Theory of Computing Systems (2021) 65:687–705
https://doi.org/10.1007/s00224-020-09993-6

Space Lower Bounds for the Signal Detection Problem

Faith Ellen[1] · Rati Gelashvili[1] · Philipp Woelfel[2] · Leqi Zhu[1]

Published online: 20 July 2020

Abstract

Many shared memory algorithms have to deal with the problem of determining whether the value of a shared object has changed in between two successive accesses of that object by a process when the responses from both are the same. Motivated by this problem, we define the *signal detection problem*, which can be studied on a purely combinatorial level. Consider a system with $n + 1$ processes consisting of n readers and one signaller. The processes communicate through a shared *blackboard* that can store a value from a domain of size m. Processes are scheduled by an adversary. When scheduled, a process reads the blackboard, modifies its contents arbitrarily, and, provided it is a reader, returns a Boolean value. A reader must return *true* if the signaller has taken a step since the reader's preceding step; otherwise it must return *false*. Intuitively, in a system with n processes, signal detection should require at least n bits of shared information, i.e., $m \geq 2^n$. But a proof of this conjecture remains elusive. For the general case, we prove a lower bound of $m \geq n^2$. For restricted versions of the problem, where the processes are oblivious or where the signaller must write a fixed sequence of values, we prove a tight lower bound of $m \geq 2^n$. We also consider a version of the problem where each reader takes at most two steps. In this case, we prove that $m = n + 1$ blackboard values are necessary and sufficient.

Keywords Signal detection · ABA problem · Space complexity · Lower bounds

This article belongs to the Topical Collection: *Special Issue on Theoretical Aspects of Computer Science (2019)*
Guest Editors: Rolf Niedermeier and Christophe Paul

✉ Philipp Woelfel
woelfel@ucalgary.ca

Extended author information available on the last page of the article.

1 Introduction

1.1 The Signal Detection Problem

Consider a system consisting of $n+1$ processes, one *signaller*, s, and n *readers*, $r_1, \ldots, r_n$, that communicate through a shared blackboard. The blackboard can contain one value from a domain of size m. Processes are scheduled to take steps one at a time by an adversarial scheduler. Whenever a process takes a step, it atomically reads the blackboard and can modify its contents arbitrarily without interruption from other processes.

In the *signal detection problem*, each time a reader, r_i, has taken a step, it must return a Boolean value. If r_i has no preceding step, it can return either *true* or *false*. Otherwise, it must return *true* if and only if the signaller has taken a step since r_i's preceding step. We are concerned with how large m has to be for this problem to be solvable.

1.2 Simple Signal Detection Algorithms

For large or unbounded values of m, there are some simple solutions to the signal detection problem. For example, the blackboard could store an unbounded *signal counter* that is initially 0. Each time the signaller takes a step, it increments the counter. When a reader is scheduled, it simply memorizes the counter value, but does not change it. To detect whether a signal has occurred since its last step, a reader only needs to compare the current counter value with the one it read in its previous step. The value stored on the blackboard grows with the number of signals that have occurred, which can be unbounded.

A similar algorithm is for the signaller to increment the counter whenever it sees the value is odd and for each reader to increment the counter whenever it sees the value is even. A reader also memorizes the resulting value of the counter. It detects whether a signal has occurred since its last step by comparing this value with the value it previously memorized. If there are many consecutive steps by the signaller, the value of the counter grows more slowly than in the previous algorithm.

The following simple algorithm stores an n-bit string $(b_1, \ldots, b_n)$ on the blackboard, i.e. the domain size is $m = 2^n$. Initially, $b_1 = \cdots = b_n = 0$. Whenever the signaller takes a step, it sets all bits to 1. For each $j \in \{1, \ldots, n\}$, reader r_j resets bit b_j to 0, it returns *false* if the old value of b_j was 0, and it returns *true* if the old value of b_j was 1.

1.3 ABA Detection

Signal detection is related to the fundamental *ABA detection* problem in asynchronous shared memory systems. In such systems, a process that observes the same value A in some shared object in two successive accesses cannot tell whether the value of the object remained unchanged between them. More formally, it cannot distinguish between an execution in which the shared object did not change and an execution in which the value of the object changed from A to some other value

B and then back to A. Many shared memory algorithms have to deal with this problem.

A well-known example is the double-collect algorithm for performing an atomic *scan* of an array [1]: A process repeatedly performs a *collect* (reading all components of the array one by one) until the sequences of values read in two consecutive collects are the same. This algorithm is only correct (linearizable) if no ABAs occur, meaning that any two consecutive reads of the same array entry return the same value if and only if the value of the array entry was not changed between the two reads. This is because it can be shown that, provided no ABAs occur, the sequence returned by a scan must be the contents of the array at the end of its second last collect and the beginning of its last collect. However, in executions in which ABAs occur, this implementation might incorrectly return a sequence of values that was not the contents of the array at any point during the execution.

A standard approach to dealing with ABAs is tagging, as introduced by IBM [4], whereby a shared object gets augmented with a tag that changes with every write operation. If tags are never reused, the ABA problem can be avoided. From a theory perspective, this solution is unsatisfactory: If there is no bound on the length of executions, then unbounded sized objects are required to accommodate ever increasing tag values. Even though, in many practical scenarios, a system may never run out of tags, it is often desirable or even necessary to use an entire word for data. In such scenarios, the tag associated with a data word could be stored in a subsequent memory location and double-width atomic instructions could be used. However, these are not supported by most of today's mainstream architectures [7].

In some cases, it is possible to store the tag in an unrelated memory location [6], but this requires technically difficult algorithms and tedious correctness proofs. As a result, algorithm designers often deal with ABAs in an ad-hoc way. For example, a pair of handshaking bits between each pair of processes can be used to detect changes in the components of the array in a wait-free implementation of a snapshot object [1]. Such solutions are algorithm specific and require individual correctness proofs.

ABAs can also occur when using compare-and-swap (CAS) objects, which are provided by most existing multiprocessor systems and are much more powerful than read/write registers. Algorithms devised in theoretical research often use load-linked store-conditional (LL/SC) objects, which do not suffer from ABAs, and can easily replace CAS objects. Unfortunately, only a small number of multiprocessor systems provide LL/SC and they are weaker than the LL/SC specification used in theoretical research. Variants of LL/SC available in modern hardware restrict programmers severely in how the objects can be used [9], and "offer little or no help with preventing the ABA problem" [8].

