アルゴリズム 62−8
（1998. 5. 20）

ロックアウトフリーな相互排除アルゴリズムの速度改善

五十嵐 善英、 車崎 裕信、永藤広正、 西谷 泰昭
群馬大学工学部情報工学科
〒 376-8515 群馬県桐生市天神町１ − ５ − １
igarashi@comp.cs.gunma-u.ac.jp

非同期 multi-writer/reader 共有メモリーモデルにおけるよく知られたロックアウトフリーな相互排除アルゴリズムの速度改善を提案し、その正当性と効率を示す。初めの二つのアルゴリズムは Peterson による N-プロセスアルゴリズムの変更であり、三番目のアルゴリズムは、Peterson と Fischer によるトーナメントアルゴリズムの変更である。これらのアルゴリズムの trying region の時間上限は、それぞれ $(2n-3)c+O(n^3l)$, $(n-1)c+O(n^3l)$, $(n-1)c+O(nl)$ となる。ここで、n はプロセスの数、c は任意のプロセスが critical region で過ごす時間の上限、l は一つのプロセスの連続する二つのステップの時間の上限である。
キーワード: 共有メモリ、相互排除、分散アルゴリズム、ロックアウトフリー

Speedup of Lockout-Free Mutual Exclusion Algorithms

Yoshihide Igarashi, Hironobu Kurumazaki, Hiroshi Nagafuji,
and Yasuaki Nishitani
Department of Computer Science,Gunma University,Kiryu, Japan 376-8515
E-mail:igarashi@comp.cs.gunma-u.ac.jp

Abstract

We propose three lockout-free mutual exclusion algorithms for the asynchronous multi-writer/reader shared memory model. The first two algorithms are modifications of the n-process algorithm by Peterson, and the third algorithm is a modification of the tournament algorithm by Peterson and Fischer. The correctness and efficiency of these modified algorithms are shown. By these modifications we can speed up the original algorithms. The running times for the trying regions of the first algorithm and the second algorithm are $(2n-3)c+O(n^3l)$ and $(n-1)c+O(n^3l)$, respectively, where n is the number of processes, l is an upper bound on the time between successive two steps, and c is an upper bound on the time that any user spends in the critical region. The running time for the trying region of the third algorithm is $(n-1)c+O(nl)$.
key words: distributed algorithms, lockout-free, mutual exclusion, shared memory

1 Introduction

An early algorithm for the mutual exclusion problem was proposed by Dijkstra [4]. The Dijkstra's algorithm guarantees mutual exclusion, but it does not guarantee high-level fairness. Subsequent algorithms are improvements on the Dijkstra's algorithm by guaranteeing fairness to the different users [10, 11] and by weakening the type of shared memory [1, 2, 3, 5, 6, 7].

In this paper we propose some modifications of two well-known mutual exclusion algorithms, the n-process algorithm by Peterson [10] and the tournament algorithm by Peterson and Fischer [11] for the asynchronous multi-writer/reader shared memory model.

In order to estimate an upper bound on the running time of a mutual exclusion algorithm, we impose an upper bound of l on the time between successive steps of each process in the trying region and the exit region, and an upper bound of c on the time that any user spends in its critical region. The bounds on the running times for the trying regions of the first modified n-process algorithm and of the second modified n-process algorithm are $(2n-3)c+O(n^3l)$ and $(n-1)c+O(n^3l)$, respectively, whereas the bound on the corresponding running time of the original n-process algorithm is $O(n^2c+n^4l)$. The bound on the running time for the trying region of the modified tournament algorithm is $(n-1)c+O(nl)$, whereas the bound of the corresponding running time of the original tournament algorithm is $(n-1)c+O(n^2l)$.

2 Preliminary

The computation model used in this paper is the asynchronous multi-writer/reader shared memory model. Each process is considered to be a state machine. All communication among the processes is via the shared memory. We assume that even if two different processes try to write on the same shared variable at almost the same time, one process's writing is earlier than the other process's writing. The reader can find a complete and formal description of the model in [8, 9].

