Lower Bound on Number of Read/Write Variables

Definition
Let \( C \) and \( C' \) be two configurations and let \( p_i \) be a processor. \( C \) and \( C' \) are similar to \( p_i \) if (\( C \sim p_i C' \))

(1) \( p_i \) is in same state in both \( C \) and \( C' \)
(2) Values of all shared registers are the same in \( C \) and \( C' \)

Note (1) says that \( p_i \) cannot distinguish between \( C \) and \( C' \).

(2) says something stronger, even if \( p_i \) were to take steps on its own future \( C \circ C' \), it still would not be able to distinguish the two configurations since the value of shared registers is the same.

Definition \( C \circ C' \) if for all \( p_i \in P, C \sim p_i C' \)

Definition
\( u \) is \( p \)-only if it only contains steps by \( p \)
\( u \) is \( P \)-only if it only contains steps by \( p \), except in \( P \)
\( u \) is \( p \)-free if it contains no steps by \( p \)
\( u \) is \( P \)-free if it contains no steps by processors in \( P \).
Lemma 1: Let \( C_1 \) and \( C_2 \) be two configurations, and let \( P \) be a set of processors. If \( C_1 \vartriangleleft C_2 \) and \( \sigma \) is a finite \( P \)-only schedule, then \( \sigma(C_1) \vartriangleleft \sigma(C_2) \).

Proof: By induction on length of \( \sigma \).

Basis: \( |\sigma| = 0 \). Then \( \sigma(C_1) = C_1 \) and \( \sigma(C_2) = C_2 \), so clearly \( \sigma(C_1) \vartriangleleft \sigma(C_2) \).

Inductive step: suppose Lemma is true for \( \sigma \), where \( |\sigma| = k - 1 \). We prove it true for \( \sigma \alpha = \sigma' \), where \( \alpha \) is a single event. By inductive hypothesis, \( \sigma(C_1) \vartriangleleft \sigma(C_2) \). Let \( \alpha \) be an event of some processor \( p_i \in P \). We prove that \( \sigma'(C_1) \vartriangleleft \sigma'(C_2) \).

All processors in \( P \) other than \( p_i \) remain in same state after event \( \alpha \). Since they are in the same state in \( \sigma(C_1) \) and \( \sigma(C_2) \), they are also in the same state in \( \sigma'(C_1) \) and \( \sigma'(C_2) \). Also, the values of registers are the same in \( \sigma(C_1) \) and \( \sigma(C_2) \) since \( p_i \) is in the same state in \( \sigma(C_1) \) and \( \sigma(C_2) \), \( p_i \) will take the same step from both configurations. There are two cases:

1. \( p_i \) reads \( R_j \) in \( \alpha \) where \( p_i \) read the same value in both \( C_1 \) and \( C_2 \) and will have the same state in \( \sigma'(C_1) \) and \( \sigma'(C_2) \). Registers are not modified, so values are same.
(2) \( p_i \) writes \( v \) in \( R_j \) in \( \alpha \).

\( p_i \) writes the same value in both configurations, and will be in the same state in \( \sigma'(C_1) \) and \( \sigma'(C_2) \). \( R_j \) has value \( v \) in both \( \sigma'(C_1) \) and \( \sigma'(C_2) \) and none of the other registers are modified, so they are the same in \( \sigma'(C_1) \) and \( \sigma'(C_2) \).

So in both cases, \( \sigma'(C_1) \cong \sigma'(C_2) \). \( \Box \)

**Definition.** A configuration \( C \) is quiescent if all processes are in remainder section in \( C \).

\[ \begin{align*}
\text{quiescent} & \quad p_j \text{ in } C \\
C & \quad p_j \text{-only} \\
\rightarrow & \quad D
\end{align*} \]

**Lemma 2.** Given a quiescent configuration \( C \), there exists a \( p_j \)-only schedule \( \sigma \) such that \( p_j \) is in critical section in configuration \( \sigma(C) = D \), for any process \( p_j \).

**Proof:** This follows from the No Deadlock property.

We will prove a lower bound on the no. of read/write shared variables required for mutual exclusion. We first prove a simpler result, with the added assumption that each shared variable can be written by only one process.
Lemma 3: Given a quiescent configuration $C$, and a processor $p_i$, in any $p_i$-only schedule $\sigma$, such that $p_i$ is in critical section in $\sigma(C)$, $p_i$ must write into some variable in $C$.

