1 Lower Bound

In this section we give several lower bounds for algorithms where all logical READs use $\leq log_2(k)$ physical reads.

▶ **Theorem 1.** *For any wait-free algorithm A that simulates a k-valued safe register using binary atomic registers, if the* READ *algorithm uses at most $log_2(k)$ reads in the worst case, then the* WRITE *algorithm uses at least $log_2(k)$ writes in the worst case.*

Proof. Consider the following set S of executions of A: $\sigma_i = \text{WRITE}(v_i)\ \alpha_i\ \text{ACK READ}_1\ \beta_i\ \text{RETURN}(w_i)$ | only the WP takes steps in α_i and only RP_1 takes steps in β_i, $0 \leq i < k$. Note that w_i must equal v_i to satisfy safety.

Now construct a decision tree of RP_1's behavior in the executions in S. The root of the decision tree corresponds to the first register that RP_1 reads, this register is the same for all executions in S as RP_1 always begins in the same state. Now depending on the first value read, RP_1 does some amount of non-read actions then does another read of some register. Because the root only can store two values there are only two registers RP_1 can decide to read from causing the root to have two children. Continue to build the tree like this. Add leaves to indicate which value RP_1 will RETURN.

▶ **Lemma 2.** *The decision tree is a complete binary tree with exactly k leaves in which every leaf is at depth $log_2(k)$ and the path from the root to the leaf labeled v_i corresponds to $\beta_i \in \sigma_i$.*

Proof. Note that there must be at least k leaves in the decision tree, as each value in V is returned in one of the executions in S. Since the decision tree is binary, basic facts from graph theory imply that each leaf must be at depth $log_2(k)$, allowing exactly k leaves and each root-to-leaf path must correspond to a different execution in S. ◀

▶ **Lemma 3.** *None of the* READ*s in an execution in S reads the same register more than once.*

Proof. Suppose in contradiction the READ in σ_i reads some number of registers more than once, for some i. Let x be the first register that RP_1 reads twice, say the a-th read and the b-th read, with $log_2(k) \geq b > a$. Consider the node in the decision tree for the b-th read in the path for σ_i. This node must have two children since the decision tree is complete. Consider a root-to-leaf path π in the decision tree that forks off from the path corresponding to σ_i at this node. By Lemma 1, there is a one-to-one correspondence between root-to-leaf paths in the decision tree and the set of executions and so π corresponds to σ_j in S, for some $j \neq i$.

However, it is not possible for σ_i and σ_j to read different values from x at the b-th read since the only process that is taking steps during the READs is RP_1: even if RP_1 writes to x during the READ, it will write the same value in σ_i as in σ_j, as nothing differs in those two executions until reaching the b-th read. ◀

Now we can finish the proof of the theorem. Let x_1 be the first register read in the READ of each execution in S. Note that in half of the elements of S, the value read from x_1 in the READ must be different from the initial value of x_1, in order to be able to reach all the leaves on that half of the decision tree. In other words, the WRITE writes to x_1 in half the executions in S. Let S_1 be the subset of S consisting of the executions in which the WRITE writes x_1; note that $|S_1| = \frac{k}{2}$. Now let x_2 be the register corresponding to the root of subtree in which x_1 was written. A similar argument shows that in half the executions in S_1, the value read from x_2 in the READ must be different from the initial value of x_2. Let S_2 be the subset of S_1 consisting of the executions in which the WRITE writes to x_2; note that $|S_2| = \frac{k}{2^2}$. Continuing this way we obtain $S_{log_2 k}$, which is of size 1, in which the WRITE writes to $x_1, x_2, ..., x_{log_2 x}$. By Lemma 2, each of these registers is distinct and thus the WRITE writes to at least $log_2(k)$ registers. ◀

▶ **Theorem 4.** *For any wait-free algorithm A that simulates a k-valued regular register using binary atomic registers, if the* READ *algorithm uses at most $log_2(k)$ reads in the worst case, A requires at least $k-1$ registers.*

