1 Introduction

In this lecture the concept of \textit{valence} is introduced. We then build upon the notion of valence and examine how to prove that objects are on a specific level of the consensus hierarchy. Finally, we use this process to show that read/write objects are on level 1 of the consensus hierarchy, while additionally proving 3 important supporting Lemmas.

2 Read/write objects in the Consensus Hierarchy

2.1 Definitions

Definition 1 The \textit{valence} of a configuration $C$ is the set of all values that are decided upon by some processor in some configuration reachable from $C$.

Definition 2 A configuration $C$ is \underline{0-valent} if its valence is $\{0\}$.

Definition 3 A configuration $C$ is \underline{1-valent} if its valence is $\{1\}$.

Definition 4 A configuration $C$ is \underline{univalent} if it is either 0-valent or 1-valent.

Definition 5 A configuration $C$ is \underline{bivalent} if some reachable configurations are 0-valent and some are 1-valent.

Definition 6 If $C$ is a bivalent configuration but the configuration resulting from some processor $p_i$ taking a step from $C$ is univalent, then $p_i$ is \underline{critical} in $C$.

2.2 Placing Objects in the Consensus Hierarchy

In order to show some object $X$ is at level $k$ of the consensus hierarchy, we need to show two properties:

1. A $k$-processor wait-free consensus algorithm can be implemented using objects of type $X$.
2. No $k+1$-processor wait-free consensus algorithm using objects of type $X$ exists.
2.3 Placing Read/write objects in the Consensus Hierarchy

Theorem 1 Read/write objects are at level 1 of the consensus hierarchy.

Showing Property 1 is trivial for Theorem 1, but to show property 2 we need a few Lemmas.

Lemma 2 Let $C_1$ and $C_2$ be univalent configurations. If $C_1 \triangleright C_2$, 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 for some $v \in \{0, 1\}$. Now consider an infinite $p_1$-only execution $\sigma$ from $C_1$. Since the algorithm is wait-free, $p_i$ will eventually decide on a value. Since $C_1$ is $v$-valent, $p_i$ decides on value $v$. However, since $C_1 \triangleright C_2$, $p_i$ can take the same infinite execution $\sigma$ from $C_2$, so $p_i$ will decide $v$ here as well. Therefore $C_2$ is also $v$-valent.

Lemma 3 There exists an initial bivalent configuration.

Proof: Let $C_0$ be the configuration $<0,0>$, $C_1$ be the configuration $<1,1>$, and $C_{01}$ be the configuration $<0,1>$. Because of the validity property, $C_0$ is 0-valent while $C_1$ is 1-valent. We prove $C_{01}$ is bivalent.

Suppose for contradiction that $C_{01}$ is univalent, and without loss of generality assume it is 0-valent. Clearly, $C_{01} \triangleright C_1$ since $p_2$ has the same initial value in both $C_{01}$ and $C_1$. Thus, by Lemma 1, $C_{01}$ and $C_1$ have the same valency. Since $C_1$ is 1-valent, then $C_{01}$ must also be 1-valent, a contradiction.

XXX add diagram here

Lemma 4 If $C$ is a bivalent configuration, then at least 1 processor is not critical in $C$.

Proof: Suppose for contradiction that we have two processors, $p_1$ and $p_2$, that are both critical in a bivalent configuration $C$. Then both $p_1(C)$ and $p_2(C)$ are univalent. $p_1(C)$ and $p_2(C)$ must have different valencies since $C$ is bivalent, so assume without loss of generality that $p_1(C)$ is 0-valent and $p_2(C)$ is 1-valent. We also assume without loss of generality that the registers are single writer (since we can construct multi writers from single writers). We must now examine two cases:

Case 1: Both processors either access different registers or read from the same register.

The key in case 1 is that $p_1(C)$ and $p_2(C)$ have no interaction, so $p_1(C) \triangleright p_1(p_2(C))$ and $p_2(C) \triangleright p_2(p_1(C))$. Thus $p_1(p_2(C)) = p_2(p_1(C))$ and no matter the order in which processors take steps, we arrive at the same configuration $C'$. Since $p_1(C)$ is 0-valent and $p_2(C)$ is 1-valent and since $p_1$ and $p_2$ are both critical in $C$, this is a contradiction by Lemma 2.

XXX add diagram here

Case 2: One processor reads and one processor writes, both to the same register.

In this case, assume without loss of generality that $p_1$ writes and $p_2$ reads. Since $p_2$ does not modify any values, $C \triangleright p_2(C)$, and thus $p_1(C) \triangleright p_1(p_2(C))$. Since $p_1(C)$ is 0-valent and $p_2(C)$ is 1-valent, and since $p_1$ and $p_2$ are both critical in $C$, this is a contradiction by Lemma 2.

XXX add diagram here
2.3.1 Proof of Theorem 1

Using the preceding Lemmas we can now prove Theorem 1.

Proof: Showing Property 1 is trivial: since there is exactly one processor \( p \), \( p \) simply decides what its input is.

To show Property 2 we construct an infinite bivalent execution. Let \( C_0 \) be the initial bivalent configuration, proven to exist by Lemma 3. We inductively define the sequence \( C_0, i_0, C_1, i_1, C_2, i_2, \ldots \)

Suppose we have constructed an execution up to \( C_k \). By Lemma 4, either \( p_1 \) or \( p_2 \) is not critical in \( C_k \). Let \( i_k \) be the non-critical processor, and it is this processor we allow to take a step. Thus \( C_{k+1} \) is bivalent.

By continually allowing the non-critical processor to take a step, the execution remains bivalent and thus no consensus is achieved. \( \Box \)