Brief Announcement: The Space Complexity of Set Agreement Using Swap

Sean Ovens
University of Toronto, Canada

Abstract

We prove that any randomized wait-free n-process k-set agreement algorithm using only swap objects requires at least $\lceil \frac{n}{k} \rceil - 1$ objects. We also sketch a proof that any randomized wait-free consensus algorithm using only readable swap objects with domain size b requires at least $\frac{n-2}{3b+1}$ objects.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases space complexity, consensus, set agreement, lower bound, shared memory

Digital Object Identifier 10.4230/LIPIcs.DISC.2023.46

Related Version *Full Version*: `https://arxiv.org/abs/2305.06507` [9]

Funding Support is gratefully acknowledged from the Natural Sciences and Engineering Research Council of Canada under grant RGPIN-2020-04178 and the Ontario Graduate Scholarships program.

1 Introduction

Consensus is one of the most well-studied problems in distributed computing. In the consensus problem, n processes begin with inputs and collectively agree on a single output. A consensus algorithm satisfies the following properties: agreement (no two outputs may differ) and validity (every output must be an input). There are known randomized wait-free [1, 5] and obstruction-free [10] consensus algorithms using n registers.

In 1993, Ellen, Herlihy, and Shavit [7] proved that $\Omega(\sqrt{n})$ registers are required to solve nondeterministic solo-terminating consensus. Lower bounds for nondeterministic solo-terminating algorithms also apply to randomized wait-free and obstruction-free algorithms. In 2016, Zhu [10] proved that $n-1$ registers are required to solve obstruction-free consensus. Finally, in 2018, Ellen, Gelashvili, and Zhu [6] used a completely new technique to prove that n registers are required to solve obstruction-free consensus. They also showed that space complexity lower bounds for obstruction-free algorithms using readable objects apply to nondeterministic solo-terminating algorithms.

The $\Omega(\sqrt{n})$ space complexity lower bound actually applies to consensus algorithms that use only historyless objects. A *historyless* object can only support two kinds of operations: *trivial* operations, which cannot change the value of the object, and *historyless* operations, which set the object to a fixed value. Registers are historyless objects because *Read* is trivial and *Write* is historyless. *Swap objects* are historyless objects that support *Swap*(v), which returns the current value of the object and then sets its value to v. Any historyless object can be simulated by one *readable swap object*, which supports *Read* and *Swap*(v).

The k-set agreement problem, first defined by Chaudhuri [4], is a generalization of consensus in which n processes begin with inputs and collectively agree on at most k distinct outputs. Obstruction-free k-set agreement is solvable using $n-k+1$ registers [2].

Ellen, Gelashvili, and Zhu [6] proved that any obstruction-free n-process k-set agreement algorithm using only registers requires at least $\lceil \frac{n}{k} \rceil$ registers. Before our results, there were no known non-constant lower bounds on the space complexity of solving k-set agreement using swap objects when $k > 1$.

Last year, we proved that any obstruction-free consensus algorithm using only readable swap objects with domain size 2 requires at least $n-2$ objects [8]. We also proved that any obstruction-free consensus algorithm using only swap objects requires at least $n-1$ objects. We have since refined the techniques from that paper to prove that at least $\frac{n-2}{3b+1}$ readable swap objects with domain size b are required to solve obstruction-free consensus, and that at least $\lceil \frac{n}{k} \rceil - 1$ swap objects are required to solve obstruction-free k-set agreement. In the full version of this paper [9], we also give an obstruction-free k-set agreement algorithm using $n-k$ swap objects, exactly matching our lower bound for $k=1$.

In Section 2, we present some definitions needed to prove our lower bounds. In Section 3, we prove our lower bound for set agreement using swap objects. In Section 4, we sketch the proof of our lower bound for consensus using readable swap objects with bounded domains.