Proof. Consider the following executions which can be created by shifting RP_1:

- WRITE(v) α_v ACK $\underline{\text{READ}_1\ \beta_v\ \text{RETURN}(v)}$ WRITE(w) α_{w_1} α_{w_2} ACK
- WRITE(v) α_v ACK WRITE(w) α_{w_1} $\underline{\text{READ}_1\ \beta_q\ \text{RETURN}(q)}$ α_{w_2} ACK
- WRITE(v) α_v ACK WRITE(w) α_{w_1} α_{w_2} ACK $\underline{\text{READ}_1\ \beta_w\ \text{RETURN}(w)}$

Because these executions can be created by shifting RP_1, RP_1 can be in the same state each time it initiates its read, we will assume this is the case for these three executions. Due to RP_1 starting in the same state for each execution we can then construct a decision tree for RP_1 in the same manor as what was done in Theorem 1, this tree will be identical for all three executions. Now because RP_1 uses the same algorithm and starts in the same state in each execution there exists a well defined function f which maps the sequence of physical returns in each READ to the value RETURNed by RP_1, where f is the same in each execution. From Lemma 2 each sequence of physical returns which is mapped by f has a length of $log_2(k)$, and because f must map to k distinct values, f must have a $1-1$ mapping of sequences to values. Let γ_v, γ_q, and γ_w represent the sequence of physical returns in β_v, β_q, and β_w respectively. Note that to satisfy regularity $f(\gamma_q) = q$ must equal either v or w. Now let v and w be the values such that the first j elements of γ_v and γ_w are equal where $0 \leq j < log_2(k) - 1$. Additionally the $(j+1)$-th element of γ_v and γ_w are 0 and 1 respectively. This occurs once more, let the J-th element, where $j < J \leq log_2(k)$, of γ_v and γ_w equal 0 and 1 respectively as well. Let t be the node in the decision tree of RP_1 which corresponds to the $(j+1)$-th physical read, note that t must read the same physical register, T, for all three executions. Now let l be the decedent of t in the decision tree which corresponds to the J-th read in the first execution, and let r be the decedent of t in the decision tree which corresponds to the J-th read in the third execution. Note that l and r are on the same level of the decision tree. Finally say that the last physical write in α_{w_1} is the first and only physical write in α_{w_1} which writes to either T or R.

Assume for contradiction that the physical register read by l and r are the same physical register, R.

It is easy to see that the first and third executions can easily be fulfilled even with our previous assumption. However, the second execution is not so simple. As stated previously, in order to uphold regularity we have with two scenarios for the second execution, one where $p = v$ and another where $p = w$.

For the first scenario it is clear that T cannot be written to in α_{w_1} otherwise γ_v, the only sequence which f maps to v, would immediately not equal γ_q. Following this means that R must be the last physical register written to in α_{w_1}. Now in order for p to equal v, the J-th physical read in the second execution must equal what was read in the first execution. However as a result from both executions using the same decision tree, the first and second execution will both read the register R, but in the first execution this read will return 0 and in the second this read will return 1 meaning $p \neq v$ causing a contradiction.

This leaves us with the second possibility, and it is clear that R cannot be written to in α_{w_1} otherwise γ_w, the only sequence which f maps to w, would not equal γ_q. It then follows that T must be the register written to in α_{w_1}. Now because the second and third executions use the same decision tree both of their J-th reads will read the same physical register, the register R. However because R was changed from 0 to 1 in the third execution and remained at 0 in the second execution $\gamma_p \neq \gamma_w$ which implies that $q \neq w$.

This means that for all l and r, if l and r read from the same register there is no way to uphold regularity. In other words no two nodes on the same level of the decision tree which have a common ancestor can read from the same register. Combining this with Lemma 3 implies that no nodes in a decision tree may read from the same physical register. Because the decision tree has exactly $k-1$ nodes, at least $k-1$ registers must be used to implement A. ◀