A Tight Space Bound for Consensus

Leqi Zhu
Department of Computer Science
University of Toronto
Canada
lezhu@cs.toronto.edu

ABSTRACT

Existing n-process randomized wait-free (and obstruction-free) consensus protocols from registers all use at least n registers. In 1992, it was proved that such protocols must use $\Omega(\sqrt{n})$ registers. Recently, this was improved to $\Omega(n)$ registers in the anonymous setting, where processes do not have identifiers. Closing the gap in the general case, however, remained an open problem. We resolve this problem by proving that every randomized wait-free (or obstruction-free) consensus protocol for n processes must use at least $n-1$ registers.

CCS Concepts

• **Theory of computation → Concurrency**

Keywords

Shared Memory Model, Consensus, Space Complexity

1. INTRODUCTION

Perhaps the most studied problem in the theory of distributed computing is the *consensus* problem, which requires n processes, each with an input value, to agree on a common output value. An attractive application of the consensus problem lies in implementing shared objects, such as stacks or queues. In particular, if there is a *wait-free* protocol for consensus, where each process decides in a finite number of its own steps, regardless of the speed or failure of other processes, then it is also possible to implement any shared object in a wait-free manner [Her91].

It is impossible to deterministcally solve wait-free consensus in an asynchronous shared memory system, where processes communicate by reading and writing shared memory locations, called *registers* [LAA87]. However, it is possible using randomization [AC08, AH90, AW96, CIL94]. Asymptotically tight bounds are known for the total number of steps taken by all processes [AC08].

STOC '16, June 18-21, 2016, Cambridge, MA, USA

DOI: http://dx.doi.org/10.1145/2897518.2897565

On the other hand, tight bounds were not known for the space complexity of this problem. In 1992, Fich, Herlihy, and Shavit proved a space lower bound of $\Omega(\sqrt{n})$ registers [FHS98]. All existing protocols use at least n registers [AH90, AW96]. Closing this gap has been a longstanding open problem.

Recently, we proved matching upper and lower bounds of n registers for a restricted class of protocols, where processes are *anonymous* (i.e. they have no identifiers) and *memoryless* (i.e. they do not use local memory) [Zhu15]. At the same time, using very interesting, different techniques, Gelashvili proved a lower bound of $\Omega(n)$ registers for protocols with anonymous processes, *without* the memoryless assumption [Gel15]. Since there are anonymous protocols that use n registers [BRS15, Zhu15], the bound is tight. Thus, the anonymous case of the problem is resolved to within a constant factor.

The general case of the problem, however, remained open. There was even evidence suggesting the possibility of a protocol using $O(\sqrt{n})$ space: *Weak leader election* is a closely related, but provably weaker, problem. In this problem, processes must choose exactly one leader, but each process only needs to know whether it has been chosen. An innovative protocol for weak leader election, using $O(\sqrt{n})$ registers, was obtained [GHHW13] a few years ago. Later, the same authors improved this to $O(\log n)$, which is optimal [GHHW15].

Our contribution.

We resolve the general case of the problem by proving that any consensus protocol for n processes in an asynchronous system uses at least $n-1$ registers. Our lower bound uses a more refined notion of *valency* (introduced in [FLP85]) combined with a *covering* argument (introduced in [BL93]). As in [FHS98, Gel15, Zhu15], the bound holds even if the registers are of unbounded size.

The lower bound shows that consensus is, fundamentally, a communication problem. In particular, having large registers cannot compensate for having too few registers. Since there is a memoryless anonymous protocol that uses n registers [Zhu15], having identifiers and large amounts of local memory also cannot compensate for having too few registers.

A nice feature of our proof is that it is simple and uses very little machinery. This is a bit surprising given the difficulty of the $\Omega(\sqrt{n})$ lower bound and the subtlety of the $\Omega(n)$ lower bound for the anonymous case.

2. PRELIMINARIES

In this section, we present concepts that are necessary for understanding our proof of the lower bound. In Section 2.1, we discuss the consensus problem and the model of computation. In Section 2.2, to make the model more concrete, we describe a simple consensus protocol. Finally, in Section 2.3, we present notation and terminology for describing executions of a protocol.

