数理解析研究所講究録
1041 巻 1998 年 19-26

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 $i_1 - 1$.

procedure *2priorityME*$(G_1 = \{1, ..., i_1\}, G_2 = \{i_1 + 1, ..., n\}, l_1)$
$\{\ 0 \leq l_1 \leq i_1 - 1\ \}$

shared variables
for every $k \in \{1, \ldots, n-1\}$:
turn$(k) \in \{1, \ldots, n\}$, initially arbitrary, writable and readable by all processes;
for every $i \in \{1, \ldots, i_1\}$:
flag$(i) \in \{0, \ldots, n-1\}$, initially 0, writable by i and readable by all $j \neq i$;
for every $i \in \{i_1 + 1, \ldots, n\}$:
flag$(i) \in \{l_1, \ldots, n-1\}$, initially l_1, writable by i and readable by all $j \neq i$;

```
process i { 1 ≤ i ≤ i1 }
  input actions { inputs from user U_i to process i }: try_i, exit_i;
  output actions { outputs to user U_i }: crit_i, rem_i;

  ** Remainder region **

  try_i:
   if 1 ≤ i ≤ i1 then
     for k = 1 to l1 do begin
       flag(i) := k;
       turn(k) := i;
       waitfor [ ∀j ≠ i (1 ≤ j ≤ i1) : flag(j) < k ] or [ turn(k) ≠ i ]
     end;
   for k = l1 + 1 to n − 1 do begin
     flag(i) := k;
     turn(k) := i;
     waitfor [ ∀j ≠ i : flag(j) < k ] or [ turn(k) ≠ i ]
   end;
  crit_i;

  ** Critical region **

  exit_i:
   if 1 ≤ i ≤ i1 then flag(i) := 0
   else flag(i) := l1;
  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 $G_1 = \{1,\ldots,i_1\}$, $G_2 = \{i_1+1,\ldots,i_2\}$, ..., $G_r = \{i_{r-1}+1,\ldots,n\}$. Each group G_j is associated with a level bound l_j $(1 \le j \le r)$, where $0 \le l_1 \le i_1 - 1$, $l_1 \le l_2 \le i_2 - 1$, ..., $l_{r-1} \le l_r = n-1$. For convenience, we let $l_0 = 0$.

procedure *rpriorityME* $(\langle G_1 = \{1, \ldots, i_1\}, l_1\rangle, \langle G_2 = \{i_1+1, \ldots, i_2\}, l_2\rangle, \ldots, \langle G_r = \{i_{r-1}+1, \ldots, n\}, l_r\rangle)$

shared variables
  for every $k \in \{1, \ldots, n-1\}$:
    $turn(k) \in \{1, \ldots, n\}$, initially arbitrary, writable and readable by all processes;
  for every $j \in \{1, \ldots, r\}$:
    for every i in G_j
      $flag(i) \in G_j$, initially l_{j-1}, writable by i and readable by all $j \ne i$;

process i $\{ i \in G_t \}$
  input actions { inputs from user U_i to process i }: try_i, $exit_i$;
  output actions { outputs to user U_i }: $crit_i$, rem_i;

  ** Remainder region **

  try_i:
    for $s := t$ **to** r **do**
      for $k := l_{s-1}$ **to** l_s **do begin**
        $flag(i) := k$;
        $turn(k) := i$;
        waitfor [$\forall j \ne i$ $(1 \le j \le i_s) : flag(j) < k$] or [$turn(k) \ne i$]
      end;
  $crit_i$;

  ** Critical region **

  $exit_i$:
    for $j = 1$ **to** r **do**
      if $i \in G_j$ **then** $flag(i) := l_{j-1}$;
  rem_i;

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

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

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

Theorem 6 *rpriorityME 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:

(1) 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 λ (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 $0, 10, \ldots, 1^{a-1}0$, 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).

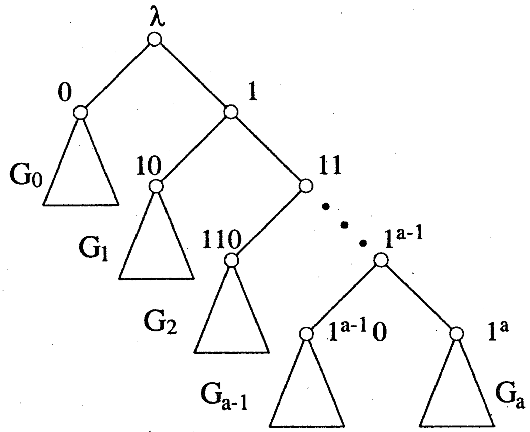

Figure 1: A priority tree.

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

- *comp*(i, k) is the ancestor of i in depth k.
- *role*(i, k) is the $(k+1)$st high-order bit of i. (i.e., *role*(i, k) indicates whether the leaf i is a descendant of the left or right child of the node for *comp*(i, k)).
- *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 $T(N, r)$:

$turn(x) \in \{0, 1\}$, 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 $T(N, r)$:
$flag(i) \in \{0, 1, \ldots, d(i)\}$, 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 is a leaf of G_t }
input actions { inputs from user U_i to process i }: try_i, $exit_i$;
output actions { outputs to user U_i }: $crit_i$, rem_i;

** Remainder region **

try_i:
