Assignment 5

An L-buffer object supports two operations, L-BUFFER-READ, which takes no input, and L-BUFFER-WRITE, which takes a bit as input. An L-BUFFER-READ operation returns the last L inputs to L-BUFFER-WRITE operations that were previously performed on the object, in order from least recent to most recent. If fewer than L have been performed on the object, L-BUFFER-READ returns the sequence of all inputs to L-BUFFER-WRITE operations that have been performed, in order from least recent to most recent.

Determine the consensus number of the L-buffer object, for all $L \geq 1$.

Claim 1. *L-buffer objects have consensus number L.*

Proof. L-buffer objects can solve L processor wait-free consensus by Lemma 2. There is no wait-free consensus algorithm for $L+1$ processors using only L-buffer objects and read/write registers by Lemma 4. Thus L-buffer objects have consensus number L as claimed. □

Lemma 2. *L-buffer objects can solve L-processor wait-free consensus.*

Proof. Lemma 3, shows that it is possible to solve L-processor wait-free *binary* consensus using only L-buffer objects and read/write registers. Using a technique discussed in-class, we combine several binary consensus gadgets to solve L-processor wait-free consensus for arbitrary inputs.

Let the L processors be p_1, p_2, ..., p_L and let processor p_i get input u_i. Further, let m be the maximum length of any u_i in binary representation.

The processors want to reach consensus for an m-bit binary string. First, each processor p_i writes down u_i in a set of m binary registers $r_1^{(i)}, ..., r_m^{(i)}$ (padding with zero as necessary). Once p_i is finished writing, it sets a check register c_i to be 1. Then p_i tries to reach consensus for each bit of the output by interacting with m L-buffer objects $l_1, ..., l_m$ in sequential order. It does so using the L-BINARY-CONSENSUS$(l_j, r_j^{(i)})$ operation which runs the wait-free binary consensus algorithm on L-buffer l_j with input $r_j^{(i)}$. If p_i ever "losses" some l_j i.e. the consensus for the j^{th} L-buffer object is different from the j^{th} bit of u_i, then p_i looks at every u_k for $k \neq i$ and adopts some u_k such that the first j bits of u_k match the consensus for $l_1, ..., l_j$ and whose check bit c_k is set to 1 (such a u_k must exist since processors must first write down their input before they interact with the L-buffer objects). See Algorithm 1 for the pseudo-code.

Observe that the termination condition holds trivially. Since the partial sequence $r_1^{(i)}, ..., r_j^{(i)}$ is the prefix of some input for every $1 \leq j \leq m$, the validity condition holds. Since the wait-free binary consensus algorithm works when upto $L-1$-processors fail, all processors must agree on all m bits so the agreement condition holds as well.

□

Lemma 3. *L-buffer objects can solve L-processor wait-free binary consensus.*

Proof. We present an algorithm which solves wait-free *binary* consensus for L processors using only L-buffer objects and read/write registers. Let the set of processors be $p_1, ..., p_L$ and let u_i be the input to p_i. Further let r_i be a single-writer register associated with processor p_i and B be a global L-buffer object.

Algorithm 1 L-processor consensus using only L-buffer objects and read/write registers: code for processor p_i.

$r_1^{(i)}, ..., r_m^{(i)} \leftarrow u_i$
$c_i \leftarrow 1$
for j from 1 to m **do**
  $v \leftarrow$ L-BINARY-CONSENSUS$(l_j, r_j^{(i)})$
  if $v \neq r_j^{(i)}$ **then**
    for k from 1 to L with $k \neq i$ **do**
      if $r_1^{(k)} = l_1$, ..., $r_j^{(k)} = l_j$ and $c_k = 1$ **then**
        $r_j^{(i)} \leftarrow r_j^{(k)}$, ..., $r_m^{(i)} \leftarrow r_m^{(k)}$
      end if
    end for
  end if
end for
$u \leftarrow r_1^{(i)}, ..., r_m^{(i)}$
return u

The high-level description of the algorithm is as follows. Each processor p_i will write $u_i \in \{0, 1\}$ to B then invoke L-BUFFER-READ to get a vector V_i (if V_i is not empty then $V_i[0]$ is the least recent bit in B). If V_i is empty then p_i outputs u_i. Otherwise p_i outputs $V_i[0]$. See Algorithm 2 for the associated pseudo-code.

