Correction to Proof of Theorem 10.3

Distributed Computing: Fundamentals, Simulations and Advanced Topics

Hagit Attiya and Jennifer Welch

Replaces material on pages 218–220, including Lemmas 10.1 and 10.2. Thanks to Soma Chaudhuri.

1 The Problem

The current proof assigns a linearization point to each logical operation and then shows that the points form a linearization. The point for a logical Write is the earliest time (during the Write) when the binary registers have the correct configuration (i.e., none of the lower registers are still 1). The point for a logical Read is the earliest time (during the Read) when the local variable holding the return value has the final return value.

Here is a counter-example. Time increases downward and ** marks linearization points, according to the definition above. Initially $B[0] = 0, B[1] = 1, B[2] = 0$, so the logical value is 1.

Reader	Writer
Read	
read 0 from B[0]	
	Write(0)
	write 1 to B[0] **
	Ack
read 1 from B[1] **	
	Write(2)
	write 1 to B[2]
	write 0 to B[1]
	write 0 to B[0] **
	Ack
read 0 from B[0]	
Return(1)	

According to the linearization points, the order of the logical operations should be Write(0), Read(1), Write(2), which violates the semantics of a read-write register.

The specific error in the proof is on p. 220 in the proof of Lemma 10.2, Case 1. It says "Since the actual value of R is k between C and C_r, it follows that r reads 1 from $B[k]$ and zeroes below it." This statement is not true (e.g., see counterexample). For the example above, R is the logical register, r is the logical Read under consideration, $k = 0$ is the value written by the latest preceding (w.r.t. the linearization points) Write, C is when the Read completes its upward scan, and C_r is the linearization point of the Read, which is equal to C in this example.

2 The Solution

Construct a permutation π of the logical operations in α as follows. First, place all the Writes in the order in which they occur in α; since there is only one writer, this order is unambiguous.

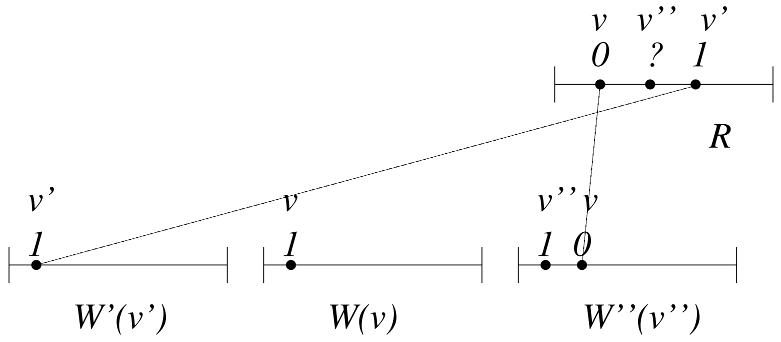

Figure 1: Scenario for Case 2.2

Define Read R in α to **read from** the Write W that contains the latest write to $B[v]$ preceding R's last read of $B[v]$, where v is the return value of R, and "preceding" means with respect to the linearization of the operations on $B[v]$.

Consider the Reads in the order in which they occur in α; since there is only one reader, this order is unambiguous. For each Read R, let W be the Write that R reads from. Place R in π immediately before the Write in π just following W.

We must show that π is a linearization of α. First note that π satisfies the read-write semantics by the way it was constructed. Now we show that the real-time ordering of operations in α is preserved in π. For Writes, this is preserved by construction.

Case 1: R-W: Suppose Read R finishes in α before Write W begins. Clearly R cannot read from a Write that starts after R ends, so R precedes W in π.

Case 2: W-R: Suppose in contradiction Write W finishes in α before Read R begins but in π, R is placed before W. Then R reads from some Write W' that precedes W. Let W write v and W' write v'. Thus the value of R is v'.

Case 2.1: $v' \leq v$. Then W overwrites the write to $B[v']$ by W' before R begins, so R cannot read from W'.