2.1 Consensus in a Shared Memory System

We consider an asynchronous shared memory system with $n \geq 2$ processes. Each processes has a unique identifier. The processes run at arbitrary speeds and may fail, at any time, by crashing. To communicate, they read from and write to shared memory locations, called *registers*.

In the *consensus* problem, each process has a local input value and they want to collectively decide on a single output value. A process *decides* a value by writing it to a special location in its local memory and then terminating.

A solution to the consensus problem is called a *consensus protocol*. Every consensus protocol must satisfy the following two properties:

- *Validity*: If a process decides a value, then that value is the input value of some process.
- *Agreement*: No two processes decide different values.

A consensus protocol *terminates* when every non-faulty process has decided a value. Processes that crash are not required to decide a value. A protocol is *wait-free* if every non-faulty process decides a value after taking a finite number of steps, regardless of what other processes are doing. A well-known result is that there is no consensus protocol that is both deterministic and wait-free [LAA87]. It is possible to circumvent this by weakening the termination condition.

A consensus protocol is *randomized wait-free* if every non-faulty process decides a value with probability approaching 1 after taking a finite number of steps. A consensus protocol is *obstruction-free* if every non-faulty process decides a value after taking a finite number of *consecutive* steps (i.e. no other process takes a step in between these steps).

We consider consensus protocols satisfying the following termination condition, which subsumes randomized wait-free and obstruction-free: A consensus protocol is *nondeterministic solo terminating* if every process *can* decide a value after taking a finite number of consecutive steps. A more precise definition of this appears in Section 2.3.

2.2 An Example Consensus Protocol

To illustrate the model, we describe, in Figure 1, a simple *binary* consensus protocol for $n \geq 2$ processes which uses $2n-1$ registers. The input value of each process is assumed to be either 0 or 1. The protocol is adapted from [Bow11].

Intuitively, a process attempts to fill the registers with its preference (either 0 or 1) to convince the other processes to decide its value. Initially, its preference is its input value and the process writes this value, along with its identifier, to the first register. Then, it reads all $2n-1$ registers, one by one. If the process sees that value v appears more times than value $\overline{v}$ in the registers, then it sets its preference to v. Otherwise, it randomly picks $v \in \{0,1\}$ and sets its preference to v. Afterwards, it checks whether the registers all contain its preference and identifier. If this is the case, then it decides its preference and terminates. Otherwise, it writes the pair containing its preference and identifier to the first register that does not contain this pair and repeats.

```
1:  preference ← input
2:  write (preference, identifier) to register 1
3:  loop
4:      a[1..2n − 1] = (⊥, . . . , ⊥)
5:      (prefer_0, prefer_1) ← (0, 0)
6:      for i = 1..2n − 1 do
7:          a[i] ← read from register i
8:          if a[i] ≠ ⊥ then
9:              v ← a[i].preference
10:             prefer_v ← prefer_v + 1
11:     if ∃v ∈ {0, 1}: prefer_v > prefer_v̄ then
12:         preference ← v
13:     else
14:         preference ← random v ∈ {0, 1}
15:     if ∀i ∈ [2n − 1]: a[i] = (preference, identifier) then
16:         decision ← preference
17:         terminate
18:     i ← min{i ∈ [2n − 1] : a[i] ≠ (preference, identifier)}
19:     write (preference, identifier) to register i
```

Figure 1: A nondeterministic solo terminating binary consensus protocol for $n \geq 2$ processes which uses $2n-1$ registers.

It can be shown that every process decides a value v after at most $(2n-1)^2 + 2n - 1$ consecutive steps and that v is the input value of some process. Hence, the protocol is nondeterministic solo terminating and satisfies Validity.

Now, consider the *first* process to decide a value. It saw that all $2n-1$ registers contained its preference, v, and its identifier. Since there are only $n-1$ other processes, it follows that these processes will see that v occurs more times than $\overline{v}$ (since $2n-1-(n-1) = n > n-1$) when they next perform the loop in lines 6 and 7. Thus, they will all set their preferences to v. From that point on, the number of registers that contain $\overline{v}$ never increases. It follows that $\overline{v}$ is never decided. Hence, the protocol satisfies Agreement.

2.3 Terminology and Notation

We now describe some common terminology and notation for describing executions of a fixed consensus protocol Π, borrowing heavily from [AE14]. We will refer to the protocol in Figure 1 for examples.

