Mutual Exclusion Space Lower Bounds

November 1, 2017

The proofs provided herein are taken from the book "Distributed Computing", by Hagit Attiya and Jennifer Welch, with some terminology/notation adaptation.

1 Lower bound on the number of registers for mutual exclusion

Definition 1.1 A configuration is a vector $C = (q_0, \cdots, q_{n-1}, r_0, \cdots, r_{m-1})$, where $q_i$ is a state of $p_i$ and $r_j$ is a value of register $R_j$.

Definition 1.2 A configuration of a mutual exclusion algorithm is quiescent if all processes are in the remainder section.

Definition 1.3 Configurations $C$ and $C'$ are indistinguishable to a set of processes $P$, denoted $C \overset{P}{\sim} C'$, if each process in $P$ has the same state in $C$ as in $C'$ and the values of all registers are the same in both configurations.

Definition 1.4 Configuration $C$ is $P$-quiescent, where $P$ is a set of processes, if there exists a quiescent configuration $D$ such that $C \overset{P}{\sim} D$.

Definition 1.5 A process covers a register in a configuration if it is about to write to it (according to its state in the configuration).
**Definition 1.6** An execution is a sequence of steps. For configuration \( C \) and execution \( \sigma \) we denote by \( C\sigma \) the configuration that is reached when \( \sigma \) is executed starting from \( C \).

**Lemma 1** Let \( C \) be a configuration that is \( p_i \)-quiescent for some process \( p_i \). Then there exists a \( p_i \)-only execution \( \sigma \) such that \( p_i \) is in the critical section in \( C\sigma \) and, during \( \sigma \), \( p_i \) writes to a register that is not covered by any other process in \( C \).

**Proof:** First we show that such an execution \( \sigma \) exists. As \( C \) is \( p_i \)-quiescent, there exists a quiescent configuration \( D \) such that \( C \xrightarrow[p_i]{} D \) holds. From deadlock-freedom, if \( p_i \) runs alone from \( D \) it eventually enters the critical section. Let \( \sigma \) denote this execution. Hence \( p_i \) will execute \( \sigma \) also after \( C \) and enter the critical section. We now show that during \( \sigma \), \( p_i \) must write to a register not covered by any other process in \( C \). Suppose not. Let \( P \) be a set of processes, not including \( p_i \), that cover some register in \( C \) and let \( W \) be the set of registers that are covered in \( C \) by some process in \( P \).

Starting at \( C \), let each process in \( P \) perform a single step. The result is that all the registers that were covered in \( W \) have been overwritten. Then extend the execution by letting every process that is not in its remainder section in \( C \) return to the remainder section. This is possible because of the deadlock-freedom assumption. Let \( \tau \) be the resulting execution and let \( Q = C\tau \) be the resulting configuration. Note that \( Q \) is quiescent.

Pick some process \( p_j \neq p_i \). From deadlock-freedom, there is a \( p_j \)-only execution, \( \pi \), such that \( p_j \) is in the critical section in \( C\tau\pi \).

Finally, observe that during \( \tau\pi \), the other processes cannot tell whether \( p_i \) has performed the steps of \( \sigma \) or not, since \( \tau \) overwrites any value that \( p_i \) may have written. Thus, \( p_j \) is in its critical section also in \( C\sigma\tau\pi \) together with \( p_i \). This violates mutual exclusion.

**Lemma 2** For all \( k \), \( 1 \leq k \leq n \), and for any quiescent configuration \( C \), there exists a configuration \( D \) reachable from \( C \) by a \( \{p_0, \ldots, p_{k-1}\} \)-only schedule such that \( p_0, \ldots, p_{k-1} \) cover \( k \) distinct registers in \( D \) and \( D \) is \( \{p_k, \ldots, p_{n-1}\} \)-quiescent.
Proof: By induction on $k$.

Basis: $k=1$. Fix a quiescent configuration $C$. Observe that before entering the critical section, a process must write to some register. Thus there exists a $p_0$-only execution $\sigma$ starting from $C$ such that $p_0$ performs at least one write in $\sigma$. Let $\sigma'$ be the prefix of $\sigma$ just before $p_0$ performs its first write, say to register $x$, and let $D = C \sigma'$. Clearly, $p_0$ covers $x$ in $D$. Since no writes were done in $\sigma'$ and since it is a $p_0$-only execution, $D$ is $\{p_1, \cdots, p_{n-1}\}$-quiescent.