Case 2.2: $v' > v$. (See Figure 1.) W writes a 1 in $B[v]$. Since R does not see this 1 and stop at $B[v]$ during its upward scan, there exists a Write after W that writes 0 to $B[v]$ before R's first read of $B[v]$. Let W'' be the Write with the latest such write to $B[v]$, and let v'' be the value of W''. Then $v'' > v$. Since R reads from W', W'' does not write to $B[v']$, and thus $v'' < v'$. But since W'' is the latest Write preceding R's first read of $B[v]$ and v'' is strictly between v and v', R will read 1 in $B[v'']$ during its upward scan, contradicting the fact that it returns v', which is greater than v''.

Case 3: R-R: Suppose in contradiction Read R_1 precedes Read R_2 in α but follows R_2 in π. This inversion happens because R_1 reads from a Write $W_1(v_1)$ that follows the Write $W_2(v_2)$ that R_2 reads from.

Case 3.1: $v_1 = v_2$. Then when W_1 writes 1 to $B[v_1]$, it is overwriting the 1 that W_2 wrote to $B[v_2]$ earlier. This contradicts the assumption that R_2 reads from W_2.

Case 3.2: $v_1 > v_2$. (See Figure 2.) Since R_2 reads from W_2, no operation to $B[v_2]$ is linearized in between W_2's write of 1 to $B[v_2]$ and R_2's last read of $B[v_2]$. Since R_1 reads from W_1, W_1's write of 1 to $B[v_1]$ precedes R_1's last read of $B[v_1]$. So $B[v_2]$ has the value 1 starting before R_1 does its last read of $B[v_1]$ and ending after R_2 does its last read of $B[v_2]$. But then R_1's last read of $B[v_2]$ would return 1, not 0, a contradiction since R_1 returns v_1, not v_2.

Case 3.3: $v_1 < v_2$. (See Figure 3.) Since R_1 reads from W_1, W_1's write of 1 to $B[v_1]$ precedes R_1's last read of $B[v_1]$. Since R_2 returns $v_2 > v_1$, R_2's first read of $B[v_1]$ must return 0. So there must be another

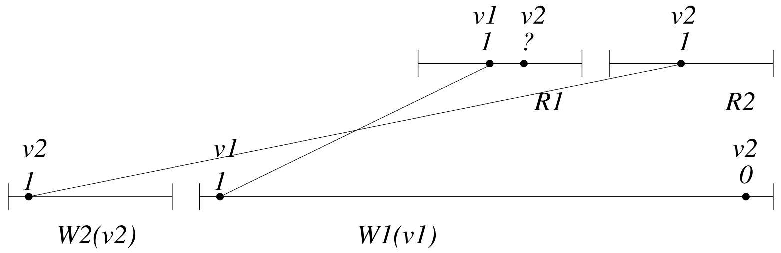

Figure 2: Scenario for Case 3.2

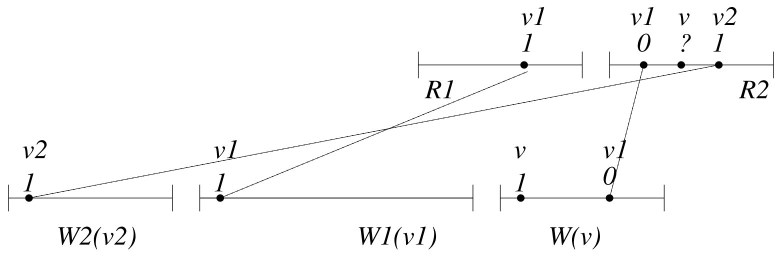

Figure 3: Scenario for Case 3.3

Write after W_1 that writes 0 in $B[v_1]$ before R_2's first read of $B[v_1]$. Let W be the latest such Write and suppose v is its value. Thus $v > v_1$. But since R_2 reads from W_2, W cannot overwrite $B[v_2]$, and so v must be less than v_2. But then R_2 would read a 1 from $B[v]$, not a 0 (W's write of 1 to $B[v]$ cannot be overwritten because it is the last Write preceding R_2's first read of $B[v_1]$).