Proof: Suppose, for contradiction, $p_i$ does not write into any variable in $C$. Consider some processor $p_j$, $j \neq i$. Now, $C \leadsto^{p_j} \sigma(C) = D$. By Lemma 2, there exists a $p_j$-only schedule $\sigma'$ such that $p_j$ is in critical section in $\sigma'(C) = E$. Consider the same $p_j$-only schedule $\sigma'$ applied to $D$ (possible since $C \overset{p_j}{\leadsto} D$). Since $C \overset{p_j}{\leadsto} D$, Lemma 1 implies that $\sigma'(C) \overset{p_j}{\leadsto} \sigma'(D)$. Since $p_j$ is in critical section in $\sigma'(C) = E$, it follows that $p_j$ is in $CS$ in $\sigma'(D) = F$. This violates Mutual Exclusion since $p_i$ and $p_j$ are in $CS$ together, a contradiction.

Q.E.D.

\[ C \overset{\sigma}{\longrightarrow} D, p_i in CS \]

\[ p_j \text{-only} \quad \sigma \quad \overset{\sigma'}{\longrightarrow} \]

\[ p_j \text{ in } CS \quad E \]

\[ (E) \quad p_i, p_j \text{ in } CS \]

Theorem: If the shared variables are single-writer, at least $n$ variables are required to provide deadlock-free access to critical section with mutual exclusion for $n$ processors.
Proof:
We prove the theorem by contradiction. Suppose that the number of shared variables is less than $n$. Then, some process, say $p_i$, will never write.

Consider a quiescent configuration $C$. By Lemma 2, there is a $p_i$-only schedule $S$ such that $p_i$ is in critical section in $S(C)$. By Lemma 3, $p_i$ must write in some variable in $S$ (otherwise, we have shown that mutual exclusion is violated). This is a contradiction to the claim above that $p_i$ never writes.
So, at least $n$ shared variables are required.

QED

In the last theorem, we showed a contradiction by not allowing a process to write. If we no longer have this restriction of single-write variables, this is no longer a valid argument. We now argue that even if a process writes, its writes go undetected by other processes, if there are fewer than $n$ variables.
At A be a deadlock free mutual exclusion algorithm. We show an execution in which all n processes are about to write into distinct shared variables.

**Definition.** A process covers a variable in a configuration if in its next step it will write to the variable (according to its state in the configuration).

We will use induction on the number of shared variables that are covered to show the existence of the desired execution (with n covered variables). We will need the configuration to appear quiescent to a certain set of processes, so they can take the required steps.

**Definition.** Configuration C is P-quiescent where P is a set of processes, if there exists a reachable quiescent configuration D such that C < D.

Before a process can enter a critical section it must write into some variable that is not covered. This is to ensure that it can notify other processes and this information it writes is not overwritten before it is read by another process.
We repeat Lemma 2 and Lemma 3.

**Lemma 2** Given a quiescent configuration $C$, there exists a $p_i$-only schedule $S$ such that $p_i$ is in $CS$ in configuration $S(C)$ for any process $p_i$.

**Lemma 3** Given a quiescent configuration $C$, and a process $p_i$, in any $p_i$-only schedule $S$ such that $p_i$ is in $CS$ in $S(C)$, $p_i$ must write into some variable in $R$.

We now prove Lemma 4.

**Lemma 4** Let $C$ be a reachable configuration that is $p_i$-quiescent for some process $p_i$. Then there exists a $p_i$-only schedule $S$ such that $p_i$ is in $CS$ in $S(C)$, and during execution $exec(S,C)$, $p_i$ writes to some variable that is not caused by any other process in $C$.

**Proof:** We first show that $S$ exists. Since $C$ is $p_i$-quiescent, there is a quiescent configuration $D$ such that $C^p_i D$. By Lemma 2, there exists a $p_i$-only schedule $S$ such that $p_i$ is in $CS$ in $S(D)$. Since, by Lemma 1, $S(C) \leq S(D)$, $p_i$ is in $CS$ in $S(C)$. Additionally, by Lemma 3, $p_i$ must write into some variable in $R$. 
We now show that $p_i$ writes into some variable not covered by any process in $C$ during exec $(C, p)$. Suppose not.

Let $W$ be the set of variables covered by at least one process $p_j$, $j \neq i$, in $C$. Let $P'$ be a set of processes $(p_i \notin P')$, so that each variable in $W$ is covered by exactly one process in $P'$. By our assumption, $p_i$ writes to $RW$ in $p$. Let $p(C) = c'$.

