SPTDC 2019: Lower Bounds in Distributed Computing Solutions

1 Covering and valence in consensus algorithms

Consider an obstruction-free binary consensus algorithm using atomic read-write registers.

Let P be bivalent from $C\beta$, where β is a block write by some $R \subseteq P$. Let γ be a schedule of some $z \notin P$ such that z decides in $C\gamma$. Show that z must write to a register not covered by R in C.

Suppose not, i.e., for some $\gamma \in z^*$, z decides a value $v \in \{0, 1\}$ in $C\gamma$ writing only to registers covered by R in C. Thus, in no process in P can distinguish $C\gamma\beta\alpha$ from $C\beta\alpha$ where α is a P-only schedule. As $C\beta$ is bivalent, we can choose $C\beta\alpha$ and, thus, $C\gamma\beta\alpha$, to decide $1 - v$—a contradiction.

2 Space complexity of mutual exclusion

1. *Show that any* 2*-process read-write mutual-exclusion alogirithm requires at least* 2 *registers.*

   Let p_1 run its TS (trying section) until it is about to perform its first write: such a schedule must exist, as otherwise p_2 may enter its CS without noticing p_1. Let α_1 be the corresponding schedule. Since α_1 is indistinhguishable to p_2 from an empty schedule, there exists a schedule $\alpha \in p_2^*$ such that p_2 is in its CS at the end of $\alpha_1\alpha$.

   If p_2 only writes to the register covered by p_1 in $\alpha_1\alpha$, then we can wake up p_1 and let it overwrite all the traces of p_2 and enter its CS—a contradiction.

   Thus, p_2 must write to a distinct register in $\alpha_1\alpha$.

2. *What about* 3 *processes? Can you show that* 3 *registers are necessary?*

3. *Finally, prove the general statement: n-process algorithm requires n registers.*

   We prove the general case, without wasting time on the 3-process scenario.

   By induction, we are going to prove the following claim:

   > Let C be any configuration in which every process is in its remainder section (RS). For all $k = 1, \dots, n$, there exists a schedule α by $P_k = \{p_1, \dots, p_k\}$, such that:
   >
   > - every process in P_k is about to write to a distinct register in $C\alpha$, and
   > - there exists a schedule α' by P_k such that (1) every process is in the remainder section in $C\alpha'$ and (2) $C\alpha$ and $C\alpha'$ differ only in the local states of processes in P_k.

The base case $k = 1$ is immediate: simply run p_1 from C until it is about to perform its first write. No other process can distinguish the resulting configuration from C.

Now suppose that the claim holds for some $k = 1, \dots, n-1$. Let C_0 be the configuration after α. Let $D_0 = C_0\beta_0\gamma_0$ be the extension of C_0 by P_k, where β_0 is the block write by P_k (on a set k distinct registers $\mathcal{B}_0$), such that every process in P_k is in its RS in D_0. Since the algorithm deadlock-free, such an extension exists.

Now we can reuse the induction hypothesis and get a configuration $C_1 = D_0\alpha_0$ in which P_k cover a (possibly different from $\mathcal{B}_0$) set of k registers $\mathcal{B}_1$, etc.

So we get an infinite chain of configurations:

$$C \overset{\alpha}{\rightsquigarrow} C_0 \overset{\beta_0\gamma_0}{\rightsquigarrow} D_0 \overset{\alpha_0}{\rightsquigarrow} C_1 \overset{\beta_1\gamma_1}{\rightsquigarrow} D_1 \overset{\alpha_1}{\rightsquigarrow} C_2 \overset{\beta_2\gamma_2}{\rightsquigarrow} \dots\dots$$

where each C_i satisfies the claim for P_k.

Since there are only finitely many registers there must exist C_i and C_j ($i < j$), such that the same set $\mathcal{B}$ of k registers is covered by P_k in C_i and C_j.

Now we extend C_i with steps of p_{k+1} until it is about to write to a register not in $\mathcal{B}$. Such an extension $C_i' = C_i\psi$ exists, as otherwise p_{k+1} can enters its CS and then all the traces of its presence in the CS will be overwritten by the subsequent block write β_i.

Notice that, since all steps of p_{k+1} are overwritten by the block write in $C_i\psi\beta_i$, the resulting configuration $C_j' = C_j'\psi\beta_i\gamma_i\alpha_i \dots \beta_{j-1}\gamma_{j-1}\alpha_{j-1}$ is indistintinguishable from C_j for any process except p_{k+1}. Thus, C_j' satisfies the claim for $P_{k+1} = \{p_1, \dots, p_{k+1}\}$: (1) every process in P_{k+1} covers a distinct register in C_j' and (2) only processes in P_k can distinguish C_j' from some configuration in which every process is in its RS.

3 RMR complexity of Peterson's algorithm

What is the RMR total work complexity of Peterson's n-process algorithms in the CC model? In the DSM model, where, e.g., every process i keeps level[i] locally?

In the CC model, the complexity is $\Theta(n^3)$.

To see this, consider an execution in which, for every $i = 0, \dots, n-2$, process i is the last to reach waitng rooms $i, \dots, n-2$. Therefore, to leave room i, process i needs to make sure that every process $j = i+1, \dots, n-1$ leaves its critical section and sets `level`$[j]$ to -1. There exists an execution in which i can first see processes $i+1, \dots, n-1$ reaching room $i+1$ (setting their `level` variables to $i+1$), after that—processes $i+2, \dots, n-1$ reaching room $i+2$, after that—processes $i+3, \dots, n-1$ reaching room $i+3$, etc.

Thus, before leaving room i, process i may have to go through $n-2-i$ phases, where each phase $j = i+1, \dots, n-2$ incurs $n-1-j$ cache misses, resulting in $\Omega((n-i)^2)$ RMRs that process i must perform in this execution. The total number of RMRs is therefore $\Omega(\sum_{i=0,\dots,n-2}(n-i)^2) = \Omega(n^3)$.

On the other hand, as every process performs $O(n)$ writes, there only can be $O(n^2)$ cache misses per process, which gives $O(n^3)$ RMRs in the total work.

In the DSM model, it is not hard to see that the complexity is unbounded, regardless of the assignment of variables. In particular, if we decide that each `level`$[i]$ variable is local to process i, the scheduler may force i to perform an unbounded number of reads of `level`$[j]$, $j \neq i$ before reaching the critical section.

Typos and mistakes are possible in this draft. If you find any, please let me know: `petr.kuznetsov@telecom-paris.fr`.