Induction: Assume the lemma is true for $k \geq 1$ and show it for $k+1$.

For purposes of explanation, assume for now that every application of the inductive hypothesis causes the same set $W$ of $k$ registers to be covered by $p_0$ through $p_{k-1}$.

By the induction hypothesis, we can get to a configuration $C_1$ that appears quiescent to $p_k, \cdots, p_{n-1}$ in which $p_0, \cdots, p_{k-1}$ cover $W$. We must now show how to cover an additional register, for a total of $k+1$ covered registers.

Lemma 1 implies that we can get $p_k$ to cover an additional register, say $x$, by starting at $C_1$ and just having $p_k$ execute steps. Call this execution $\pi'$. However, the resulting configuration does not necessarily appear quiescent to $p_{k+1}, \cdots, p_{n-1}$ since $p_k$ may have written to some (covered) registers.

From $C_1 \pi'$, we can get to a $\{p_{k+1}, \cdots, p_{n-1}\}$-quiescent configuration while still keeping $x$ covered by $p_k$ as follows. First, we overwrite all traces of $p_k$ by having $p_0, \cdots, p_{k-1}$ each execute its write step. Second, we use the deadlock-freedom assumption to cause $p_0, \cdots, p_{k-1}$ to return to the remainder section. Call the resulting execution $\tau$ and let $D_1' = C_1 \pi' \tau$ be the resulting configuration.

Finally, we would like to invoke the induction hypothesis on $D_1'$ to get to another configuration in which $W$ is covered again and which appears quiescent to $p_{k+1}, \cdots, p_{n-1}$. But the inductive hypothesis requires that we start with a (totally) quiescent configuration, and $D_1'$ is not quiescent since $p_k$ is in its entry section. However, this problem can be solved by noting that applying $\tau$ to $C_1$ produces a configuration $D_1$ that is quiescent. Thus by the induction hypothesis, there is a $\{p_0, \cdots, p_{k-1}\}$-only schedule $\sigma$ such that $C_2 = D_1 \sigma$ is $\{p_k, \cdots, p_{n-1}\}$-quiescent and $W$ is
covered by \( p_0, \cdots, p_{k-1} \). Since \( D_1 \) is indistinguishable from \( D'_1 \) to \( p_0, \cdots, p_{k-1} \), \( D'_1 \sigma \) exists. Thus in \( C'_2 = D'_1 \sigma \), \( k + 1 \) registers are covered and \( C'_2 \) appears quiescent to \( p_{k+1}, \cdots, p_{n-1} \).

However, it is not necessarily the case that every application of the induction hypothesis causes the same set of \( k \) registers to be covered. But since there is only a finite number of registers used by the algorithm, there is only a finite number of \( k \)-tuples of registers that can be covered. Thus, we repeatedly apply the induction hypothesis, cycling between quiescent configurations \( (D_1, D_2, \cdots) \) and \( \{p_k, \cdots, p_{n-1}\} \)-quiescent configurations in which \( k \) registers are covered \( (C_1, C_2, \cdots) \). Eventually we will find two configurations \( C_i \) and \( C_j, j > i \), in which the same set of \( k \) registers is covered. We can then use essentially the same argument we did above with \( C_1 \) and \( C_2 \).

We now proceed with the details. Let \( C \) be a quiescent configuration. We now inductively define an infinite execution starting with \( C = D_0 \) and passing through configurations \( C_1, D_1, C_2, D_2, \ldots \).

Given configuration \( C_m, m > 0 \), define \( D_m \) as follows. First, let each of the processes \( p_0, \cdots, p_{k-1} \) perform their write step. This causes every register in \( W_m \) to be written. Now use the deadlock-freedom assumption in order to cause each of \( p_0, \cdots, p_{k-1} \) to get to the remainder section. Call this execution \( \tau_m \). Let \( D_m \) be the resulting configuration. \( D_m \) is clearly quiescent.