2 Definitions

Two configurations C and C' are *indistinguishable* to a process p_i, denoted $C \overset{p_i}{\sim} C'$, if and only if p_i has the same state in C and C'. Two executions α and α' from C and C', respectively, are *indistinguishable* to a process p_i if $C \overset{p_i}{\sim} C'$ and p_i takes the same sequence of steps (and obtains the same responses) in α and α'. We use *value*(B, C) to denote the value of the object B in configuration C.

In the *m-valued k-set agreement* problem, all process inputs are in $\{0, \ldots, m-1\}$. The 2-valued 1-set agreement problem is also called *binary consensus*. A nonempty set of processes $\mathcal{P}$ is *v-univalent* in a configuration of a binary consensus algorithm C if, for every $\mathcal{P}$-only execution from C in which some process $p \in \mathcal{P}$ decides, v is the value decided by p. If $\mathcal{P}$ is neither 0-univalent nor 1-univalent in C, then $\mathcal{P}$ is *bivalent* in C.

An algorithm is *nondeterministic solo-terminating* if, for every configuration C of the algorithm and every process p, there is a solo-terminating execution by p from C. An algorithm is *obstruction-free* if it is deterministic and nondeterministic solo-terminating. An algorithm is *randomized wait-free* if, for every scheduler, the expected number of steps in any execution produced by that scheduler is finite.

3 Lower Bound for Set Agreement Using Swap Objects

Our lower bound relies on the following technical lemma.

▶ **Lemma 1.** *Consider an initial configuration C in which a set of processes $\mathcal{Q}$ have the same input v. Let α be an execution from C that does not involve $\mathcal{Q}$ such that k distinct values different from v are decided in $C\alpha$. Then the algorithm uses at least $|\mathcal{Q}|$ swap objects.*

Proof. Let $\mathcal{Q} = \{q_1, \ldots, q_{|\mathcal{Q}|}\}$. Define $\mathcal{Q}_i = \{q_1, \ldots, q_i\}$, for $1 \leq i \leq |\mathcal{Q}|$, and define $\mathcal{Q}_0 = \emptyset$. Let D be an initial configuration in which all processes have input v. For $0 \leq i \leq |\mathcal{Q}|$, we show that there is a set of i swap objects $\mathcal{A}_i$ and a pair of $\mathcal{Q}_i$-only executions γ_i and δ_i from $C\alpha$ and D, respectively, such that *value*$(B, C\alpha\gamma_i) =$ *value*$(B, D\delta_i)$, for all $B \in \mathcal{A}_i$. For $i = |\mathcal{Q}|$, this claim proves the lemma. We use induction on i. When $i = 0$, γ_i and δ_i are empty executions, $\mathcal{A}_i = \emptyset$, and the claim is trivially satisfied.

Now let $0 \leq i < |\mathcal{Q}|$ and suppose there exists γ_i, δ_i, and $\mathcal{A}_i$ such that *value*$(B, C\alpha\gamma_i) =$ *value*$(B, D\delta_i)$, for all $B \in \mathcal{A}_i$. Notice that $C\alpha\gamma_i \overset{q_{i+1}}{\sim} D\delta_i$, since q_{i+1} has input v in both configurations and takes no steps in α, γ_i, or δ_i. Consider a q_{i+1}-only solo-terminating execution σ from $D\delta_i$. By validity, q_{i+1} decides v in σ. Let τ be the longest prefix of σ such that q_{i+1} only accesses objects in $\mathcal{A}_i$ during τ. Since *value*$(B, C\alpha\gamma_i) =$ *value*$(B, D\delta_i)$, for all

$B \in \mathcal{A}_i$, there is a q_{i+1}-only execution τ' from $C\alpha\gamma_i$ such that $\tau' \overset{q_{i+1}}{\sim} \tau$. If $\tau = \sigma$, then q_{i+1} decides v in τ and τ'. This is impossible, since $k+1$ different values are decided in $C\alpha\gamma_i\tau'$. Thus, τ is a proper prefix of σ. Then in $C\alpha\gamma_i\tau'$ and $D\delta_i\tau$, q_{i+1} is poised to apply a *Swap* operation s to an object $B^\star \notin \mathcal{A}_i$.

