Consensus number proofs for registers and FIFO queues

December 7, 2017

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

1 Consensus from read/write registers

In this section we prove that wait-free consensus for 2 processes or more cannot be implemented from (multi-reader/multi-writer) registers. We actually prove that even the more restricted binary consensus object cannot be implemented in a wait-free manner from registers. In all the discussion that follows we assume that 2 or more processes share the consensus object.

Definition 1.1 The valency of a configuration $C$ is the set of all values that are decided in some execution that starts from $C$. We say that $C$ is bivalent if its valency is $\{0, 1\}$. We say that it is 0-valent (respectively 1-valent) if it contains only 0 (respectively 1).

Let us recall the following definition which we saw on lecture 2.

Definition 1.2 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.
Lemma 1 Let $C_1$ and $C_2$ be two univalent configurations. If $C_1 \overset{p_i}{\sim} C_2$ holds for some process $p_i$, then $C_1$ is $v$-valent if and only if $C_2$ is $v$-valent, for $v = 0, 1$.

Proof: Suppose $C_1$ is $v$-valent. Then, from wait-freedom, if $p_i$ runs by itself starting from $C_1$ it will decide. As $C_1$ is $v$-valent, it must decide $v$. The same argument can be applied to $C_2$. As $C_1 \overset{p_i}{\sim} C_2$ holds, it follows that $C_2$ is also $v$-valent. 

Lemma 2 There exists a bivalent initial configuration.

Proof: Let $I_0$ be the initial configuration in which all processes start with 0, and let $I_1$ be the initial configuration in which all processes start with 1. From the validity requirements, $I_0$ is 0-valent and $I_1$ is 1-valent. Let $I_{01}$ be the initial configuration in which $p_0$ starts with 0 and the remaining processes start with 1. Since $I_{01} \overset{p_0}{\sim} I_0$, we get from Lemma 1 that $I_{01}$ cannot be 1-valent. Since $I_{01} \overset{p_0}{\sim} I_1$, $I_{01}$ cannot be 0-valent. Thus it must be bivalent.

Definition 1.3 We say that process $p$ is critical in configuration $C$ if $C$ is bivalent but $Ce$ is univalent, where $s$ is the step enabled by $p$ at $C$.

Lemma 3 If $C$ is a bivalent configuration then there is at least one process that is not critical in $C$.

Proof: To obtain a contradiction, assume that all processes are critical in $C$. It follows that there are two processes $p_i$ and $p_j$ with enabled steps $s_i$ and $s_j$, respectively, such that $Cs_i$ is 0-valent and $Cs_j$ is 1-valent. We consider the following possibilities.

- Steps $s_i$ and $s_j$ access distinct registers or read the same register. In this case we have $Cs_i s_j \not\overset{p}{\sim} Cs_j s_i$. Thus, from Lemma 1, $Cs_i s_j$ and $Cs_j s_i$ cannot have different valencies. This contradicts our assumption that $Cs_i$ is 0-valent and $Cs_j$ is 1-valent.

- Step $s_i$ writes to some register $r$ and step $s_j$ reads $r$. Clearly $Cs_i \overset{p_i}{\sim} Cs_j s_i$ holds. However, $Cs_i$ is 0-valent whereas $Cs_j s_i$ is 1-valent. This contradicts Lemma 1.
• Steps $s_i, s_j$ write to the same register. Clearly $C s_j \sim_{p_i} C s_i s_j$ holds. However, $C s_j$ is 1-valent whereas $C s_i$ is 0-valent. Again, this is a contradiction to Lemma 1.

The result follows.

Theorem 4  There is no wait-free implementation of the consensus object, shared by two or more processes, from registers.

Proof: We inductively construct an execution of infinite length such that all the configurations reached by it are bivalent; thus no process can decide in this execution.

The base case is established by Lemma 2. Assume $C$ is a bivalent configuration reached after an execution of length $k$. From Lemma 3, there is a non-critical process in $C$. We extend the execution by letting this process apply its step. We thus reach a bivalent configuration after an execution of length $k + 1$.