Since the set of registers is finite, there exist \( i \) and \( j > i \) such that \( W_i = W'_j \). Recall that \( C_i \) is \( \{p_k, \cdots, p_{n-1}\} \)-quiescent. By Lemma 1, there is a \( p_k \)-only execution \( \pi \) such that \( p_k \) is in the critical section in \( C_i \pi \), and \( p_k \) writes to some register not in \( W_i \) during \( \pi \). Let \( \pi' \) be the prefix of \( \pi \) just before \( p_k \)'s first write to some register, say \( x \), not in \( W_i \). We can extend \( C_i \), by \( \pi' \tau_i \sigma_{i+1} \tau_{i+1} \cdots \sigma_j \), since \( \tau_i \) overwrites all the writes of \( p_k \) to \( W_i \) (if any) and the rest of the execution consists of steps by processes other than \( p_k \). Let \( C'_j \) be the resulting configuration.

We finish by showing that \( C'_j \) is the desired configuration \( D \). Since \( \pi' \) is \( p_k \)-only, the beginning of \( \tau_i \) writes to all registers in \( W_i \), and \( \tau_i \sigma_{i+1} \tau_{i+1} \cdots \sigma_j \) involves only \( p_0 \) through \( p_{k-1} \), it follows that \( C'_j P \sim C_j \), where \( P \) is the set of all processes except \( p_k \). Thus \( C'_j \) is \( \{p_{k+1}, \cdots, p_{n-1}\} \)-quiescent and, in \( C'_j, p_0, \cdots, p_{k-1} \) cover \( W_j \) and \( p_k \) covers \( x \). Since \( x \notin W_i \) and \( W_i = W_j \), \( k + 1 \) distinct registers are covered in \( C'_j \).
Instantiating Lemma 2 with $k = n$ and with an initial configuration $C$ gives the following theorem.

**Theorem 3** Any deadlock-free mutual exclusion algorithm that uses only read/write registers must use at least $n$ registers, regardless of their size.

## 2 Lower bound on the number of bits for mutual exclusion

**Definition 2.1** A mutual exclusion algorithm provides $k$-bounded waiting if, in every execution, no process enters the critical section more than $k$ times while another process is waiting in the entry section.

**Theorem 4** If a mutual exclusion algorithm provides $k$-bounded waiting (for some $k$), then the algorithm uses at least $\lceil \log_2 n \rceil$ bits.

**Proof:** Let $C$ be an initial configuration. Clearly, $C$ is quiescent. From deadlock freedom, there is a finite $p_0$-only execution, $\tau_0$, after which $p_0$ is in the critical section. Let $C_0 = C\tau_0$. Inductively, construct for every $i \in \{1, \cdots, n - 1\}$ a $p_i$-only execution-fragment, $\tau_i$, such that $p_i$ is waiting in the entry section in $C_i = C_{i-1}\tau_i$ ($p_i$ takes a bounded number of steps to go from the remainder, through the doorway, and start waiting in the entry section). Thus, $p_0$ is in the critical section and $p_1, \cdots, p_{n-1}$ are waiting in the entry section at $C_{n-1} = C\tau_0, \cdots, \tau_{n-1}$.

To obtain a contradiction, assume the algorithm uses less than $\lceil \log_2 n \rceil$ shared bits. This implies that there are two configuration, $C_i$ and $C_j$, $0 \leq i < j \leq n - 1$, such that all registers have the same values at $C_i$ and $C_j$. Note that $p_0, \cdots, p_i$ do not perform any steps in $\tau_{i+1}, \cdots, \tau_j$ and so $C_i \{p_0, \cdots, p_i\} \sim C_j$. Furthermore, in $C_i$ (hence also in $C_j$), $p_0$ is in the critical section and $p_1, \cdots, p_i$ are in the entry section.

It follows that, in a long enough execution fragment, $\rho$, in which only $p_0, \cdots, p_i$ perform steps after $C_i$, some of these processes, $p_l$, enters the critical section more than $k$ times. Since
$C_i \{p_0, \ldots, p_k\} 

C_j$ holds, $p_l$ enters the critical section more than $k$ times also if $\rho$ is executed after $C_j$, while $p_j$ is waiting in the entry section. ■