Formally speaking, a consensus protocol specifies a set of possible states for each process. In each state, there is at most one next *step* that the process may take. Each step is either a read or write of a register. After taking a step, the process can change state, based on its previous state, the response it received from its step, and its local computation. If, from some state, given some response, a process has multiple possible next states, then the protocol is *nondeterministic*. Otherwise, the protocol is *deterministic*.

To illustrate this, consider the protocol in Figure 1. Lines 2, 7, and 19 denote steps. The rest is local computation. The protocol is nondeterministic since, from some states, a process may have two possible next states, even if it receives the same response. In particular, if the number of 1's in the registers is equal to the number of 0's, then a process may change its preference to either 0 or 1.

A *configuration* of Π consists of the state of each process and the contents of each register. An *initial* configuration is determined by the input value of each process. The contents of the registers are the same in all initial configurations.

A configuration C is *indistinguishable* from a configuration C' to a set of processes P if every process in P is in the same state in C as it is in C' and each register has the same contents in C as it does in C'.

A step e by a process p is *applicable* at a configuration C of Π if e is the next step of process p given its state in C. If e is a read from a register r, then e returns the contents of r in C. Otherwise, if e writes the value v to register r, then the contents of register r is set to v and e returns an acknowledgement. If e is applicable at C, then we use Ce to denote the configuration resulting from p taking step e.

A sequence of steps $\alpha = e_1, e_2, \ldots$ is *applicable* at a configuration C of Π if e_1 is applicable at C and, for each $i \geq 1$, e_{i+1} is applicable at $Ce_1 \cdots e_i$. In this case, α is an *execution* from C. A configuration C of Π is *reachable* if there exists a finite execution from an initial configuration of Π that results in C.

For a finite execution α from a configuration C of Π, we use $C\alpha$ to denote the configuration reached after applying α to C. Note, if α is empty, then $C\alpha = C$. We say an execution α is *P-only*, for a set of processes P, if all steps in α are by processes in P. Note, if configurations C and C' are indistinguishable to a set of processes P, then any P-only execution from C is applicable at C'.

Using this terminology, a consensus protocol Π is *nondeterminstic solo terminating* if, for every process, p, and every reachable configuration, C, of Π, there exists a $\{p\}$-only execution α from C such that p has decided a value (and terminated) in $C\alpha$.

3. LOWER BOUND

Let Π be any nondeterministic solo terminating binary consensus protocol for $n \geq 2$ processes. In this section, we show that Π uses at least $n-1$ registers, even if the registers are of unbounded size. In Section 3.1, we describe a more refined notion of valency. In Section 3.2, we extend some traditional ideas of covering arguments, taking into account our notion of valency. The proof of the main result is presented in Section 3.3.

3.1 Valency

The notion of the *valency* of a configuration was introduced by Fischer, Lynch, and Paterson [FLP85]. Informally, they consider the values that can decided by processes from a reachable configuration of a binary consensus protocol. The configuration is *bivalent* if both 0 and 1 can be decided. Otherwise, the configuration is *univalent*.

We refine their notion of valency by considering the values that specific (non-empty) subsets of the processes can decide from a reachable configuration. In this view, the notion of valency is no longer attached to the entire configuration, but to subsets of processes in the configuration.

Definition 1. Let C be a reachable configuration of Π, and let P be a non-empty set of processes. *P can decide $v \in \{0,1\}$ from C* if there exists a P-only execution from C in which v is decided. If P can decide *both* 0 and 1 from C, then *P is bivalent from C*. If P can decide v, but not $\overline{v}$, from C, then *P is v-univalent from C*.

The following facts are easy consequences of Definition 1.

PROPOSITION 1. *Let C be a reachable configuration and let P be a non-empty set of processes.*

(i) P can decide some value from C.

(ii) If P can decide $v \in \{0,1\}$ from C, then any superset of P can decide v from C.

(iii) If P is v-univalent from C, then every non-empty subset of P is v-univalent from C.

(iv) If φ is an execution from C in which $v \in \{0,1\}$ is decided, then P is v-univalent from $C\varphi$.

A standard part of a valency argument is to show that there is an initial bivalent configuration. In the next proposition, this is stated a bit more carefully using our notion of valency.