To study the complexity of ABA detection, Aghazadeh and Woelfel [2] defined an *ABA detecting register*, which extends a read/write register with the ability to detect ABAs. It supports the operations `DWrite`(x), which changes the value of the object to x, and `DRead`(), which returns the current value of the object together with a Boolean flag. The flag is true if and only if the process has previously performed `DRead`() and, since its last preceding `DRead`(), some process performed `DWrite`(). The authors proved space lower bounds and time-space-tradeoffs for linearizable implementations of ABA detecting registers in shared memory systems with n processors that provide bounded atomic base objects, such as bounded read/write registers or bounded CAS objects. For example, if only bounded read/write registers are available as base objects, then at least $n-1$ of them are needed to obtain an obstruction-free ABA detecting register. If bounded CAS objects are also available, then any implementation using m base objects has step-complexity $\Omega(n/m)$.

All the lower bound results in [2] are specific to the base objects provided by the system, and provide no insights for systems using different sets of base objects. But we conjecture that there is a large, general lower bound for the amount of information that needs to be stored in a system for processes to detect ABAs: Intuitively, the system state needs to keep track of whether the value of the object has changed since each process last accessed the object. This requires at least n bits of information. Hence, it seems believable that detecting ABAs in any system with arbitrarily powerful base objects requires at least n bits of information to be stored either in the base object or in the hardware implementing the base objects (for example, implementing LL/SC objects). Using the reasonable assumption that a single base object can store $O(\log n)$ bits of information, this would imply that $\Omega(n/\log n)$ base objects are required for implementing a single ABA detecting object.

The signal detection problem is a restricted version of the problem of detecting ABAs in asynchronous shared memory systems, stripped down to the essentials necessary for determining the information theoretic requirements. Its definition is self-contained, and the problem can be studied without any background knowledge of shared memory systems. If n processes can detect ABAs in a standard asynchronous shared memory system with arbitrarily strong primitives, then they can also solve signal detection. A reader simply remembers the last value it read from the blackboard. When it reads the blackboard again, it returns *true* if it sees a different value or it detected an ABA; otherwise it returns *false*. Therefore, if signal detection requires that the blackboard has domain size at least m^*, then $\log_2 m^*$ is a lower bound for the number of bits needed for ABA detection.

1.4 Results

We conjecture that any solution to the signal detection problem requires the domain size, m, of the blackboard to be at least 2^n. Although we prove this conjecture when there are $n \leq 2$ readers, a proof when $n \geq 3$ has eluded us. This simply defined combinatorial problem does not seem to have a simple solution. Even a proof of a polynomial lower bound is surprisingly non-trivial. We show the following result in Section 6.

Theorem 1 *In any algorithm for the signal detection problem, the blackboard has domain size at least* $n(n+1)/2$.

To obtain better understanding, we consider several restricted versions of the signal detection problem and prove tight upper and lower bounds for them.

First, we consider the b-read-bounded version of signal detection, where no reader takes more than b steps, but the signaller can take arbitrarily many steps. In this case, the second algorithm from Section 1.2 uses a domain of size $2bn + 1$. We show how to improve this algorithm.

Theorem 2 *For $b \geq 2$, the b-read-bounded signal detection problem can be solved using a blackboard with domain size $(b-1)n + 1$.*

Thus, the b-read-bounded problem is strictly easier than the unrestricted problem when $b \leq \lceil n/2 \rceil$. For $b = 2$, we also prove that this algorithm is optimal.

Theorem 3 *In any algorithm for the 2-read-bounded signal detection problem, the blackboard has domain size at least $n + 1$.*

Next, we consider signal detection when the actions of the signaller do not depend on what steps the readers have taken. Signal detection with *fixed signals* is the special case where the signaller writes the same sequence of values to the blackboard in every execution. Note that the simple algorithm above with $m = 2^n$ uses fixed signals.

Theorem 4 *In any algorithm for the signal detection problem with fixed signals, the blackboard has domain size at least 2^n.*

Then we consider the case of *write oblivious* processes. Here each process p is equipped with a fixed function $f_p : \{0, \ldots, m-1\} \to \{0, \ldots, m-1\}$. When taking a step it replaces the blackboard contents x with $f_p(x)$. Hence, what a process writes to the blackboard is independent of the internal state of the process. However, the return value of a reader's step may depend on its internal state. In the simple algorithm above, which uses $m = 2^n$ blackboard values, processes are write oblivious. We prove that, when processes are write oblivious, no better algorithm exists.

Theorem 5 *In any algorithm for the signal detection problem with write oblivious processes, the blackboard has domain size at least 2^n.*

The signal detection problem with write oblivious processes is similar to determining the minimum size of a dictionary in a sequential system. A *dictionary* supports three operations, *insert*(x), *query*(x), and *reset*$()$, where x is a parameter chosen from a domain of size n. A call to *query*(x) returns true if there has been an *insert*(x) operation since the last *reset*$()$ operation or since the beginning of the execution, if there has been no *reset*$()$. Otherwise, it returns false. A dictionary implemented using $b(n)$ bits immediately yields a solution to the signal detection problem with oblivious processes as follows: A blackboard with $m = 2^{b(n)}$ possible values is used to store the dictionary. When a signaller takes a step, it simulates a *reset*$()$ operation on the dictionary stored on the blackboard. Similarly, when reader r_i takes a step, it simulates *query*(i) followed by *insert(i)* on the dictionary and then returns the return value of its query operation. However, an arbitrary solution to the signal detection problem does not seem to yield an implementation of a dictionary. The difficulty is that the return value of a step by a reader r_i can depend on the state of the reader and, thus, its entire past execution. In contrast, the result of a *query*(i) operation is only a function of the state of the dictionary. Hence, the n-bit information theory lower bound for implementing a dictionary cannot be used to obtain Theorem 5.

In the simple algorithm that uses $m = 2^n$ blackboard values, the response each reader returns also does not depend on its internal state, but only on the contents of the blackboard. We call such processes *response oblivious*. The same lower bound holds for any algorithm with response oblivious processes, even if the algorithm supports only 2 steps by each reader.