Since q_{i+1} applies the same sequence of operations in τ' and τ and $value(B, C\alpha\gamma_i) = value(B, D\delta_i)$ for all $B \in \mathcal{A}_i$, it follows that $value(B, C\alpha\gamma_i\tau') = value(B, D\delta_i\tau)$ for all $B \in \mathcal{A}_i$. By definition of *Swap*, $value(B^\star, C\alpha\gamma_i\tau's) = value(B^\star, D\delta_i\tau s)$. Taking $\gamma_{i+1} = \gamma_i\tau's$, $\delta_{i+1} = \delta_i\tau s$, and $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{B^\star\}$ completes the inductive step. ◀

We can now apply Lemma 1 to obtain the desired lower bound.

▶ **Theorem 2.** *For all $n > k \geq 1$, every nondeterministic solo-terminating, n-process $(k+1)$-valued k-set agreement algorithm from swap objects uses at least $\lceil \frac{n}{k} \rceil - 1$ objects.*

Proof. Consider such an algorithm for the set of processes $\mathcal{P} = \{p_0, \dots, p_{n-1}\}$. We will use induction on k. When $k = 1$, the algorithm solves binary consensus. Consider an initial configuration C of the algorithm in which process p_0 has input 0 and all other processes have input 1. Note that p_0 decides 0 in its solo-terminating execution α from C. Therefore, by Lemma 1, the algorithm uses at least $n-1$ swap objects.

Now let $1 < k < n$ and suppose the theorem holds for $k-1$. Let $\mathcal{R}$ be some set of $\lceil \frac{n(k-1)}{k} \rceil$ processes in $\mathcal{P}$. Let $\mathcal{I}$ be the set of all initial configurations in which $\mathcal{R}$'s inputs are in $\{0, \dots, k-1\}$. If, for every initial configuration $C \in \mathcal{I}$ and every $\mathcal{R}$-only execution α from C, at most $k-1$ different values are decided in α, then the algorithm solves nondeterministic solo-terminating k-valued $(k-1)$-set agreement among the processes in $\mathcal{R}$. Hence, by the inductive hypothesis, the algorithm uses at least $\lceil \frac{|\mathcal{R}|}{k-1} \rceil - 1 = \lceil \frac{n}{k} \rceil - 1$ swap objects.

Otherwise, there is a $C \in \mathcal{I}$ and an $\mathcal{R}$-only execution α from C in which $0, \dots, k-1$ are decided. Notice $|\mathcal{P} - \mathcal{R}| = n - \left\lceil \frac{n(k-1)}{k} \right\rceil = \left\lfloor \frac{n}{k} \right\rfloor \geq \left\lceil \frac{n}{k} \right\rceil - 1$. By Lemma 1 (with $\mathcal{Q} = \mathcal{P} - \mathcal{R}$ and $v = k$), the algorithm uses at least $|\mathcal{P} - \mathcal{R}| \geq \lceil \frac{n}{k} \rceil - 1$ swap objects. ◀

4 Lower Bound for Consensus Using Readable Swap Objects

Let $\mathcal{Q} = \{q_0, q_1\}$ be a pair of processes and let $\mathcal{P} = \{p_0, \dots, p_{n-3}\}$ be the rest of the processes. For all $0 \leq i \leq n-3$, define $\mathcal{P}_i = \{p_i, \dots, p_{n-3}\}$. In particular, $\mathcal{P}_0 = \mathcal{P}$. Define $\mathcal{P}_{n-2} = \emptyset$. Let $\mathcal{A}$ be the set of all readable swap objects with domain size b used by the algorithm.

A set S of processes *covers* a set $\mathcal{B}$ of objects in a configuration C if, for every object $B \in \mathcal{B}$, there is a unique process in S that is poised to apply a *Swap* to B in C. A *block swap* by S is an execution that consists of a single step by each process in S.