2 Consensus from FIFO queue

Theorem 5  There is no wait-free implementation of consensus, shared by three or more processes, from FIFO queues and read/write registers.

Proof: Assume, in contradiction, that there is such an algorithm for 3 processes, $p_0, p_1$ and $p_2$. From Lemma 2, there is an initial bivalent configuration $B$. We let the three processes take steps until we reach a critical configuration $C$, such that $C$ is bivalent, but all of $C s_0, C s_1$ and $C s_2$ are univalent, where $s_i$ is the step of process $p_i$ enabled at $C$, for $i = 0, 1, 2$. From definition 1.1, it cannot be that all of $C s_0, C s_1$ and $C s_2$ have the same valency. WLOG, assume that $C s_0$ is 0-valent and $C s_1$ is 1-valent.

We now consider the operations applied by $s_0$ and $e p_1$. If both steps access different objects, or both access read/write registers then, by using arguments similar to those used in Theorem 4, we
obtain a contradiction. We therefore need only consider the case where both these steps access the same fifo queue $Q$.

The following possibilities exist.

1. Both $s_0$ and $s_1$ are dequeue operations on $Q$. In this case, $Cs_0 \sim p_0 Cs_1$ holds. From Lemma 1, this implies in turn that $Cs_0$ and $Cs_1$ cannot have the same uni-valency. This is a contradiction to our assumption that $Cs_0$ is 0-valent and $Cs_1$ is 1-valent.

2. $s_0$ is an enqueue operation on $Q$ and $s_1$ is a dequeue operation on $Q$ (or vice versa). If $Q$ is non-empty, then $Cs_0s_1 \sim p_0 Cs_1s_0$ holds. This yields a contradiction by the argument used in the previous case. Assume, then, that $Q$ is empty. In this case, $Cs_0 \sim p_0 Cs_1e_0$ holds, a contradiction.

3. Both $s_0$ and $s_1$ are enqueue operations on $Q$. Let $a$ and $b$ be the values enqueued by $s_0$ and $s_1$, respectively. Let $k-1$ be the number of items in $Q$ in $C$. Thus, in $Cs_0$, $a$ is the $k$’th item in $Q$. Similarly, $b$ is the $k$’th item in $Q$ in $Cs_1$.

Since $Cs_0$ is 0-valent and from wait-freedom, starting from $Cs_0s_1$ there exists an execution $\sigma$ by $p_0$ in which it decides 0. We claim that in $\sigma$, $p_0$ must dequeue the $k$’th item of $Q$. Assume otherwise to obtain a contradiction. Then, in $\sigma$, $p_0$ does fewer than $k$ dequeue operations and does not dequeue $a$. Hence $p_0$ decides 0 also in $Cs_1s_0\sigma$, a contradiction. It follows that, in $\sigma$, $p_0$ does at least $k$ dequeue operations on $Q$. Let $\sigma'$ be the longest prefix of $\sigma$ that does not include $p_0$’s $k$th dequeue operation.

Starting from $Cs_0s_1\sigma'$, there exists an execution by $p_1$, $\tau$, in which $p_1$ decides 0. We claim that $p_1$ must perform at least one dequeue operation on $Q$ in $\tau$. Assume otherwise, then $p_1$ decides 0 also in $Cs_1s_0\sigma'\tau$, which contradicts our assumption that $Cs_1$ is 1-valent. Let $\tau'$ be the longest prefix of $\tau$ that does not include $p_1$’s dequeue on $Q$.

Consider two extensions from $C$. First, consider the execution in which, after $Cs_0s_1\sigma'$, $p_0$ dequeues $a$ from $Q$, then $p_1$ executes $\tau'$, then $p_1$ dequeues $b$ from $Q$. Let $D_0$ be the resulting
configuration.

Second, consider the execution in which, after $Cs_1s_0\sigma'$, $p_0$ dequeues $b$ from $Q$, then $p_1$ executes $\tau'$, then $p_1$ dequeues $a$ from $Q$. Let $D_1$ be the resulting configuration. Clearly, $D_0 \leq D_1$, thus, from Lemma 1, $D_1$ and $D_2$ must have the same uni-valency. This is a contradiction.