Theorem 6 *In any algorithm for 2-read-bounded signal detection with response oblivious processes, the blackboard has domain size at least* 2^n.

Theorem 2 implies that, for the unrestricted signal detection problem when there is only $n = 1$ reader, the domain size is at least $2 = 2^n$. In Section 7, we investigate the unrestricted signal detection problem for the case of $n = 2$ readers. First, we prove a tight lower bound of 4 for the size of the blackboard domain. Then we present an algorithm for two readers, r_1 and r_2, which uses a bounded number of blackboard values, and for which every reachable configuration C satisfies the following property: As long as only readers take steps starting in C, at most three different blackboard values are obtained. This observation indicates that a tight lower bound of $m \geq 2^n$ for n readers for the unrestricted signal detection problem may have to be fundamentally different from our lower bound proof for fixed signals. In that proof, we showed that one can reach a configuration, C, from which 2^n different blackboard values result from the 2^n schedules that are sub-sequences of $(r_1, \ldots, r_n)$. In our third simple algorithm for n readers in Section 1.2, each execution that ends with the signaller taking a step results in a configuration with this property. But our two reader algorithm indicates that a lower bound proof for the unrestricted signal detection problem cannot rely on this property.

2 Preliminaries

We consider a deterministic, asynchronous system in which $n + 1$ processes, $s, r_1, \ldots, r_n$ communicate with one another using a single shared *blackboard*. Each time a process takes a *step*, it atomically reads the blackboard, may change the value of the blackboard based on its state and the value it read, and updates its state.

A *configuration* C consists of a value, $\mathsf{v}(C)$, for the blackboard and a state for each process. An *execution* is an alternating sequence of configurations and steps, starting and ending with a configuration, such that each step can be performed in the configuration that precedes it, resulting in the configuration that follows it. Configuration C is *reachable* if there is an execution starting with an initial configuration and ending with C. For any set of processes, P, a P*-only execution* is an execution in which only processes in P take steps in the execution. A solo execution is a P-only execution in which P contains only one process, i.e., all steps in the execution are by the same process.

A *schedule* is a sequence of processes (in which the same process can occur multiple times). A P*-only schedule* is a schedule in which only processes in P appear.

For any deterministic algorithm and for any configuration C, a schedule determines a unique execution starting from C in which the processes take steps in the order specified by the schedule. If α is a finite schedule, then $C\alpha$ denotes the configuration at the end of this execution.

Two configurations, C and C', are *indistinguishable* to a set of processes, P, if $\mathsf{v}(C) = \mathsf{v}(C')$ and each process in P has the same state in C as it does in C'. If C and C' are indistinguishable to P and α is a finite P-only schedule, then the same sequence of steps is taken in the executions determined by α from C and C'. Moreover, $C\alpha$ and $C'\alpha$ are indistinguishable to P.

Given a set of readers, R, let $\mathbf{R}$ denote the schedule consisting of one occurrence of each reader in R, in order of their identifiers, and let $M(R)$ denote the set $\{r_i : i \leq j \text{ for some } r_j \in R\}$ of all readers whose identities are less than or equal to the largest identity of the readers in R. In particular, $M(\emptyset) = \emptyset$. For example, $M(\{r_1, r_4, r_8\}) = \{r_1, r_2, \ldots, r_8\}$. Notice that, for any two sets of readers R and R', either $M(R) \subseteq M(R')$ or $M(R') \subseteq M(R)$. There are $n + 1$ such sets, i.e., $|\{M(R) : R \subseteq \{r_1, \ldots, r_n\}\}| = n + 1$.

Lemma 1 *For every signal detection algorithm in which the domain size of the blackboard is finite, there is a reachable configuration D such that, for every set of readers, T, $\mathsf{v}(D\mathbf{T}s\beta_T) = \mathsf{v}(D\mathbf{T})$ for some $(M(T) \cup \{s\})$-only schedule β_T.*

Proof Consider any signal detection algorithm. Assume that, for each reachable configuration C, there is a set of readers, T, such that $\mathsf{v}(C\mathbf{T}s\beta) \neq \mathsf{v}(C\mathbf{T})$ for all $(M(T) \cup \{s\})$-only schedules β. We define an infinite sequence of reachable configurations as follows. Let C_0 be the initial configuration. Let $j \geq 1$ and suppose that C_{j-1} is a reachable configuration. Let T_j be a set of readers such that $\mathsf{v}(C_{j-1}\mathbf{T}_j s\beta) \neq \mathsf{v}(C_{j-1}\mathbf{T}_j)$ for all $(M(T_j) \cup \{s\})$-only schedules β. The existence of T_j follows from the assumption, since C_{j-1} is reachable. Let $C_j = C_{j-1}\mathbf{T}_j s$.

Since there is only a finite number of readers, there exists a set M such that $\{j \in \mathbb{Z}^+ : M(T_j) = M\}$ is infinite. Let M be the largest such set, let $J = \{j \in \mathbb{Z}^+ : M(T_j) = M\}$, and let $k^* = \min\{k \geq 1 : M(T_j) \subseteq M \text{ for all } j \geq k\}$. Note that, for all $k, \ell \in J$ such that $k^* \leq k < \ell$, the schedule $\mathbf{T}_{k+1}s\mathbf{T}_{k+2}s \cdots \mathbf{T}_\ell$ is $(M \cup \{s\})$-only. Thus, by definition of T_k, $\mathsf{v}(C_k\mathbf{T}_k) \neq \mathsf{v}(C_k\mathbf{T}_k s\mathbf{T}_{k+1}s \cdots \mathbf{T}_\ell) = \mathsf{v}(C_\ell\mathbf{T}_\ell)$. Hence, the domain size of the blackboard is infinite. □

3 Read-Bounded Signal Detection

In the b-read-bounded signal detection problem, no reader takes more than b steps, but the signaller can take arbitrarily many steps.