PROPOSITION 2. *There is an initial configuration I of Π and processes p_0 and p_1 such that, for each $v \in \{0,1\}$, $\{p_v\}$ is v-univalent from I, hence, $\{p_0, p_1\}$ is bivalent from I.*

PROOF. For $v \in \{0,1\}$, let I_v be the initial configuration where every process starts with v. By the Validity property of consensus, every process is v-univalent from I_v. Consider an initial configuration I where process p_0 starts with input 0 and process p_1 starts with input 1. The rest of the processes may start with any input. Since no processes have taken steps, for each $v \in \{0,1\}$, I is indistinguishable from I_v to p_v. Thus, $\{p_v\}$ is v-univalent from I. □

The fact that a set of processes P is bivalent from a configuration C is not very helpful on its own. Indeed, the P-only executions which decide 0 (or 1) may be very complex and involve many processes. It is possible that the exclusion of any particular process in P may suddenly make the remaining processes univalent from C. Our next lemma shows that this problem can be avoided, so we can do induction arguments on $|P|$. It is one of the reasons why we introduced a notion of valency defined for subsets of processes.

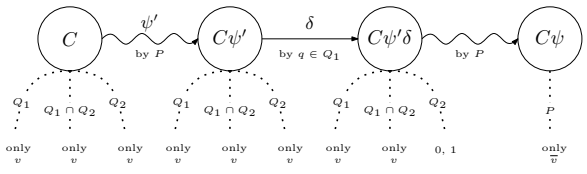

Figure 2: Diagram of configurations for Lemma 1.

LEMMA 1. *Let C be a reachable configuration of Π and let P be a set of processes with $|P| \geq 3$. If P is bivalent from C, then there exists a P-only execution φ from C and a process $z \in P$ such that $P - \{z\}$ is bivalent from $C\varphi$.*

PROOF. Pick any two processes $z_1, z_2 \in P$. Let $Q_1 = P - \{z_1\}$ and $Q_2 = P - \{z_2\}$. Since $|P| \geq 3$, $|Q_1| + |Q_2| = 2|P| - 2 > |P|$ so $Q_1 \cap Q_2 \neq \emptyset$ and $Q_1 \cup Q_2 = P$.

By Proposition 1(i), $Q_1 \cap Q_2$ can decide $v \in \{0,1\}$ from C. Hence, by Proposition 1(ii), both Q_1 and Q_2 can decide v from C. If Q_i can decide $\overline{v}$ from C, then we are done with $z = z_i$ and φ being the empty execution. So, assume Q_1 and Q_2 are both v-univalent from C.

Since P is bivalent from C, there is a P-only execution ψ from C in which $\overline{v}$ is decided. By Proposition 1(iv), Q_1 and Q_2 are both $\overline{v}$-univalent from $C\psi$. Let ψ' be the longest prefix of ψ such that Q_1 and Q_2 are both v-univalent from C. Then $\psi' \neq \psi$. Consider the next step δ in ψ after ψ'. Since $Q_1 \cup Q_2 = P$, we may assume, without loss of generality, that δ is a step by a process in Q_1. This is illustrated in Figure 2.

Since Q_1 is v-univalent from $C\psi'$, Q_1 is v-univalent from $C\psi'\delta$. Thus, Q_2 is not v-univalent from $C\psi'\delta$, so Q_2 can decide $\overline{v}$ from $C\psi'\delta$. On the other hand, Proposition 1(iii) implies that $Q_1 \cap Q_2$ is v-univalent from $C\psi'\delta$. Thus, by Proposition 1(ii), Q_2 can decide v from $C\psi'\delta$. Therefore, the claim is true with $\varphi = \psi'\delta$ and $z = z_2$. □

3.2 Covering

The first covering argument is due to Burns and Lynch [BL93]. Since then, covering arguments have become the main tool for proving space lower bounds in shared memory systems.

The main idea is the following: Suppose there is a set of processes R that are about to write to a set of registers V. Then any process $z \notin R$ which needs the other processes to see its actions must perform a write to a register not in V. Otherwise, as observed in [BL93], the processes in R can "obliterate" the information that z writes (to registers in V) by performing their writes all at once. The next definition formally captures these notions.

