm of Peterson and Fischer [14] so that some users have priority over some other users in getting access to the resource.

A simple priority tree is a binary tree structure recursively defined as follows:
it consists of a single node, or
(2) it is composed of three disjoint sets of nodes, a root node, a single node as its left subtree, and a simple priority tree as its right subtree.

Each node of a binary tree is labelled by the following rules.

(1) The root is labelled by \( \lambda \) (the null string).
(2) If the label of a node \( x \) is \( l(x) \), the label of its left son is \( l(x)0 \) (i.e., the juxtaposition of \( l(x) \) and 0) and the label of its right son is \( l(x)1 \).

Suppose that \( 2^{r+1} \leq N \). Let \( a = \lfloor \frac{N}{2^r} \rfloor - 1 \) and \( r' = \lceil \log_2(N - 2^r a) \rceil \). A priority tree \( T(N, r) \) is a binary tree constructed as follows:

(1) Let \( T_s(N, r) \) be a simple priority tree with leaves labelled with \( 0, 10, \ldots, 1^{a-1}0, 1^a \).
(2) Each leaf of \( 0, 10, \ldots, 1^{a-1}0, 1^a \) of \( T_s(N, r) \) is replaced with the complete binary tree with \( 2^r \) leaves, and leaf \( 1^a \) is replaced with an essentially complete binary tree with \( N - 2^r a \) leaves.

We consider a one-to-one correspondence between the \( N \) processes and the \( N \) leaves of \( T(N, r) \). The label associated with a process in \( T(N, r) \) is called the index of the process. We denote the complete binary tree and the essentially complete binary tree that are replacements at the leaves of \( T_s(N, r) \) by \( G_0, G_1, \ldots, G_{a-1} \) and \( G_a \) from left to right (see Figure 1).

Figure 1: A priority tree.

For \( T(N, r) \) and each process \( i \), we introduce the following notations.

- \( \text{comp}(i, k) \) is the ancestor of \( i \) in depth \( k \).
- \( \text{role}(i, k) \) is the \((k+1)\)st high-order bit of \( i \). (i.e., \( \text{role}(i, k) \) indicates whether the leaf \( i \) is a descendant of the left or right child of the node for \( \text{comp}(i, k) \)).
- \( \text{opponents}(i, k) \) is the opponents of process \( i \) in the depth \( k \) competition of process \( i \) (i.e., the set of process indices with the high-order \( k \) bits as \( i \) and the opposite \((k+1)\)st bit.)

```
procedure tournamentME (N, r)
shared variables
for every binary string \( x \) in the set of labels of \( T(N, r) \):
```
\( \text{turn}(x) \in \{0, 1\} \), initially arbitrary, writable and readable by those processes \( i \) for which \( x \) is a prefix of the index of \( i \);

for every \( i \) in the set of leaves of \( T(N, r) \):

\( \text{flag}(i) \in \{0, 1, \ldots, d(i)\} \), initially \( d(i) \), writable by \( i \) and readable by all \( j \neq i \) in the set of leaves of \( T(N, r) \), where \( d(i) \) is the depth of \( i \);

**Process** \( i \{ i \text{ is a leaf of } G_t \} \)

**Input actions** \{ inputs from user \( U_i \) to process \( i \) \}: \( \text{try}_i, \text{exit}_i \);

**Output actions** \{ outputs to user \( U_i \) \}: \( \text{crit}_i, \text{rem}_i \);

**Remainder region**

\( \text{try}_i; \)

\( \text{for } k = d(i) - 1 \text{ downto } 0 \text{ do } \{ d(i) = t + r + 1 \text{ if } t \leq a - 1, \text{ and otherwise } d(i) = t + r' \} \)

\( \text{begin} \)

\( \text{flag}(i) := k; \)

\( \text{turn} \text{ comp}(i, k) := \text{role}(i, k); \)

\( \text{waitfor } [ \forall j \in \text{opponents}(i, k) : \text{flag}(j) > k ] \text{ or } [ \text{turn} \text{ comp}(i, k) \neq \text{role}(i, k) ] \)

\( \text{end}; \)

\( \text{crit}_i; \)

**Critical region**

\( \text{exit}_i; \)

\( \text{flag}(i) := d(i); \)

\( \text{rem}_i; \)

**Assertion 4** In any reachable system state by tournamentME on \( T(N, r) \), and for any depth \( k \), \( 0 \leq k \leq a + r' - 1 \), at most one process in depth \( k \) in any subtree rooted in depth \( k \) is a winner, where \( a = \lfloor \frac{N}{2^r} \rfloor - 1 \) and \( r' = \lfloor \log_2(N - 2^a) \rfloor \).