Consider the following algorithm that solves this restricted problem for $b = 2$ using a blackboard with domain size $m = n + 1$:

- The blackboard initially has value 0.
- Whenever s takes a step, it resets the blackboard contents to 0.
- When r_i takes its first step, it changes the blackboard contents to i if it reads 0; otherwise it leaves the blackboard unchanged. In either case, r_i locally stores

the value $v_i \neq 0$ of the blackboard immediately after its first step and returns *true*.
- When r_i takes its second step, it returns *false* if it reads v_i from the blackboard; otherwise it returns *true*. It does not change the value of the blackboard in either case.

Note that only the signaller changes the blackboard contents to 0 and readers only change the blackboard contents from 0. Thus, if the signaller does not take any steps between the two steps of reader r_i, then the value of the blackboard remains v_i during this interval and r_i returns *false*.

If the signaller does take a step between the two steps of reader r_i, then the blackboard is reset to 0. Consider the last step, S', by the signaller during this interval. If no reader takes its first step after S', but before the second step by r_i, then r_i will read 0 from the blackboard on its second step and return *true*. Otherwise, consider the first step after S' in which a reader r_j takes its first step. It will change the blackboard contents to j. Note that $j \neq v_i$, since r_j is the only reader that can change the blackboard contents to j and r_j has not previously taken a step. In this case, r_i will read j from the blackboard on its second step and return *true*.

A similar algorithm works for b-read-bounded signal detection for any larger b, but the size of the blackboard domain also increases.

Theorem 2 *For $b \geq 2$, the b-read-bounded signal detection problem can be solved using a blackboard with domain size $(b-1)n+1$.*

Proof Consider the following algorithm for a signaller and n readers:

- The blackboard domain is $\{0\} \cup \{(i, j) : 1 \leq i \leq n$ and $1 \leq j \leq b-1\}$. The blackboard initially has value 0.
- Whenever s takes a step, it resets the blackboard contents to 0.
- Each reader r_i has a local b-bounded counter c_i, which is initially 0 and is incremented each time r_i reads 0 from the blackboard. It also has a local variable v_i in which it stores a non-zero value from the blackboard domain. It initially contains $(i, 1)$.
- When r_i takes a step, it reads the blackboard. It will return *false* if the value read is the same as the value stored in v_i. Otherwise, it returns *true*. If the value read is non-zero, it stores the result in v_i and does not change the blackboard. If r_i reads 0, it increments c_i and, if $c_i < b$, it changes the blackboard to (i, c_i) and also sets v_i to the same value. When $c_i = b$, reader r_i can take no additional steps.

The counter c_i is initially 0. Before changing the blackboard to (i, c_i), reader r_i increments c_i and checks that it is less than b. Hence, the blackboard only contains values in its domain. Note that only r_i changes the blackboard to (i, j) for $1 \leq j \leq b-1$. Thus, whenever the blackboard is changed to a nonzero value, its new value is a value that it never previously had.

Since the blackboard initially contains 0 and v_i initially contains $(i, 1)$, the first time r_i takes a step, it will read a value different from v_i and will return *true*.

At the end of each step by r_i, the blackboard contains v_i. Readers only change the blackboard when it contains 0. Thus, if the signaller does not take any steps between two steps of reader r_i, then the value of the blackboard remains v_i during this interval and r_i returns *false*.

If the signaller does take a step between two consecutive steps of reader r_i, then the blackboard is reset to 0. Then either the blackboard has value 0 when it is later read during the second of these steps, or some other process changed the blackboard to have a value it never previously had. In either case, its value is not equal to v_i and r_i returns *true*.

The counter c_i is incremented each time r_i reads 0 from the blackboard. Since it is initially 0 and otherwise is not incremented, it records the number of times that r_i reads 0 from the blackboard. If r_i takes at most b steps, then $c_i < b$ at the beginning of each of its steps, so r_i does not stop prematurely. □

Note that, if $b \leq \lceil n/2 \rceil$, then $(b-1)n + 1 \leq (n-1)n/2 + 1 < n(n+1)/2$ for $n \geq 2$. Hence, the domain size used in this algorithm is strictly less than the lower bound in Theorem 1 for any algorithm solving signal detection with an unbounded number of reads.

When $b = 2$, we can show that the domain size of this algorithm cannot be any smaller. The following simple lemma is the key to our lower bound for 2-read-bounded signal detection.

Lemma 2 *Let C be a configuration and let r be a reader. If α is an $(\{r_1, \ldots, r_n\} - \{r\})$-only schedule and β is a $(\{s, r_1, \ldots, r_n\} - \{r\})$-only schedule, then, for every configuration D in the execution determined by α from $C' = Cr$ and for every configuration E in the execution determined by β from $C'\alpha s$, $\mathsf{v}(D) \neq \mathsf{v}(E)$.*

Proof Suppose not. Then there are some such configurations D and E such that $\mathsf{v}(D) = \mathsf{v}(E)$. Since r does not occur in the schedule $\alpha s \beta$, D and E are indistinguishable to r. Note that r must return *false* if it takes a step in configuration D, because s has not taken any steps since r last took a step. However, r must return *true* if it takes a step in configuration E, because s has taken a step since r last took a step. This is impossible, because D and E are indistinguishable to r. □

We can now prove Theorem 3.

Theorem 3 *In any algorithm for the 2-read-bounded signal detection problem, the blackboard has domain size at least $n + 1$.*

Proof Let C_0 be the initial configuration. For $1 \leq j \leq n$, let $C_j = C_{j-1} s r_j$ and let $C_{n+1} = C_n s$. Let α denote the empty schedule.

For $1 \leq i < n$, let β denote the schedule $r_{i+1} \cdots s r_n s$. By Lemma 2 with $C' = C_i$ and $r = r_i$, $\mathsf{v}(C_i) \neq \mathsf{v}(E)$ for all configurations E in the execution starting from $C_i s$ determined by β. In particular, $\mathsf{v}(C_i) \neq \mathsf{v}(C_j)$ for $i + 1 \leq j \leq n + 1$.

For $i = n$, let β denote the empty schedule. By Lemma 2 with $C' = C_n$ and $r = r_n$, $\mathsf{v}(C_n) \neq \mathsf{v}(C_{n+1})$.