The mutual exclusion problem is the problem how to allocate a single indivisible, nonsharable resource among n users, $U_1, ..., U_n$. A user with access to the resource is modeled as being in a critical region. When a user is not involved in anyway 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 the use of 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.

We assume that the n processes are numbered $1, \cdots, n$ in Section 3, each process i corresponding to user U_i $(1 \leq i \leq n)$, and numbered $0, \cdots, n-1$ in Section 4, each process i corresponding to user U_i $(0 \leq i \leq n-1)$. The inputs to process i from user U_i are try_i which means a request by U_i for access to the resource, and $exit_i$ which means an announcement of the end of the use of the resource by U_i. The outputs from process i to user U_i are $crit_i$ which means the grant of the resource to U_i, and rem_i which tells U_i that it can continue with the rest of its work. These are external actions of the shared memory system.

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 Dijkstra's mutual exclusion algorithm guarantees mutual exclusion [4]. However, it may allow one user to be repeatedly granted for access to its critical region while other users trying to gain access never succeed in doing so. This situation is called lockout or starvation. Lockout is an undesirable property. If a mutual exclusion algorithm satisfies the following two conditions, it is said to be lockout-free.

(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 Speedup of the n-process algorithm

The following procedure *n-processME* is a well-known lockout-free mutual exclusion algorithm, known as the n-process algorithm by Peterson [10]. In order to analyze the efficiency of the algorithm we use the program described on the I/O automata model given in [8].

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

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

  ** Remainder region **
  $try_i$:
    for $k = 1$ to $n-1$ do
      begin
        $flag(i) := k$;
        $turn(k) := i$;
        waitfor $[\forall j \neq i : flag(j) < k]$ or $[turn(k) \neq i]$
      end;
  $crit_i$;
  ** Critical region **
  $exit_i$:
    $flag(i) := 0$;
  $rem_i$;
```

In *n-processME* and subsequent algorithms, process i is said to be a winner at level k if it has left the *waitfor* statement in the kth loop of the **for** statement. Note that if a process is a winner at level k then for any $1 \leq t \leq k$, the process is also a winner at level t. An exponential time bound, $2^{n-1}c + O(2^n nl)$ for the trying region of the *n-process* algorithm is given in [8]. A finer analysis can give a polynomial complexity time bound for the trying region of *n-processME*.

Theorem 1 *In n-processME, the time from when a particular process enters its trying region until it enters its critical region is at most $O(n^2c + n^4nl)$.*

The following procedure *n-processFME1* is an algorithm modified from *n-processME*.

```
procedure n-processFME1
shared variables: the same as the shared variables of n-processME
process $i$
  input/output actions: the same as the input/output actions of n-processME