▶ **Lemma 3** ([8]). *Let C be a configuration in which $\mathcal{Q}$ is bivalent and a set $S \subseteq \mathcal{P}$ of processes cover a set $\mathcal{B}$ of readable swap objects. Then there is a $\mathcal{Q}$-only execution γ from C such that $\mathcal{Q}$ is bivalent in $C\gamma\beta$, where β is a block swap by S.*

The following result uses Lemma 3 and appears in the full version of the paper [9].

▶ **Lemma 4.** *Let $p_i \in \mathcal{P}$, let C be a configuration in which $\mathcal{Q}$ is bivalent, let C' be a configuration such that $C \overset{p_i}{\sim} C'$, and let δ be p_i's solo-terminating execution from C'. Suppose δ consists of r steps and, for all $s \in \{0, \dots, r\}$, let δ_s be the prefix of δ that consists of the first s steps by p_i. Then there is a $j \in \{0, \dots, r-1\}$ such that,*

(a) *for all $j' \in \{0, \dots, j\}$, there is a $(\mathcal{Q} \cup \mathcal{P}_i)$-only execution $\alpha_{j'}$ from C such that $\mathcal{Q}$ is bivalent in $C\alpha_{j'}$ and $\alpha_{j'} \overset{p_i}{\sim} \delta_{j'}$.*

Consider any $(\mathcal{Q} \cup \mathcal{P}_i)$*-only execution* α_j *from* C *such that* $\mathcal{Q}$ *is bivalent in* $C\alpha_j$ *and* $\alpha_j \overset{p_i}{\sim} \delta_j$*. Let* d *be the operation that* p_i *is poised to apply to the object* B *in* $C'\delta_j$*. Then for every* $(\mathcal{Q} \cup \mathcal{P}_{i+1})$*-only execution* λ' *from* $C\alpha_j$*,*

(b) *if value*$(B, C\alpha_j\lambda') =$ *value*$(B, C'\delta_j)$*, then* $\mathcal{Q}$ *is univalent in* $C\alpha_j\lambda'd$*, and*

(c) *if value*$(B, C'\delta_j) =$ *value*$(B, C'\delta_j d)$ *and in some configuration of* λ' *the value of* B *is value*$(B, C'\delta_j)$*, then* $\mathcal{Q}$ *is univalent in* $C\alpha_j\lambda'$*.*

We now sketch a proof of our main technical lemma. For all $0 \leq i \leq n-2$, it constructs a configuration C_i and two functions f_i and g_i that map objects to increasingly large sets of values that are forbidden in certain executions from C_i.

▶ **Lemma 5.** *For all* $0 \leq i \leq n-2$*, there is a configuration* C_i *reachable from* C_0*, a set of processes* $S_i \subseteq \mathcal{P} - \mathcal{P}_i$*, and a pair of functions* f_i, g_i *that map objects to subsets of* $\{0, \ldots, b-1\}$ *such that the following holds for every* $(\mathcal{Q} \cup \mathcal{P}_i)$*-only execution* λ *from* C_i*:*

(a) $\mathcal{Q}$ *is bivalent in* C_i*,*

(b) S_i *covers a set of* $|S_i|$ *objects in* C_i*,*

(c) *for every process* $p \in S_i$*, if* p *is poised to apply a* Swap(B, x) *operation in* C_i*, then* $x \notin f_i(B) \cup g_i(B)$*,*

(d) $\sum_{B \in \mathcal{A}} \big(2 \cdot |f_i(B)| + |g_i(B)|\big) + |S_i| \geq i$*,*

(e) *if the value of some object* B *is equal to some value in* $f_i(B)$ *in some configuration of* λ*, then* $\mathcal{Q}$ *is univalent in* $C_i\lambda$*, and*

(f) *if some process* $p \in \mathcal{P}_i$ *is poised to apply a* Swap(B, x) *operation in* $C_i\lambda$ *for some object* B *and some* $x \in g_i(B)$*, then* $\mathcal{Q}$ *is univalent in* $C_i\lambda$*.*

Proof sketch. We use induction on i. Let $S_0 = \emptyset$ and let $f_0(B) = g_0(B) = \emptyset$ for all $B \in \mathcal{A}$. All properties of the lemma are simple to verify for $i = 0$.