Hence $|\{\mathsf{v}(C_1), \ldots, \mathsf{v}(C_n), \mathsf{v}(C_{n+1})\}| = n + 1$. $\square$

4 Fixed Signals

Suppose that, whenever the signaller takes a step, the resulting value of the blackboard does not depend on what its value was. In other words, for all $k \geq 1$, there exists a value v_k in the blackboard domain such that, immediately after the signaller's k'th step in any execution, the blackboard has value v_k. For example, this is the case if the signaller always resets the contents of the blackboard to a fixed value, say 0. Using Lemma 1, we can show that, under this restriction, the blackboard has domain size at least 2^n.

Theorem 4 *In any algorithm for the signal detection problem with fixed signals, the blackboard has domain size at least 2^n.*

Proof Suppose the domain size of the blackboard is finite. Then, by Lemma 1, it is possible to reach a configuration D such that, for any set of readers T, there is a $(M(T) \cup \{s\})$-only schedule β_T such that $\mathsf{v}(D\mathbf{T}s\beta_T) = \mathsf{v}(D\mathbf{T})$.

Suppose there exist two different sets of readers $R, R' \subseteq \{r_1, \ldots, r_n\}$ such that $\mathsf{v}(D\mathbf{R}) = \mathsf{v}(D\mathbf{R}')$. Without loss of generality, $\mathbf{R} = \mathbf{T}x\mathbf{X}$ and $\mathbf{R}' = \mathbf{T}\mathbf{X}'$, where $x \in R - R'$ and $\mathbf{T}$ is the longest common prefix of $\mathbf{R}$ and $\mathbf{R}'$. Note that $M(T)$ is disjoint from $\{x\} \cup X \cup X'$, since $\mathbf{R}$ and $\mathbf{R}'$ are sorted. By definition of D, there is a $(M(T) \cup \{s\})$-only schedule β_T such that $\mathsf{v}(D\mathbf{T}s\beta_T) = \mathsf{v}(D\mathbf{T})$. The signaller s occurs exactly once in each of the schedules $\mathbf{T}s$ and $\mathbf{T}xs$. Since s has fixed signals, $\mathsf{v}(D\mathbf{T}s) = \mathsf{v}(D\mathbf{T}xs)$. The configurations $D\mathbf{T}s$ and $D\mathbf{T}xs$ are indistinguishable to $M(T)$, so, by induction on the length of β, the configurations $D\mathbf{T}s\beta$ and $D\mathbf{T}xs\beta$ are indistinguishable to $M(T)$ for every prefix β of β_T. In particular, $\mathsf{v}(D\mathbf{T}xs\beta_T)= \mathsf{v}(D\mathbf{T}s\beta_T) = \mathsf{v}(D\mathbf{T})$. Since $M(T) \cup \{s, x\}$ is disjoint from X', configurations $D\mathbf{T}xs\beta_T$ and $D\mathbf{T}$ are indistinguishable to the set of readers X'. Thus $\mathsf{v}(D\mathbf{T}xs\beta_T\mathbf{X}') = \mathsf{v}(D\mathbf{T}\mathbf{X}') = \mathsf{v}(D\mathbf{R}') = \mathsf{v}(D\mathbf{R}) = \mathsf{v}(D\mathbf{T}x\mathbf{X})$. Since $x \notin M(T) \cup X \cup X' \cup \{s\}$, it follows that $D\mathbf{T}xs\beta_T\mathbf{X}'$ and $D\mathbf{T}x\mathbf{X}$ are indistinguishable to x. Note that x must return *false* if it takes a step in configuration $D\mathbf{T}x\mathbf{X}$, because s has not taken any steps since x last took a step. However, x must return *true* if it takes a step in configuration $D\mathbf{T}xs\beta_T\mathbf{X}'$, because s has taken a step since x last took a step. This is impossible, because these two configurations are indistinguishable to x.

Hence, $\mathsf{v}(D\mathbf{R}) \neq \mathsf{v}(D\mathbf{R}')$ for all different sets of readers R and R'. It follows that $|\{\mathsf{v}(D\mathbf{R}) : R \subseteq \{r_1, \ldots, r_n\}\}| = 2^n$. $\square$

5 Oblivious Processes

In this section, we consider processes whose actions are oblivious to their states, either when writing to the blackboard or responding to a read.

Recall that a process is write oblivious if what it writes to the blackboard in a step only depends on the value of the blackboard at the beginning of that step. We prove that the simple algorithm in Section 1.2 with domain size 2^n is optimal if both the signaller and the readers are write oblivious.

Theorem 5 *In any algorithm for the signal detection problem with write oblivious processes, the blackboard has domain size at least* 2^n.

Proof Suppose the domain size of the blackboard is finite. For any (possibly empty) set of readers R and any positive integer i, consider the schedule $(s\mathbf{R})^i$ which consists of $s\mathbf{R}$ repeated i times. Because the domain size of the blackboard is finite, there exist $0 < i < j$ such that $\mathsf{v}(C_0(s\mathbf{R})^i) = \mathsf{v}(C_0(s\mathbf{R})^j)$, where C_0 is the initial configuration.

Let $R' \neq R$ be a different set of readers and let $0 < i' < j'$ be such that $\mathsf{v}(C_0(s\mathbf{R}')^{i'}) = \mathsf{v}(C_0(s\mathbf{R}')^{j'})$. Without loss of generality, suppose there is a reader $r_k \in R' - R$.

To obtain a contradiction, assume that $\mathsf{v}(C_0(s\mathbf{R})^i) = \mathsf{v}(C_0(s\mathbf{R}')^{i'})$. Since processes are write oblivious, it follows that $\mathsf{v}(C_0(s\mathbf{R}')^{i'}(s\mathbf{R})^{j-i}) = \mathsf{v}(C_0(s\mathbf{R})^i(s\mathbf{R})^{j-i}) = \mathsf{v}(C_0(s\mathbf{R})^j) = \mathsf{v}(C_0(s\mathbf{R})^i) = \mathsf{v}(C_0(s\mathbf{R}')^{i'})$. Since r_k takes no steps in $(s\mathbf{R})^{j-i}$, configurations $C_0(s\mathbf{R}')^{i'}(s\mathbf{R})^{j-i}$ and $C_0(s\mathbf{R}')^{i'}$ are indistinguishable to r_k. The signaller has taken a step after r_k's last step in the execution determined by $(s\mathbf{R}')^{i'}(s\mathbf{R})^{j-i}$ starting from C_0, so r_k must return *true* if it takes a step in configuration $C_0(s\mathbf{R}')^{i'}(s\mathbf{R})^{j-i}$. However, the signaller has not taken a step after r_k's last step in the execution determined by $(s\mathbf{R}')^{i'}$ starting from C_0, so r_k must return *false* if it takes a step in configuration $C_0(s\mathbf{R}')^{i'}$. This is impossible, so $\mathsf{v}(C_0(s\mathbf{R})^i) \neq \mathsf{v}(C_0(s\mathbf{R}')^{i'})$.