Definition 2. Let C be a reachable configuration of Π. A process *covers* a register r in C if it is poised to perform a write to r in C. If every process in a set of processes R covers a register in C, then R is a set of *covering processes* in C and a *block write* by R is an execution in which each process in R performs its write (and nothing else).

Note that, if every process in R covers a different register, then the order of the writes does not matter, since the resulting configurations are indistinguishable. For technical reasons, we consider $R = \emptyset$ a valid set of covering processes, even though R covers no registers. In this case, the block write by R is the empty execution.

Given a set of covering processes R in a configuration C, it is tempting to think that any process $z \notin R$ must write to a register not covered by R before it can decide a value. If this were true, then we could inductively obtain a configuration in which n different registers are covered. Unfortunately, this is not true. For example, if the set of all processes is v-univalent from C, then z does not have to write anything. It can simply decide v and terminate. The next lemma gives us a way to guarantee that a process will write to a register that is not covered. A similar result appears in all existing space lower bounds for consensus [FHS98, Gel15, Zhu15].

LEMMA 2. *Let C be a reachable configuration of Π, let P be a set of processes, let $R \subseteq P$ be a set of covering processes in C, and let β be a block write by R. If P is bivalent from $C\beta$, then, for every $z \notin P$, every deciding $\{z\}$-only execution from C contains a write to a register not covered by R in C.*

PROOF. Let ζ be any $\{z\}$-only execution from C in which some value $v \in \{0,1\}$ is decided. By Proposition 1(iv), P is v-univalent from $C\zeta$. Since β is performed by processes in P, it follows that P is v-univalent from $C\zeta\beta$. If all writes in ζ are to registers covered by R, then $C\zeta\beta$ is indistinguishable from $C\beta$ to P. This is impossible since P is bivalent from $C\beta$ and v-univalent from $C\zeta\beta$. □

A block write by a set of covering processes $R \subseteq P$ can change P from being bivalent to being univalent. The next lemma shows that, as long as the set of remaining processes, $P - R$, is bivalent, it is possible to ensure that P remains bivalent after the block write by R. The main configurations in the proof are illustrated in Figure 3.

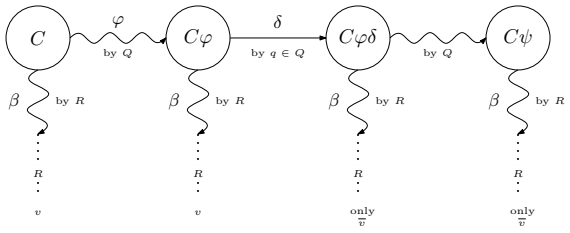

Figure 3: Diagram of configurations for Lemma 3.

LEMMA 3. *Let C be a reachable configuration of Π, let P be a set of processes, let $R \subseteq P$ be a non-empty set of covering processes in C, and let β be a block write by R. If $Q = P - R$ is bivalent from C, then there exists a Q-only execution φ from C and a process $q \in Q$ such that $R \cup \{q\}$ is bivalent from $C\varphi\beta$.*

PROOF. Since $R \neq \emptyset$, by Proposition 1(i), there exists $v \in \{0,1\}$ such that R can decide v from $C\beta$. Since Q is bivalent from C, there is a Q-only execution ψ from C which decides $\overline{v}$. By Proposition 1(iv), R is $\overline{v}$-univalent from $C\psi$. Since β is a block write by R, R is $\overline{v}$-univalent from $C\psi\beta$. Note that, since processes in R take no steps in ψ, their block write, β, is applicable at $C\varphi$, for any prefix φ of ψ. Let φ be the longest prefix of ψ such that R can decide v from $C\varphi\beta$. Then $\varphi \neq \psi$. Let δ be the next step in ψ after φ, which is by some process $q \in Q$.

If δ is a read or a write to a register covered by R, then $C\varphi\delta\beta$ is indistinguishable from $C\varphi\beta$ to R and, hence, R can decide v from $C\varphi\delta\beta$. This is impossible since R is $\overline{v}$-univalent from $C\varphi\delta\beta$. So δ must be a write to a register not covered by R and, hence, not written to in β. Thus, $C\varphi\beta\delta$ is indistinguishable from $C\varphi\delta\beta$ to R, so R is $\overline{v}$-univalent from $C\varphi\beta\delta$. Since R can decide v from $C\varphi\beta$ and R can decide $\overline{v}$ from $C\varphi\beta\delta$, $R \cup \{q\}$ is bivalent from $C\varphi\beta$. □

3.3 Main Result

We are now ready to prove the main technical lemma. We begin with a high level outline. Intuitively, this lemma says that, whenever we have a configuration C from which a set of processes P is bivalent, we can reach a nice configuration D from which a *pair* of processes in P is bivalent and the remaining processes form a set of well spread covering processes in D (i.e., every process covers a different register).