for $k = d(i) - 1$ **downto** 0 **do** { $d(i) = t + r + 1$ if $t \leq a - 1$, and otherwise $d(i) = t + r'$ }
begin
$flag(i) := k$;
$turn(comp(i, k)) := role(i, k)$;
waitfor [$\forall j \in opponents(i, k) : flag(j) > k$] or [$turn(comp(i, k)) \neq role(i, k)$]
end;
$crit_i$;

** Critical region **

$exit_i$:
$flag(i) := d(i)$;
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 $r' = \lceil \log_2(N - 2^r a) \rceil$.*

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 $(c + 4l)2^{t+r+1} + 2^{t+2r+1}al + (t + r + 1 - a)2^r l$.*

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 $opponents(i,k)$, $flag(j) > k$ holds or $turn(comp(i,k))$ be set to be not equal to $role(i,k)$. In the latter case, within at most time $(2^r+3)l+c+T(k-1)$ after $\pi_i(k)$, for every j in $opponents(i,k)$, $flag(j) > k$ holds or $turn(comp(i,k))$ be set to be not equal to $role(i,k)$. Then, within at most time $2^r l$ 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 $T(d(i))$.

$$\begin{aligned}
T(0) &\leq l \\
T(k) &\leq 2T(k-1)+c+((a-k+2)2^r+3)l
\end{aligned}$$

Then we can derive the following inequality.

$$\begin{aligned}
T(k) &\leq (c+3l)(1+2+2^2+\cdots+2^{k-1})+2^k l+(2^k a-a+k)2^r l \\
&\leq (c+4l)2^k+2^{k+r}al+(k-a)2^r l.
\end{aligned}$$

For $0 \leq t \leq a-1$, $T(d(i)) \leq (c+4l)2^{t+r+1}+2^{t+2r+1}al+(t+r+1-a)2^r l$, 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.

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

[1] J.E.Burns, "Mutual exclusion with linear waiting using binary shared variables", *ACM SIGACT News*, vol.10, pp.42–47, 1978.

[2] J.E.Burns, P.Jackson, N.A.Lynch, M.J.Fischer, and G.L.Peterson, "Data requirements for implementation of *N*-process mutual exclusion using a single shared variable", *J. of the ACM*, vol.29, pp.183–205, 1982.

[3] J.E.Burns, and N.A.Lynch, "Bounds on shared memory for mutual exclusion", *Information and Computation*, vol.107, pp.171-184, 1993.

[4] A.B.Cremers and T.N.Hibbard, "Mutual exclusion of N processors using an $O(N)$-valued message variable", *5th International Colloquium on Automata, Languages and Programming*, Udine Italy, *Lecture Notes in Computer Science*, vol.62, pp.165–176, 1978.

[5] E.W.Dijkstra, "Solution of a problem in concurrent programming control", *Communications of the ACM*, vol.8, p.569, 1965.

[6] M.J.Fischer, N.A.Lynch, J.E.Burns, and A.Borodin, "Resource allocation with immunity to limited process failure", *20th Annual Symposium on Foundations of Computer Science*,San Juan, Puerto Rico, pp.234–254, 1979.

[7] M.J.Fischer, N.A.Lynch, J.E.Burns, and A.Borodin, "Distributed FIFO allocation of identical resources using small shared space", *ACM Trans. on Programming Languages and Systems*, vol.11, pp.90–114, 1989.

[8] H.Kurumazaki, M.Kawa, Y.Nishitani, Y.Igarashi "Some Modifications of Lockout-Free Mutual Exclusion Algorithms", *To appear in SIGAL-TR*, Information Processing Society of Japan, May, 1998

[9] L.Lamport, "The mutual exclusion problem. Part II : Statement and solutions", *J. of the ACM*, vol.33, pp.327–348, 1986.

[10] L.Lamport, "A new solution of Dijkstra's concurrent programming problem", *Communications of the ACM*, vol.17, pp.453–455, 1974.

[11] N.A.Lynch, "Distributed Algorithms", *Morgan Kaufmann*, San Francisco, California, 1996.

[12] N.A.Lynch and M.J.Fischer, "On describing the behavior and implementation of distributed systems", *Theoretical Computer Science*, vol.13, pp.17–43, 1981.

[13] G.L.Peterson, "Myths about the mutual exclusion problem", *Information Processing Letters*, vol.12, pp.115–116, 1981.

[14] G.L.Peterson and M.J.Fischer, "Economical solutions for the critical section problem in a distributed system", *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, Boulder, Colorado, pp.91–97, 1977.

[15] M.Raynal, "Algorithms for Mutual Exclusion", *MIT Press*, Cambridge, Massachusetts, 1986.