Since this holds for any two different sets of readers R and R', it follows that the blackboard domain has size at least 2^n. □

We now consider response oblivious readers, whose responses only depend on the contents of the blackboard, but not their current state. However, processes can modify the blackboard based on both the blackboard contents and their current state. Even for the 2-read-bounded signal detection problem, at least 2^n different blackboard values are necessary when readers are response oblivious.

Theorem 6 *In any algorithm for 2-read-bounded signal detection with response oblivious processes, the blackboard has domain size at least* 2^n.

Proof Consider a set of readers R and let $Q = \{r_1, \ldots, r_n\} - R$. Note that, in the execution from the initial configuration C_0 determined by schedule $\mathbf{Q}s\mathbf{R}$, each reader takes exactly one step.

Let $R' \neq R$ be a different set of readers and let $Q' = \{r_1, \ldots, r_n\} - R'$. Without loss of generality, suppose there is a reader $r_k \in R' - R$. In the execution from C_0 determined by schedule $\mathbf{Q}'s\mathbf{R}'$, reader r_k takes a step after the signaller's last step, so r_k must return *false* if it takes a step in configuration $C_0\mathbf{Q}'s\mathbf{R}'$. However, in the execution from C_0 determined by schedule $\mathbf{Q}s\mathbf{R}$, reader r_k does not take a step after the signaller's last step, so r_k must return *true* if it takes a step in configuration $C_0\mathbf{Q}s\mathbf{R}$. By response obliviousness, this implies that $\mathsf{v}(C_0\mathbf{Q}s\mathbf{R}) \neq \mathsf{v}(C_0\mathbf{Q}'s\mathbf{R}')$.

Since this holds for any two different sets of readers R and R', the blackboard domain has size at least 2^n. □

6 The General Setting

Let $\mathcal{M} = \{M(R) : R \subseteq \{r_1, \ldots, r_n\}\}$. Recall that $|\mathcal{M}| = n + 1$. For any schedule α, let $M(\alpha)$ denote $M(R)$, where R is the set of readers that take steps in α.

Lemma 3 *If the blackboard can only store a finite number of different values, then, from any configuration, it is possible to reach a configuration D such that, for any schedules α and β, there exists an $(M(\alpha) \cup M(\beta) \cup \{s\})$-only schedule γ such that* $\mathsf{v}(D\alpha\gamma) = \mathsf{v}(D\beta)$.

Proof Let C_0 be an arbitrary configuration. Suppose that, for each configuration C reachable from C_0, there are two schedules, α and β, such that, for each $(M(\alpha) \cup M(\beta) \cup \{s\})$-only schedule γ , $\mathsf{v}(C\alpha\gamma) \neq \mathsf{v}(C\beta)$.

We inductively define an infinite sequence of configurations C_j, for $j \geq 0$, such that C_{j+1} is reachable from C_j. Given C_j, which is reachable from C_0, there exist two schedules, α_{j+1} and β_{j+1}, such that $\mathsf{v}(C_j\alpha_{j+1}\gamma) \neq \mathsf{v}(C_j\beta_{j+1})$ for all $(M(\alpha_{j+1}) \cup M(\beta_{j+1}) \cup \{s\})$-only schedules γ. Let $C_{j+1} = C_j\alpha_{j+1}$.

For $j \geq 1$, let $M_j = M(\alpha_j) \cup M(\beta_j) \in \mathcal{M}$. Since $\mathcal{M}$ is finite, there exists at least one set that is equal to M_j for an infinite number of j's. Let M' denote the largest such set and let $J = \{j \geq 1 : M_j = M'\}$ be the set of indices of the occurrences of M'. Let $k^* = \min\{k \geq 1 : M_j \subseteq M' \text{ for all } j \geq k\}$ be the first index after which no set larger than M' occurs. Note that, if $k^* \leq k < \ell$, then $\gamma = \alpha_{k+1} \cdots \alpha_{\ell-1}\beta_\ell$ is an $(M' \cup \{s\})$-only schedule. Hence, if $k, \ell \in J$, then $\mathsf{v}(C_{k-1}\beta_k) \neq \mathsf{v}(C_{k-1}\alpha_k\gamma) = \mathsf{v}(C_{\ell-1}\beta_\ell)$. Thus $\{\mathsf{v}(C_{k-1}\beta_k) : k \geq k^* \text{ and } k \in J\}$ is an infinite set of values that can appear on the blackboard. □

For $0 \leq i < j \leq n$, let $\delta(i, j)$ denote the schedule $r_1sr_2s \ldots r_isr_{i+1}r_{i+2} \ldots r_j$. For example, $\delta(0, 3) = r_1r_2r_3$ and $\delta(2, 5) = r_1sr_2sr_3r_4r_5$. Note that $\delta(i, j)$ contains i occurrences of s.

Lemma 4 *Let D be a reachable configuration such that, for any schedules α and β, there exists an $(M(\alpha) \cup M(\beta) \cup \{s\})$-only schedule γ such that* $\mathsf{v}(D\alpha\gamma) = \mathsf{v}(D\beta)$. *If $0 \leq i < j \leq n$, $0 \leq i' < j' \leq n$, and either $i \neq i'$ or $j \neq j'$, then* $\mathsf{v}(D\delta(i, j)) \neq \mathsf{v}(D\delta(i', j'))$.