Combined with Lemma 3, this allows us to construct an infinite sequence of nice configurations $D_0, D_1, D_2, \ldots$, where D_{i+1} is reachable from D_i by an execution that contains a block write by a set of well spread covering processes in D_i. There are only finitely many registers, so by the pigeonhole principle, there are two distinct configurations D_i and D_j where the covering processes in D_i and D_j cover the same set of registers, V.

Now, suppose that we have a process $z \notin P$. Consider any solo execution by z starting from immediately before the block write to V between D_i and D_{i+1}. By Lemma 2, z writes to a register not in V. If we stop z immediately before its first write to a register not in V, then, after the block write, the processes in P can't detect that z took any steps. Thus, they can perform the same sequence of steps as they did to reach D_j. The resulting configuration is a nice bivalent configuration with a larger set of well spread covering processes (which includes z). This suggests that can we use Lemma 1 to do induction on $|P|$ to get successively larger sets of well spread covering processes.

We now proceed with the formal proof. The construction is illustrated in Figure 4.

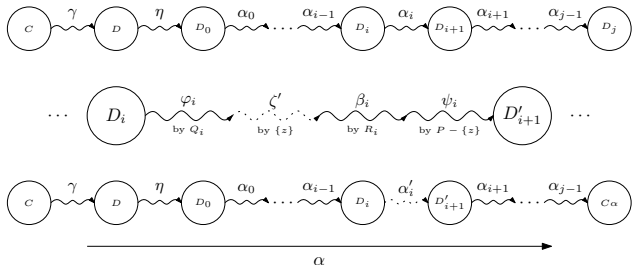

Figure 4: Diagram of configurations in Lemma 4.

LEMMA 4. *Let C be a reachable configuration of Π, and let P be a set processes with $|P| \geq 2$. If P is bivalent from C, then there exists a P-only execution α from C and a pair of processes $Q \subseteq P$ such that Q is bivalent from $C\alpha$ and every process in $P - Q$ covers a different register in $C\alpha$.*

PROOF. By induction on $|P|$. The base case is when $|P| = 2$ and the claim is satisfied with an empty execution α. Now, suppose that $|P| \geq 3$ and the claim holds for $|P| - 1$. By Lemma 1, there is a P-only execution γ from C and $z \in P$ such that $P - \{z\}$ is bivalent from $D = C\gamma$.

We construct a sequence of configurations $(D_i)_{i \geq 0}$ reachable from D by $(P - \{z\})$-only executions such that, for $i \geq 0$, there is a pair of processes $Q_i \subseteq P - \{z\}$ that is bivalent from D_i and every process in $R_i = (P - \{z\}) - Q_i$ covers a different register. Furthermore, D_{i+1} is reachable from D_i by a $(P - \{z\})$-only execution α_i that contains a block write β_i by R_i.

Constructing D_0: Since $P - \{z\}$ is bivalent from D and $|P - \{z\}| = |P| - 1$, by the induction hypothesis, there is a $(P - \{z\})$-only execution η from D and a pair of processes $Q_0 \subseteq P - \{z\}$ such that Q_0 is bivalent from $D_0 = D\eta$ and every process in $R_0 = (P - \{z\}) - Q_0$ covers a different register in D_0.

Constructing D_{i+1}: If $R_i = \emptyset$, then let $D_{i+1} = D_i$ and let α_{i+1} be empty. Otherwise, let β_i be a block write by R_i. Since $R_i \neq \emptyset$, by Lemma 3, there is a Q_i-only execution φ_i from D_i and a process $q \in Q_i$ such that $R_i \cup \{q\}$ is bivalent from $D_i\varphi_i\beta_i$. Hence, by Proposition 1(ii), $P - \{z\}$ is bivalent from $D_i\varphi_i\beta_i$. By the induction hypothesis, there is a $(P - \{z\})$-only execution ψ_i from $D_i\varphi_i\beta_i$ and a pair of processes $Q_{i+1} \subseteq P - \{z\}$ such that Q_{i+1} is bivalent from $D_{i+1} = D_i\varphi_i\beta_i\psi_i$ and every process in $R_{i+1} = (P - \{z\}) - Q_{i+1}$ covers a different register in D_{i+1}. Let $\alpha_i = \varphi_i\beta_i\psi_i$.