  ** Remainder region **
  $try_i$:
    for $k = 1$ to $n-1$ do
      begin
        $flag(i) := k$;
        $turn(k) := i$;
        waitfor $[\forall j \neq i : flag(j) \notin \{k, k+1\}]$ or $[turn(k) \neq i]$
        end;
    $crit_i$;
    ** Critical region **
    $exit_i$:
      $flag(i) := 0$;
    $rem_i$;
```

Note that *n-processFME1* is different from *n-processME* at the first condition of the *waitfor* statement. We can prove the following assertion.

Assertion 1 *In any reachable system state of n-processFME1, for any k, $1 \leq k \leq n-1$, there are at most $n-k$ winners at level k.*

By Assertion 1, *n-processFME1* guarantees mutual exclusion. We now analyze the running time for the trying region of *n-processFME1*.

Theorem 2 *In n-processFME1, the time from when a particular process has just entered its trying region until it enters its critical region is at most $(2n-3)c + O(n^3 l)$.*

Proof. For each k, $1 \leq k \leq n-1$, define $F(k)$ to be the maximum time from when a process enters the competition at level k of the trying region until it becomes a winner at level k. The worst situation for a process at level $n-1$ to reach the critical region is the case where two processes have just entered the competition at level $n-1$ and then the process becomes a loser at level $n-1$. In this case, the time to decide a winner at level $n-1$ is at most $(n+2)l$, and the time from the end of the competition at level $n-1$ until the winner resets its $flag$ to 0 in the exit region is at most $3l + c$. Then the loser becomes a winner within at most nl after the reset of the winner's $flag$. Hence, $F(n-1) \leq (2n+5)l + c$.

When there are at least two competitors at level k, the time to decide winners among the competitors at level k is at most $(n+2)l$. A loser in the competition at level $n-2$ becomes a winner at level $n-2$ when a new process joins the competition at level $n-2$, or when all the winners at level $n-2$ have reset their $flag$ values to 0 in their exit regions. We should note that the $flag$ value of any winner at level $n-1$ keeps its value until it resets the $flag$ value to 0 in the exit region. Hence, $F(n-2) \leq (n+2)l + F(n-1) + c + 3l + nl = F(n-1) + (2n+5)l + c$. For $1 \leq k \leq n-3$, a loser at level k becomes a winner when a new process joins the competition at level k, or when all the winners at level k become winners at level $k+1$. Note that a process at level $k \leq n-3$ sets its $flag$ value to $k+1$ just after it wins the competition at level k. Hence, for $1 \leq k \leq n-3$, $F(k) \leq F(k+1) + 2(n+1)l$.

Then the time from when a process has just entered its trying region until it enters its critical region is bounded by

$$\sum_{k=1}^{n-1} F(k) \leq ((2n+5)l + c)(2n-3) + 2(n+1)l \sum_{k=1}^{n-3} k \leq (2n-3)c + O(n^3 l).$$

□

The progress in the exit region of *n-processFME1* is obvious. Hence, from Assertion 1 and Theorem 2 we have the following theorem.

Theorem 3 *n-processFME1 solves the mutual exclusion problem and is lockout-free.*

The following algorithm, *n-processFME2* is also obtained from *n-processME* by another modification, adding statement **for** $k := n-1$ **downto** 1 **do** $turn(k) := i$ in the exit region of each process i.