Proof First consider the case when $i \neq i'$. Without loss of generality, suppose that $i < i'$. The state of reader r_{i+1} is the same in configurations $D\delta(i, j)$ and $D\delta(i', j')$. In configuration $D\delta(i, j)$, if r_{i+1} takes a step, it must return *false*, because s has not taken any steps since r_{i+1} last took a step. In configuration $D\delta(i', j')$, if r_{i+1} takes a step, it must return *true*, because s has taken $i' - i$ steps since r_{i+1} last took a step. If $\mathsf{v}(D\delta(i, j)) = \mathsf{v}(D\delta(i', j'))$, then configurations $D\delta(i, j)$ and $D\delta(i', j')$ are indistinguishable to r_{i+1}, which is impossible. Thus $\mathsf{v}(D\delta(i, j)) \neq \mathsf{v}(D\delta(i', j'))$.

Now consider the case when $i = i'$ and $j \neq j'$. Without loss of generality, suppose that $j < j'$, so $\delta(i', j') = \delta(i, j)r_{j+1} \cdots r_{j'}$. Let $\alpha = \delta(i, j)s$ and $\beta = \delta(i, j)$. By assumption, there exists an $\{r_1, \dots, r_j, s\}$-only schedule γ such that $\mathsf{v}(D\delta(i, j)s\gamma) = \mathsf{v}(D\delta(i, j))$. To obtain a contradiction, suppose that $\mathsf{v}(D\delta(i', j')) = \mathsf{v}(D\delta(i, j))$. Configurations $D\delta(i', j')$ and $D\delta(i, j)$ are indistinguishable to $r_1, \dots, r_j$, and s, since the signaller and these readers take no steps in $r_{j+1} \cdots r_{j'}$. Then $\mathsf{v}(D\delta(i', j')s\gamma) = \mathsf{v}(D\delta(i, j)s\gamma) = \mathsf{v}(D\delta(i, j)) = \mathsf{v}(D\delta(i', j'))$. Since r_{j+1} does not appear in $s\gamma$, configurations $D\delta(i', j')s\gamma$ and $D\delta(i', j')$ are indistinguishable to r_{j+1}. Note that r_{j+1} must return *false* if it takes a step in configuration $D\delta(i', j')$, because s has not taken any steps since r_{j+1} last took a step. However, r_{j+1} must return *true* if it takes a step in configuration $D\delta(i', j')s\gamma$, because s has taken a step since r_{j+1} last took a step. This is impossible, because $D\delta(i', j')s\gamma$ and $D\delta(i', j')$ are indistinguishable to r_{j+1}. □

Using this lemma, we obtain Theorem 1.

Theorem 1 *In any algorithm for the signal detection problem, the blackboard has domain size at least $n(n + 1)/2$.*

Proof Consider any algorithm for signal detection in which the blackboard stores a finite number of different values. By Lemma 3, there is a reachable configuration D such that, for any schedules α and β, there exists an $(M(\alpha) \cup M(\beta) \cup \{s\})$-only schedule γ such that $\mathsf{v}(D\alpha\gamma) = \mathsf{v}(D\beta)$. By Lemma 4, for all different choices of $0 \leq i < j \leq n$, the value of the blackboard in configuration $D\delta(i, j)$ is different. There are $n(n + 1)/2 \in \Omega(n^2)$ such choices. □

7 Two Readers

In this section, we examine the special case when there are exactly two readers. The simple signal detection algorithm presented in Section 1.2 with one bit for each reader has a blackboard domain of size $2^2 = 4$. First, we prove that this is optimal.

To prove our conjecture that 2^n blackboard values are necessary for n readers, one approach would be to show the existence of a configuration from which P-only executions lead to different blackboard configurations for different subsets P of the readers. When $n = 2$, we show that such a configuration does not always exist. In particular, in Section 7.2, we present an algorithm for two readers, r_1 and r_2, with the property that, from every reachable configuration C, any reader-only execution leads to at most three different blackboard values. Thus, in order to show that four different blackboard values can be reached from some reachable configuration C, the signaller must also take steps, as is the case in our lower bound in Section 7.1. We conjecture that there are similar counterexample algorithms for $n > 2$.

7.1 A Tight Lower Bound

We begin with a straightforward observation about when two configurations have the same blackboard value.

Observation 7 For any configuration C and any schedule α such that $\mathsf{v}(C) = \mathsf{v}(C\alpha)$, if β is a schedule consisting of processes that don't take steps in α, then $\mathsf{v}(C\beta) = \mathsf{v}(C\alpha\beta)$.

The proof follows from the fact that configurations C and $C\alpha$ are indistinguishable to the processes that take steps in β.

In some situations, it is also easy to prove that two configurations have different blackboard values.

Lemma 5 *If C is a reachable configuration, ρ is a schedule that contains reader r but not the signaller s, σ is a schedule that contains s but not r, and γ is a schedule that contains neither s nor r, then $\mathsf{v}(C\rho\sigma) \neq \mathsf{v}(C\rho\gamma)$.*

Proof If reader r takes a step in configuration $C\rho\sigma$, it must return *true*, because r's previous step occurred during ρ and s took a step during σ. However, if r takes a step in configuration $C\rho\gamma$, it must return *false*, because s takes no steps during $\rho\gamma$.

Since r takes no steps in σ or γ, the state of r in configurations $C\rho\sigma$ and $C\rho\gamma$ must be the same. Now suppose that $\mathsf{v}(C\rho\sigma) = \mathsf{v}(C\rho\gamma)$. Then configurations $C\rho\sigma$ and $C\rho\gamma$ are indistinguishable to r, so it must return the same value if it takes a step in $C\rho\sigma$ and $C\rho\gamma$. This is a contradiction. $\square$

Corollary 1 *If C is a reachable configuration, ρ is a schedule that contains reader r but not the signaller s, and σ is a schedule that contains s but not r, then $\mathsf{v}(C\rho\sigma) \neq \mathsf{v}(C\rho)$.*

These results will be used repeatedly to obtain a tight lower bound when there are two readers.