Now suppose that the lemma holds for some $0 \leq i \leq n-3$. Let δ be p_i's solo-terminating execution from $C_i\beta_i$, where β_i is a block swap by S_i. Suppose that δ consists of r steps by p_i, and let δ_s be the prefix of δ that consists of its first s steps. Let $0 \leq j \leq r-1$ be the value that satisfies the conditions of Lemma 4 (with $C = C_i$ and $C' = C_i\beta_i$).

In the full version of this paper [9], we prove that, for all $B \in \mathcal{A}$ and all forbidden values $x \in f_i(B) \cup g_i(B)$, process p_i does not apply any *Swap*(B, x) operations during δ_{j+1}.

Let d be the final step of δ_{j+1} by p_i. Let $B^\star$ be the object accessed by p_i in step d. Let $v^\star =$ *value*$(B^\star, C_i\beta_i\delta_j)$. By Lemma 4(a), there is a $(\mathcal{Q} \cup \mathcal{P}_i)$-only execution α_j from C_i such that $\mathcal{Q}$ is bivalent in $C_i\alpha_j$ and $\alpha_j \overset{p_i}{\sim} \delta_j$. Define $C_{i+1} = C_i\alpha_j$, so property (a) holds for $i+1$.

Case 1. Step d does not change the value of $B^\star$ when it is applied in $C_i\beta_i\delta_j$. Define $f_{i+1}(B) = f_i(B)$ for all $B \in \mathcal{A} - \{B^\star\}$, $g_{i+1}(B) = g_i(B)$ for all $B \in \mathcal{A}$, and $f_{i+1}(B^\star) = f_i(B^\star) \cup \{v^\star\}$.

If there is a process $p \in S_i$ that is poised to apply a *Swap*$(B^\star, v^\star)$ operation in C_i, then define $S_{i+1} = S_i - \{p\}$. Otherwise, define $S_{i+1} = S_i$. Since no process in S_i takes steps during α_j and $S_{i+1} \subseteq S_i$, property (b) holds for $i+1$. In addition, property (c) holds for i, so it holds for $i+1$ as well.

Suppose $v^\star \in f_i(B^\star)$. Then process p_i does not apply *Swap*$(B^\star, v^\star)$ during δ_{j+1}. Hence, *value*$(B^\star, C_i\beta_i) = v^\star$. Property (c) for i implies that no process applies *Swap*$(B^\star, v^\star)$ during β_i. Thus, *value*$(B^\star, C_i) = v^\star$. By property (e) for i (where λ is the empty execution), $\mathcal{Q}$ is univalent in C_i. This contradicts property (a) for i. Hence, $v^\star \notin f_i(B^\star)$. This implies that $|f_{i+1}(B^\star)| = |f_i(B^\star)| + 1$. Since $|S_{i+1}| \geq |S_i| - 1$, property (d) holds for $i+1$.

Let λ be a $(\mathcal{Q} \cup \mathcal{P}_{i+1})$-only execution from C_{i+1}. Then $\alpha_j\lambda$ is a $(\mathcal{Q} \cup \mathcal{P}_i)$-only execution from C_i. By property (e) for i, if the value of some object B is equal to a value in $f_i(B)$ in any configuration of $\alpha_j\lambda$, then $\mathcal{Q}$ is univalent in $C_i\alpha_j\lambda$. Lemma 4(c) (with $\lambda' = \lambda$) implies that, if the value of $B^\star$ is $v^\star$ in some configuration of λ, then $\mathcal{Q}$ is univalent in $C_i\alpha_j\lambda$. This gives us property (e) for $i+1$. Since $g_{i+1}(B) = g_i(B)$ for all $B \in \mathcal{A}$, property (f) for $i+1$ follows from property (f) for i.