Since there are only finitely many registers, there exists $0 \leq i < j$ such that R_i covers the same set of registers in D_i as R_j does in D_j. We now insert steps of z so that no process in $P - \{z\}$ can detect them. Since Π is a nondeterministic solo terminating protocol, there is a $\{z\}$-only execution ζ from $D_i\varphi_i$ that decides a value $v \in \{0, 1\}$. By Lemma 2, ζ contains a write to a register not covered by R_i in D_i. Let ζ' be the longest prefix of ζ containing only writes to registers covered by R_i in D_i. It follows that, in $D_i\varphi_i\zeta'$, z is poised to write to a register not covered by R_i in D_i and, hence, R_j in D_j.

$D_i\varphi_i\zeta'\beta_i$ is indistinguishable from $D_i\varphi_i\beta_i$ to $P - \{z\}$, so the $(P - \{z\})$-only execution $\psi_i\alpha_{i+1} \cdots \alpha_{j-1}$ is applicable at $D_i\varphi_i\zeta'\beta_i$. Let $\alpha = \gamma\eta\alpha_0 \cdots \alpha_{i-1}\varphi_i\zeta'\beta_i\psi_i\alpha_{i+1} \cdots \alpha_{j-1}$. Every process in $P - \{z\}$ is in the same state in $C\alpha$ as it is in D_j. In particular, $Q_j \subseteq P - \{z\}$ is bivalent from D_j and, hence, from $C\alpha$, and every process in $R_j = (P - \{z\}) - Q_j$ covers a different register in D_j and, hence, $C\alpha$. Moreover, since z takes no steps after $D_i\varphi\zeta'$, in $C\alpha$, z covers a register not covered by R_i in D_i and, hence, R_j in D_j or $C\alpha$. Therefore, every process in $R_j \cup \{z\} = P - Q_j$ covers a different register in $C\alpha$. □

The lower bound follows immediately from Lemma 4.

THEOREM 1. *Let S be an asynchronous shared memory system with $n \geq 2$ processes. Then every nondeterministic solo terminating binary consensus protocol Π designed for S uses at least $n - 1$ registers.*

PROOF. By Proposition 2, there is an initial configuration I of Π with processes p_0 and p_1 such that $\{p_v\}$ is v-univalent from I and, hence, $\{p_0, p_1\}$ is bivalent from I. In the case when $n = 2$, if no process ever writes to a register, then p_0 can decide 0 and p_1 would not be able to tell the difference. Hence it can decide 1, which violates Agreement. Now, suppose $n \geq 3$. By Lemma 4, starting from I, it is possible to reach a configuration C' from which a pair of processes, Q, is bivalent and the remaining $n - 2$ processes, R, all cover different registers. By Lemma 3, there is a Q-only execution α from C' and a process $q \in Q$ such that $R \cup \{q\}$ is bivalent from $C\alpha\beta$, where β is the block write by R. Let $z \in Q - \{q\}$. By Lemma 2, z writes to a register not covered by R in its solo terminating execution from $C\alpha$. Hence, Π uses at least $|R| + 1 = n - 1$ registers. □

4. CONCLUSION AND FUTURE WORK

We have shown that any nondeterministic solo terminating binary consensus protocol for n processes uses at least $n-1$ registers. The best known randomized and obstruction-free consensus protocols use n registers. We conjecture that the true space complexity is n (we have proved this for $n \leq 3$ in the general case and for $n \leq 4$ in the anonymous case).

Another interesting avenue is to see if our techniques can be used to prove an $\Omega(n - k)$ space lower bound for *k-set agreement*. This problem is a generalization of the consensus problem which allows up to k different values to be decided. In particular, consensus is another name for 1-set agreement. The best existing protocols for k-set agreement use $n-k+1$ registers [BRS15].