Algorithm 2 L-processor binary consensus using only L-buffer objects and read/write registers: code for processor p_i.

L-BUFFER-WRITE(B, u_i)
$V_i \leftarrow$ L-BUFFER-READ(B)
if $V_i = \emptyset$ **then**
  return u_i
else
  return $V_i[0]$
end if

The termination condition holds trivially. The validity condition holds since every bit written to B is an input of some p_i. To see that the agreement condition holds, suppose that p_i was the first to write u_i to B. Then p_i outputs u_i. For all p_j where $i \neq j$, $V_j[0] = u_i$ since there can be at most L bits in B; every non-faulty processor writes once so there can be at most L write operations. □

Lemma 4. *There is no $L+1$ wait-free consensus algorithm for $L+1$ processors using only L-buffer objects and read/write registers.*

Proof. Suppose for a contradiction there exists wait-free consensus algorithm for $L + 1$ processors using only L-buffer objects and read/write registers (to simplify the argument we can restrict our consideration to binary-consensus). We proceed by the valency argument. As discussed in-class, there exists an initial bivalent configuration. We will show that there does not exist a critical configuration C during the execution of the algorithm. Since the algorithm cannot end in a bivalent configuration, the algorithm cannot end, contradicting the termination condition.

Let p_1, ..., p_{L+1} be the $L+1$ processors. Suppose C is a critical configuration. Let α_i be a step by processor p_i from C. Since C is a critical configuration, we can partition the processors into two disjoint, non-empty sets U and V such that for $p_u \in U$, configuration $C_u = C\alpha_u$ is 0-valent and for $p_v \in V$, configuration $C_v = C\alpha_v$ is 1-valent. Let $p_i \in U$ and $p_j \in V$. If α_i and α_j are L-BUFFER-READ or L-BUFFER-WRITE operations to different L-buffer objects then α_i and α_j are commutative. From the perspective of p_i or p_j configurations $C_i\alpha_j$ and $C_j\alpha_i$ are indistinguishable. This is a contradiction since C_i and C_j have different valency. The same is true if α_i and α_j are L-BUFFER-READ operations to the same L-buffer object. Next suppose without loss of generality that α_i is an L-BUFFER-READ operation and α_j is an L-BUFFER-WRITE operation both to L-buffer B. Then $C_j \overset{p_j}{\sim} C_i\alpha_j$. p_j should output the same value starting from C_j and starting from $C_i\alpha_j$, but this is again a contradiction. Thus for every processor p_k, α_k must be an L-BUFFER-WRITE operation to the same L-buffer object B.

Observe that $p_i \in U$ and $p_j \in V$ must write different values to B. Otherwise the operations are again commutative. Thus, WLOG, assume operation α_i writes 0 and operation α_j writes 1 to B. Let $k_1, ..., k_{L-1}$ be some permutation of the indices $\{1, ..., L+1\} - \{i, j\}$. Consider the configurations $C' = C_i\alpha_{k_1} \cdots \alpha_{k_{L-1}}$ and $C'' = C_j\alpha_i\alpha_{k_1} \cdots \alpha_{k_{L-1}}$. There are L objects in B in configuration C' and $L+1$ objects in B in configuration C''. Any L-BUFFER-READ operation executed by p_i will return the most recent L bits in B which are the same in both cases so $C' \overset{p_i}{\sim} C''$.

Since in all cases a pair of configurations originating from C is indistinguishable to some processor even though the configurations have different valency, C cannot be a critical configuration. Thus there does not exists a wait-free consensus algorithm for $L+1$ processors using L-buffer objects and read/write registers. $\square$