From the two assertions above, the next theorem is immediate.

**Theorem 7** tournamentME satisfies mutual exclusion.

As in the previous section, let \( l \) and \( c \) be upper bounds on process step time and critical region time, respectively. We show a time bound for tournamentME in the following lemma.

**Lemma 8** In tournamentME on a \( T(N, r) \), \( N = 2^r(a + 1) \), the time from when a process \( i \) in the set of leaves of \( G_t \) has just entered the trying region until it enters the critical region is at most \( (c + 4l)(2^{l+r+1} + 2l + 2^r + 1) + (t + r + 1 - a)2^r l \).

**Proof.** For \( k, 0 \leq k \leq a + r - 1 \), define \( T(k) \) to be the maximum time from when a process \( i \) wins in depth \( k \) or it has just entered the trying region in depth \( k \) (this event is denoted by \( \pi_i(k) \)) until it enters the critical region. It is immediate that \( T(0) \leq l \) since only one step is needed to enter the critical region after winning the final competition.

We can consider the following two cases just after event \( \pi_i(k) \). One is the case where \( i \) is a winner at a node \( 1^s \) for some \( s (1 \leq s \leq a) \), and the other is the case where \( i \) is a winner at a node that is not a node \( 1^s \) for any \( s (1 \leq s \leq a) \). In the former case, within at most time \( ((a-k+1)2^r + 3)l + c + T(k - 1) \) after \( \pi_i(k) \), for
every $j$ in \textit{opponents}(i, k), \text{flag}(j) > k$ holds or \textit{turn}(\text{comp}(i, k)) be set to be not equal to \textit{role}(i, k). In the latter case, within at most time $(2^{r}+3)l+c+T(k-1)$ after $\pi_{i}(k)$, for every $j$ in \textit{opponents}(i, k), \text{flag}(j) > k$ holds or \textit{turn}(\text{comp}(i, k)) be set to be not equal to \textit{role}(i, k). Then, within at most $2Tl$ in the former case and within at most time $(a-k+1)2^{r}l$ in the latter case, process $i$ moves up one level as a winner in depth $k-1$. Hence, the total time from event $\pi_{i}(k)$ until process $i$ arrives at the entrance to the critical region is at most $2T(k-1) + c + ((a-k+2)2^{r} + 3)l$. Thus, we need to solve the following recurrence for $T(d(i))$.

\[
\begin{align*}
T(0) & \leq l \\
T(k) & \leq 2T(k-1) + c + ((a-k+2)2^{r} + 3)l
\end{align*}
\]

Then we can derive the following inequality.

\[
T(k) \leq (c + 3l)(1 + 2 + 2^{2} + \cdots + 2^{k-1}) + 2^{k}l + (2^{k}a - a + k)2^{r}l
\]  
\[
\leq (c + 4l)2^{k} + 2^{k+r}al + (k-a)2^{r}l.
\]

For $0 \leq t < a - 1$, $T(d(i)) \leq (c + 4l)2^{t+r+1} + 2^{t+2r+1}al + (t + r + 1 - a)2^{r}l$, and for $t = a$, $T(d(i)) \leq (c + 4l)2^{t+r} + 2^{t+2r}al + (t + r - a)2^{r}l$. Thus, the lemma holds.

We have the following theorem from Theorem 7 and Lemma 8.

\textbf{Theorem 9} tournamentME solves the mutual exclusion problem and is lockout-free.

A speed-up version of tournamentME is given in [8]. If we use the speed-up version on the complete binary tree with $n$ leaves, its running time is $(n-1)c+O(nl)$ [8]. This is an improvement over the original tournament algorithm [11, 14] whose running time is $(n-1)c + O(n^{2}l)$.

5 Concluding remarks

There may be a natural request to design a distributed operating system such that some processes have advantage of access in some degree to the resource over other processes. We have proposed two such mutual exclusion algorithms in shared memory model. One is a modification of the $N$-process algorithm by Peterson, and the other is a modification of the tournament algorithm by Peterson and Fischer. We show the lockout-freedom of these algorithms by analyzing time bounds for trying region. The time bounds shown in this paper seem not to be tight. We need a finer analysis to derive better time bounds for these algorithms. In order to obtain a substantial improvement of the time efficiency, further modifications of these algorithms will be needed. The mutual exclusion algorithms given in this paper do not guarantee the FIFO property. Namely, even if a process in a high priority group enters the trying region earlier than a process in a low priority group, the latter process may catch up the former process and it enters the critical region earlier than the process in the high priority group. For our purpose, we need a mutual exclusion algorithm that guarantees the advantage of access to the resource in a stronger sense. Designing such algorithms is also worthy for farther investigation.
References