Case 2. Step d changes the value of $B^\star$ when it is applied in $C_i\beta_i\delta_j$. Then d is a $\mathit{Swap}(B^\star, v')$ operation, for some $v' \in \{0, \ldots, b-1\} - \{v^\star\}$. Define $f_{i+1}(B) = f_i(B)$ for all $B \in \mathcal{A}$, $g_{i+1}(B) = g_i(B)$ for all $B \in \mathcal{A} - \{B^\star\}$, and $g_{i+1}(B^\star) = g_i(B^\star) \cup \{v^\star\}$.

If some process $p \in S_i$ is poised to access $B^\star$ in C_i, then define $S_{i+1} = (S_i - \{p\}) \cup \{p_i\}$. Otherwise, define $S_{i+1} = S_i \cup \{p_i\}$. In either case, we obtain property (b) for $i+1$.

Since p_i does not apply $\mathit{Swap}(B, x)$ during δ_{j+1}, for any $B \in \mathcal{A}$ and any $x \in f_i(B) \cup g_i(B)$, it follows that $v' \notin f_i(B^\star) \cup g_i(B^\star)$. Furthermore, we know that $v' \neq v^\star$. Hence, $v' \notin f_{i+1}(B^\star) \cup g_{i+1}(B^\star)$. This along with property (c) for i gives us property (c) for $i+1$.

If $B^\star$ is not covered by S_i in C_i, then $S_{i+1} = S_i \cup \{p_i\}$, so $|S_{i+1}| = |S_i| + 1$. Furthermore, $|g_{i+1}(B^\star)| \geq |g_i(B^\star)|$. Property (d) for $i+1$ follows from this and property (d) for i.

Otherwise, $B^\star$ is covered by S_i in C_i. By property (c) for i, $\mathit{value}(B, C_i\beta_i) \notin f_i(B) \cup g_i(B)$ for all objects B covered by S_i in C_i. Furthermore, p_i does not apply $\mathit{Swap}(B, x)$ during δ_{j+1}, for any $B \in \mathcal{A}$ and any $x \in f_i(B) \cup g_i(B)$. Hence, $\mathit{value}(B, C_i\beta_i\delta_j) \notin f_i(B) \cup g_i(B)$ for all objects B covered by S_i in C_i. Since $B^\star$ is covered by S_i in C_i, it follows that $v^\star \notin f_i(B^\star) \cup g_i(B^\star)$. Hence, $|g_{i+1}(B^\star)| = |g_i(B^\star)| + 1$. Furthermore, $|S_{i+1}| = |S_i|$. Combined with property (d) for i, this gives us property (d) for $i+1$.

Let λ be a $(\mathcal{Q} \cup \mathcal{P}_{i+1})$-only execution from C_{i+1}. Then $\alpha_j\lambda$ is a $(\mathcal{Q} \cup \mathcal{P}_i)$-only execution from C_i. Since $f_{i+1}(B) = f_i(B)$ for all $B \in \mathcal{A}$, property (e) for $i+1$ follows from property (e) for i. Suppose there is a $p \in \mathcal{P}_{i+1}$ poised to apply a $\mathit{Swap}(B^\star, v^\star)$ operation t in $C_{i+1}\lambda$. Recall that $\alpha_j \overset{p_i}{\sim} \delta_j$, so p_i is poised to apply d in $C_i\alpha_j = C_{i+1}$. Since p_i takes no steps in λ, it is poised to apply d in $C_{i+1}\lambda$. Since d is a $\mathit{Swap}(B^\star, v')$ operation, p_i covers $B^\star$ in $C_{i+1}\lambda$. If $\mathcal{Q}$ is bivalent in $C_{i+1}\lambda$, then, by Lemma 3 (with $C = C_{i+1}\lambda$ and $S = \{p_i\}$), there is a $\mathcal{Q}$-only execution γ from $C_{i+1}\lambda$ such that $\mathcal{Q}$ is bivalent in $C_{i+1}\lambda\gamma d$ and, hence, in $C_{i+1}\lambda\gamma t d$. However, Lemma 4(b) (with $\lambda' = \lambda\gamma t$) implies that $\mathcal{Q}$ is univalent in $C_{i+1}\lambda\gamma t d$. Hence, $\mathcal{Q}$ is univalent in $C_{i+1}\lambda$. Property (f) for $i+1$ follows from this and property (f) for i. ◀