Let $p$ be a schedule with one step by each proc $p \in P'$ since $p$ commutes all updates by $p_i$ in $q$, registers have same value in $M$ and $M'$. 

\[ \quad \]
where $M = \rho(C)$ and $M' = \rho(C')$. Except for $p_i$, all other proc have some state in $M$ and $M'$, so $M \cong M'$ for all $i \neq i$.

Let $\tau$ be a schedule where only processes in entry, CS or Exit in $M$ take steps until some process $p_j$ is in CS. This exist by No Deadlock since $p_i \in \text{REM}$ in $M$, $\gamma$ is $p_i$-free.

Let $\gamma(M) = Q$ and $\gamma(M') = Q'$. Now, $Q \cong Q', j \neq i$ since $p_j$ is in CS in $Q$, $p_j$ is also in CS in $Q'$. But, $p_i$ is also in CS in $Q'$, violating mutual exclusion.

QED.

Lemma 5. For all $k$, $1 \leq k \leq n$, and for all reachable quiescent configurations $C$, there exist config 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 variables in $D$, and $D \in \{p_k, \ldots, p_{n-1}\}$-quiescent.

Proof (by induction on $k$)

Base ($k = 1$)

Fix a quiescent configuration $C$. We show there is a $p_0$-only schedule $\sigma$ such that $p_0$ covers a variable in $\sigma'(C)$. By Lemma 3, a $p_0$-only schedule $\sigma$ exists s.t. $p_0$ is in CS in $\sigma(C)$, and $p_0$ writes to some variable.
in \( s \), let \( s' \) be the longest prefix of \( s \) that does not contain a write and let \( D = s'(c) \).

Since \( s = s' s'' \) and the first event in \( s'' \) is a write, \( p_i \) causes a variable in \( D \).

Also, since there are no writes in the \( p_0 \)-only schedule \( s' \), \( D \) is \( p_0 \)-quiescent.

**Note:** \( P_i = \{ p_0, \ldots, p_i \} \), \( P_i = \{ p_{i+1}, \ldots, p_{n-1} \} \)

\[ C \quad \text{p}_0 \text{-only} \quad s' \quad \text{p}_0 \text{ writes to} \quad X \]

\[ \text{quiescent} \quad \text{read} \quad \overline{\text{p}_0 \text{-quiescent}} \]

**Induction Step**

Assume lemma true for \( k \geq 1 \), and show it true for \( k+1 \) (simple case).
Problem: some set $W$ may not be covered each time

C = $E_0$ quiescent

$E_0$ → $E_1$ quiescent → $E_{i-1}$ quiescent

$C_1$ $P_{k-1}$ covers $W$

$C_2$ $P_{k-1}$ covers $W_2$

$C_l$ $P_{k-1}$ covers $W_l$

$D_1$ quiescent

$D_2$ quiescent

$D_i$ quiescent

$E_l$ quiescent

$P_{k-1}$ covers $x \notin W$

by Lemma 4

not nec

$P_k = P_0, P_1, \ldots P_{k-1}$ (one step each)

$P_k$ covers $x \notin W$

$P_k$ in $E_1$

$P_k$ in $E_0$

rest in $REM$

$P_{k-1}$ in $REM$

$P_{k-1}$ covers $W$

$P_{k-1}$ covers $W_u$ for $u = 1, 2, \ldots$

$P_{k-1}$ covers $W_i$

$P_{k-1}$ covers $W_i$

$P_{k-1}$ covers $x \notin W$

$P_k$ covers $x \notin W$

$P_k$ covers $x \notin W$
We consider the infinite execution

\[ E_0 \rightarrow C_1 \rightarrow D_1 \rightarrow E_1 \rightarrow C_2 \rightarrow D_2 \rightarrow E_2 \rightarrow \ldots \]

where \( P_{k-1} \) covers \( W_i \) in \( C_i \). By PHP, there exists \( i \neq j \) such that \( W_i = W_j \), i.e., some set of variables is covered in \( C_i \) and \( C_j \). We can now use a similar argument to the one earlier.

\[ P_{k-1} \] covers \( C_i \), \( P_k \) only writes to \( R_i \in W_i \)
\[ \sim \]
\[ P_k \] writes to \( R_i \in W_i \)

Then, any no deadlock MG alg needs \( n \) shared variables.

Proof: Apply Lemma 5 with \( k = n \).