Finally, the $\Omega(\sqrt{n})$ lower bound in [FHS98] actually holds for *historyless* base objects, such as *swap* objects. It is not clear how to modify our lower bound to work in this case. The difficulty is that, when a process performs swap, it sees the value it overwrote. Thus, it might be able to detect whether some other process has performed a swap on this object since it last accessed this object.

5. ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Faith Ellen. She was the one who sparked my interest in this problem and her energy, patience, and insights in discussing this problem and reading many drafts of this work have proven invaluable to me. I would also like to thank Rati Gelashvili, Mika Göös, and the anonymous reviewers for their helpful comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

6. REFERENCES

[AC08] Hagit Attiya and Keren Censor. Tight bounds for asynchronous randomized consensus. *J. ACM*, 55(5), 2008.

[AE14] Hagit Attiya and Faith Ellen. *Impossibility Results for Distributed Computing*. Morgan & Claypool Publishers, 2014.

[AH90] James Aspnes and Maurice Herlihy. Fast randomized consensus using shared memory. *J. Algorithms*, 11(3):441–461, 1990.

[AW96] James Aspnes and Orli Waarts. Randomized consensus in expected o(n $\log^2$ n) operations per processor. *SIAM J. Comput.*, 25(5):1024–1044, 1996.

[BL93] James E. Burns and Nancy A. Lynch. Bounds on shared memory for mutual exclusion. *Inf. Comput.*, 107(2):171–184, 1993.

[Bow11] Jack R. Bowman. Obstruction-free snapshot, obstruction-free consensus, and fetch-and-add modulo k. Technical Report TR2011-681, Dartmouth College, Computer Science, Hanover, NH, June 2011.

[BRS15] Zohir Bouzid, Michel Raynal, and Pierre Sutra. Brief Announcement: Anonymous Obstruction-free (n, k)-Set Agreement with n–k+1 Atomic Read/Write Registers. In *Proceedings of 29th International Symposium on Distributed Computing (DISC 2015)*, volume 9363 of *LNCS*, pages 668–669. Springer, 2015.

[CIL94] Benny Chor, Amos Israeli, and Ming Li. Wait-free consensus using asynchronous hardware. *SIAM J. Comput.*, 23(4):701–712, 1994.

[EFR08] Faith Ellen, Panagiota Fatourou, and Eric Ruppert. The space complexity of unbounded timestamps. *Distributed Computing*, 21(2):103–115, 2008.

[FHS98] Faith Ellen Fich, Maurice Herlihy, and Nir Shavit. On the space complexity of randomized synchronization. *J. ACM*, 45(5):843–862, 1998. A preliminary version appeared in PODC '93.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[Gel15] Rati Gelashvili. On the optimal space complexity of consensus for anonymous processes. In *Proceedings of 29th International Symposium on Distributed Computing (DISC 2015)*, volume 9363 of *LNCS*, pages 452–466. Springer, 2015.

[GHHW13] George Giakkoupis, Maryam Helmi, Lisa Higham, and Philipp Woelfel. An $O(\sqrt{n})$ space bound for obstruction-free leader election. In *Proceedings of 27th International Symposium on Distributed Computing (DISC 2013)*, volume 8205 of *LNCS*, pages 46–60. Springer, 2013.

[GHHW15] George Giakkoupis, Maryam Helmi, Lisa Higham, and Philipp Woelfel. Test-and-set in optimal space. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC 2015)*, pages 615–623, 2015.

[GR05] Rachid Guerraoui and Eric Ruppert. What can be implemented anonymously? In *Proceedings of 19th International Symposium on Distributed Computing (DISC 2005)*, volume 3724 of *LNCS*, pages 244–259. Springer, 2005.

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, 1991.

[HHPW14] Maryam Helmi, Lisa Higham, Eduardo Pacheco, and Philipp Woelfel. The space complexity of long-lived and one-shot timestamp implementations. *J. ACM*, 61(1):7:1–7:25, 2014.

[LAA87] Michael C. Loui and Hosame H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163–183, 1987.

[Zhu15] Leqi Zhu. Brief Announcement: Tight Space Bounds for Memoryless Anonymous Consensus. In *Proceedings of 29th International Symposium on Distributed Computing (DISC 2015)*, volume 9363 of *LNCS*, pages 665–666. Springer, 2015.