```
procedure n-processFME2
shared variables: the same as the shared variables of n-processME
 process i
  input/output actions: the same as the input/output actions of n-processME
   ** Remainder region **
  try_i:
    for $k = 1$ to $n - 1$ do
      begin
        $flag(i) := k$;
        $turn(k) := i$;
        waitfor $[\forall j \neq i : flag(j) < k]$ or $[turn(k) \neq i]$
      end;
  crit_i;
  ** Critical region **
  exit_i:
    for $k = n - 1$ downto 1 do
      $turn(k) := i$;
    $flag(i) := 0$;
  rem_i;
```

Assertion 2 *In any reachable system state of n-processFME2, for any k, $1 \leq k \leq n - 1$, there are at most $n - k$ winners at level k.*

By Assertion 2, *n-processFME2* guarantees mutual exclusion. We next analyze the running time for the trying region of *n-processFME2*.

Theorem 4 *In n-processFME2, the time from when a particular process has just entered its trying region until it enters its critical region is at most $(n-1)c + O(n^3 l)$.*

Proof. For each k, $1 \leq k \leq n-1$, define $F(k)$ to be the maximum time from when a process enters the competition at level k of the trying region until it becomes a winner at level k.

Consider an event $\pi(i,k)$ that process i has just entered level k of the trying region. If just after $\pi(i,k)$ and during the checks of the $flag$ values by process i at level k, for all $j \neq i$, $flag(j) < k$, then process i becomes a winner at level k within at most time $(n+2)l$.

Suppose that for some $j \neq i$, $flag(j) \geq k$ just after $\pi(i,k)$ or during the checks of the $flag$ values by process i at level k. Assume that among the processes $j \neq i$ with $flag(j) \geq k$ after $\pi(i,k)$, process t is the first process that reaches the critical region. Then process t reaches the critical region within at most time $(n+2)(n-k)l$ after $\pi(i,k)$ and $turn(k)$ is reset within at most time $c + (n - k + 1)l$ after the end of the critical region of process t. Hence, $F(k) \leq c + [(n+2)(n-k) + (2n-k+1)]l$.

Thus the time from when a process has just entered the trying region until it enters its critical region is bounded by

$$\sum_{k=1}^{n-1} F(k) \leq \sum_{k=1}^{n-1} [c + (n^2 + (4-k)n - 3k + 1)l] \leq (n-1)c + O(n^3 l).$$

□

The progress in the exit region of *n-processFME2* is obvious. Hence, from Assertion 2 and Theorem 4 we have the following theorem.

Theorem 5 *n-processFME2 solves the mutual exclusion problem and is lockout-free.*

We can realize an execution by *n-processFM1* and an execution by *n-processFM2* that show the time bounds given in Theorem 2 and Theorem 4, respectively to be tight. Comparing the time bound given in Theorem 1 with these time bounds, we can say that both *n-processFME1* and *n-processFME2* are substantially faster than *n-processME.*

4 Speedup of the tournament algorithm

In this section we modify the tournament algorithm by Peterson and Fischer [11]. For simplicity, we assume that the number of processes, n is a power of 2. The algorithm is described on the complete binary tree with n leaves, called an n-leaves binary tournament tree. We number the n processes as $0, \cdots, n-1$ rather than $1, \cdots, n$. The leaves of the complete binary tree are labeled as $0, \cdots, n-1$ in binary representation from left to right. All logarithms in this section are to the base 2. Each internal node at level k of the complete binary tree, $1 \leq k \leq \log n$, is labeled as the high-order $\log n - k$ bits of the binary representation of any of its descendants. Note that the root of the tree is labeled as λ, the null length string. The following notations will be used to described the tournament algorithm of Peterson and Fischer, and the modified tournament algorithm.

- $comp(i, k)$ is the ancestor of i at level k (i.e., the high order $\log n - k$ bits of the binary representation of i).
- $role(i, k)$ is the $(\log n - k + 1)$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 at the level k competition of process i (i.e., the set of processes with the high-order $\log n - k$ bits as i and the opposite $(\log n - k + 1)$st bit.)
- $opposite(i, k)$ is the son of $comp(i, k)$ that is not an ancestor of i.

The following procedure, *n-tournamentME* is the tournament algorithm by Peterson and Fischer quoted from [8].

```
procedure n-tournamentME

shared variables
  for every binary string $x$ of length at most $\log n - 1$:
    $turn(x) \in \{0,1\}$, initially arbitrary, writable and readable by those processes $i$
    for which $x$ is a prefix of the binary representation of $i$;
  for every $i$, $0 \leq i \leq n-1$:
    $flag(i) \in \{0, 1, \ldots, \log n\}$, initially 0, writable by $i$ and readable by all $j \neq i$
process $i$:
  input actions {inputs to process $i$ from user $U_i$}: $try_i$, $exit_i$;
  output actions {outputs from process $i$ to user $U_i$}: $crit_i$, $rem_i$;
   ** Remainder region **
  $try_i$:
    for $k = 1$ to $\log n$ do
      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 **
  $exit_i$:
    $flag(i) := 0$;
  $rem_i$;
```

Theorem 6 [8] *In n-tournamentME, the time from when a particular process i enters its trying region until it enters its critical region is at most* $(n-1)c + O(n^2 l)$.

The following procedure, *n-tournamentFME* is a modification of *n-tournamentME.* Note that *n-tournamentFME* uses about twice as many $flag$ variables as $flag$ variables of *n-tournamentME.*