Lemma 5(d) (with $i = n-2$) says that $\sum_{B \in \mathcal{A}} (2 \cdot |f_{n-2}(B)| + |g_{n-2}(B)|) + |S_{n-2}| \geq n-2$. By part (b), S_{n-2} covers a set of $|S_{n-2}|$ objects in C_{n-2}. Hence, $|S_{n-2}| \leq |\mathcal{A}|$. Moreover, $\sum_{B \in \mathcal{A}} (2 \cdot |f_{n-2}(B)| + |g_{n-2}(B)|) \leq 3 \cdot b \cdot |\mathcal{A}|$ since $f_{n-2}(B)$ and $g_{n-2}(B)$ are subsets of $\{0, \ldots b-1\}$. Thus, $3 \cdot b \cdot |\mathcal{A}| + |\mathcal{A}| \geq n-2$, which implies the lower bound.

▶ **Theorem 6.** *For all $n, b \geq 2$, any n-process, obstruction-free binary consensus algorithm from readable swap objects with domain size b uses at least $\frac{n-2}{3b+1}$ objects.*

5 Conclusion

Determining the exact space complexity of solving obstruction-free k-set agreement using swap objects when $k > 1$ remains an open problem. We conjecture that $n - k$ swap objects are required. Theorem 6 implies that any obstruction-free consensus algorithm from readable swap objects with constant-sized domain requires $\Omega(n)$ objects. This asymptotically matches Bowman's [3] algorithm, which uses $2n - 1$ registers with domain size 2.

References

1 James Aspnes and Maurice Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 11(3):441–461, 1990. doi:10.1016/0196-6774(90)90021-6.

2 Zohir Bouzid, Michel Raynal, and Pierre Sutra. Anonymous obstruction-free (n, k)-set agreement with n-k+1 atomic read/write registers. *Distributed Comput.*, 31(2):99–117, 2018. doi:10.1007/s00446-017-0301-7.

3 Jack R. Bowman. Obstruction-free snapshot, obstruction-free consensus, and fetch-and-add modulo k. Master's thesis, Dartmouth College, Computer Science, 2011. URL: https://digitalcommons.dartmouth.edu/senior_theses/67.

4 S. Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, 1993. doi:10.1006/inco.1993.1043.

5 B. Chor, A. Israeli, and M. Li. Wait-free consensus using asynchronous hardware. *SIAM J. Comput.*, 23:701–712, 1994.

6 Faith Ellen, Rati Gelashvili, and Leqi Zhu. Revisionist simulations: A new approach to proving space lower bounds. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18, pages 61–70, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3212734.3212749.

7 Faith Fich, Maurice Herlihy, and Nir Shavit. On the space complexity of randomized synchronization. *J. ACM*, 45(5):843–862, September 1998. A preliminary version appeared in PODC '93. doi:10.1145/290179.290183.

8 Sean Ovens. The space complexity of consensus from swap. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, pages 176–186, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538420.

9 Sean Ovens. The space complexity of consensus from swap. *CoRR*, abs/2305.06507, 2023. arXiv:2305.06507.

10 Leqi Zhu. A tight space bound for consensus. *SIAM J. Comput.*, 50(3), 2019. A preliminary version appeared in STOC '16. doi:10.1137/16M1096785.