```
procedure n-tournamentFME
shared variables
  for every binary string $x$ of length at most $\log n - 1$:
    $turn(x) \in \{0,1\}$, initially arbitrary, writable and readable by those processes $i$
    for which $x$ is a prefix of $i$;
  for every binary string $x$ of length at most $\log n$:
    $flag(x) \in \{0,1\}$, initially 0, writable by those processes $i$ for which $x$ is a prefix
    of $i$, and readable by those processes for which $i$ is a descendant of $x$'s parent
    but $i$'s bit at the position corresponding to the last bit of $x$ is opposite from $x$;
process $i$
  input/output actions: the same as the input/output actions of n-tournamentME
  ** Remainder region **
  $try_i$:
    for $k = 1$ to $\log n$ do
      begin
        $flag(comp(i, k-1)) := 1$;
        $turn(comp(i, k)) := role(i, k)$
        waitfor $[flag(opposite(i, k)) = 0]$ or $[turn(comp(i, k)) \neq role(i, k)]$
      end;
  $crit_i$;
  ** Critical region **
  $exit_i$:
    for $k = \log n$ downto 1 do
      $flag(comp(i, k)) := 0$;
  $rem_i$;
```

Assertion 3 *In any reachable system state by n-tournamentFME, for any i, $0 \leq i \leq n-1$, and any k, $1 \leq k \leq \log n$, at most one process from the subtree rooted at $comp(i,k)$ is a winner at $comp(i,k)$.*

Theorem 7 *In n-tournamentFME, the time from when a particular process i enters its trying region until it enters its critical region is at most $(n-1)c + O(nl)$.*

Proof. Define $T(0)$ to be the maximum time from when a process enters the trying region until it enters the critical region. For k, $1 \leq k \leq \log n$, define $T(k)$ to be the maximum time from when a process wins at level k of the trying region until it enters the critical region.

It is immediate that $T(\log n) \leq l$. Consider a situation that process i has just entered the competition at level k of the trying region. Let this event be denoted by $\pi(i,k)$. Within at most $4l+T(k)+c+(\log n-k+2)l$ after $\pi(i,k)$, $flag(opposite(i,k)) = 0$ or $turn(comp(i,k)) \neq role(i,k)$ is satisfied. Then within at most further time $2l$, process i becomes a winner at level k of the trying region. Hence, the maximum time from $\pi(i,k)$ until process i reaches the entrance to the critical region is at most $2T(k) + c + (\log n - k + 8)l$. Thus, we need to solve the following recurrence for $T(0)$.

$$T(\log n) \leq l,$$

$$T(k-1) \leq 2T(k) + c + (\log n - k + 8)l.$$

Then we have the following inequality.

$$T(0) \leq (1+2+\cdots+2^{\log n-1})(c+7l)+(2^{\log n}+2^{\log n-1}+2\cdot 2^{\log n-2}+\cdots+(\log n-1)\cdot 2^1+\log n\cdot 2^0)l$$

$$\leq (n-1)c + O(nl).$$

□

The progress in the exit region of *n-tournamentFME* is obvious. Hence, from Assertion 3 and Theorem 7, we have the following theorem.

Theorem 8 *n-tournamentFME solves the mutual exclusion problem and is lockout-free.*

5 Concluding remarks

We have proposed three lockout-free mutual exclusion algorithms. These algorithms are some modifications of the well-known algorithms. Although the modifications are simple, the bound on the running time of each modified algorithm is much improved over its original algorithm. The third algorithm, *n-tournamentFME* uses almost twice as many *flag* variables as the *flag* variables of *n-tournamentME.* The increase of the number of the shared variables is a disadvantage for *n-tournamentFME.* However, the *flag* variables of *n-tournamentFME* assume only two values 0 and 1, whereas the *flag* variables of *n-tournamentME* assume $\log n$ different values. Hence, *n-tournamentFME* is also good at the size of the shared variables. We are interested in a problem whether the number of shared variables of *n-tournamentFME* can be reduced without increasing its running time. This would be a worthy problem of further 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] 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.

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

[5] 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.

[6] 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.

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

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

[9] 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.

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

[11] 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.