Theorem 8 *Any algorithm for signal detection among $n = 2$ readers requires blackboard domain size at least 4.*

Proof Consider any algorithm, let C_0 be its initial configuration, and let $D = C_0 r_1 r_2$. Since the blackboard domain size is finite, there are two integers $k, \ell \geq 1$, such that $\mathsf{v}(Ds^k) = \mathsf{v}(Ds^{k+\ell})$. We first show that $\mathsf{v}(Ds^k r_1) \neq \mathsf{v}(Ds^k) \neq \mathsf{v}(Ds^k r_2)$. To obtain a contradiction, suppose that $\mathsf{v}(Ds^k) = \mathsf{v}(Ds^k r_i)$ for some $i \in \{1, 2\}$. Then $\mathsf{v}(Ds^{k+\ell}) = \mathsf{v}(Ds^k r_i s^\ell)$ by Observation 7 with $C = Ds^k$, $\alpha = r_i$ and $\beta = s^\ell$. Hence $\mathsf{v}(Ds^k r_i) = \mathsf{v}(Ds^k) = \mathsf{v}(Ds^{k+\ell}) = \mathsf{v}(Ds^k r_i s^\ell)$. But $\mathsf{v}(Ds^k r_i s^\ell) \neq \mathsf{v}(Ds^k r_i)$ by Corollary 1 with $C = Ds^k$, $r = r_i$, $\rho = r_i$, and $\sigma = s^\ell$. This is a contradiction. Therefore $\mathsf{v}(Ds^k) \neq \mathsf{v}(Ds^k r_1), \mathsf{v}(Ds^k r_2)$.

Applying Corollary 1 with $C = C_0$, $r = r_2$, $\rho = r_1 r_2$, and $\sigma = s^k r_1$ gives $\mathsf{v}(Ds^k r_1) = \mathsf{v}(C_0 r_1 r_2 s^k r_1) \neq \mathsf{v}(C_0 r_1 r_2) = \mathsf{v}(D)$. Similarly, $\mathsf{v}(Ds^k r_2) \neq \mathsf{v}(D)$. Applying Corollary 1 with $C = C_0$, $r = r_1$, $\rho = r_1 r_2$, and $\sigma = s^k$, gives $\mathsf{v}(Ds^k) = \mathsf{v}(C_0 r_1 r_2 s^k) \neq \mathsf{v}(C_0 r_1 r_2) = \mathsf{v}(D)$. If $\mathsf{v}(Ds^k r_1) \neq \mathsf{v}(Ds^k r_2)$, then $\mathsf{v}(D)$, $\mathsf{v}(Ds^k)$, $\mathsf{v}(Ds^k r_1)$, and $\mathsf{v}(Ds^k r_2)$ are all distinct, so suppose that $\mathsf{v}(Ds^k r_1) = \mathsf{v}(Ds^k r_2)$.

Applying Corollary 1 with $C = C_0$, $r = r_2$, $\rho = r_1 r_2$, and $\sigma = s^k r_1 s$ gives $\mathsf{v}(Ds^k r_1 s) = \mathsf{v}(C_0 r_1 r_2 s^k r_1 s) \neq \mathsf{v}(C_0 r_1 r_2) = \mathsf{v}(D)$. Applying Corollary 1 with $C =$

Ds^k, $r = r_1$, $\rho = r_1$, and $\sigma = s$ gives $\mathsf{v}(Ds^k r_1 s) \neq \mathsf{v}(Ds^k r_1)$. If $\mathsf{v}(Ds^k r_1 s) \neq \mathsf{v}(Ds^k)$, then $\mathsf{v}(D)$, $\mathsf{v}(Ds^k)$, $\mathsf{v}(Ds^k r_1)$, and $\mathsf{v}(Ds^k r_1 s)$ are all distinct, so suppose that $\mathsf{v}(Ds^k r_1 s) = \mathsf{v}(Ds^k)$.

By Observation 7 with $C = Ds^k$, $\alpha = r_1 s$ and $\beta = r_2$, it follows that $\mathsf{v}(Ds^k r_1 s r_2) = \mathsf{v}(Ds^k r_2)$, so $\mathsf{v}(Ds^k r_1 s r_2) = \mathsf{v}(Ds^k r_1)$. Applying Corollary 1 with $C = Ds^k$, $r = r_1$, $\rho = r_1$, and $\sigma = sr_2$ gives $\mathsf{v}(Ds^k r_1 s r_2) \neq \mathsf{v}(Ds^k r_1)$. This is a contradiction. Therefore, either $\mathsf{v}(Ds^k r_1) \neq \mathsf{v}(Ds^k r_2)$ or $\mathsf{v}(Ds^k r_1 s) \neq \mathsf{v}(Ds^k)$ and, hence, the domain size is at least 4. $\square$

7.2 An Algorithm for Two Readers Based on a Bounded Sequential Timestamp System

A *bounded sequential timestamp system* for a set of processes consists of a finite set of labels, L, and a *dominance* relation $\prec$ between every two different labels, i.e. if ℓ_1 and ℓ_2 are two different labels then exactly one of $\ell_1 \prec \ell_2$ and $\ell_2 \prec \ell_1$ is true. At every configuration, each process has a different label. There is one atomic operation, getTimestamp(), which may change the label of the process that called it and which ensures that its label dominates the labels of all other processes. A solution to the signal detection problem can be obtained from a bounded sequential timestamp system by using the blackboard to store the timestamps of the signaller and each reader. When either a reader or the signaller takes a step, it performs getTimestamp() and updates its timestamp stored on the blackboard, if it has changed. A reader returns *true* if and only if its timestamp was smaller (with respect to $\prec$) than the timestamp of the signaller before it was updated. The domain size of the blackboard is $|L|^{n+1}$ in this algorithm, where n is the number of readers. Note that this algorithm provides a lot more information than is required by the signal detection problem. Specifically, each reader can also determine which of the other readers have taken steps since it last took a step.

Israeli and Li [5] gave a construction of a bounded sequential timestamp system for $n + 1$ processes with a label set of size 3^n and proved that, for any $\epsilon > 0$, there exists a bounded sequential timestamp system for $n + 1$ processes whose label set has size $(2 + \epsilon)^{n+1}$, for n sufficiently large. They also proved a lower bound of $2^{n+1} - 1$ for the size of the label set of any bounded sequential timestamp system for $n + 1$ processes.

The algorithm below is based on the bounded timestamp system of Israeli and Li for three processes with $3^2 = 9$ labels, but it stores only two labels on the blackboard: the signaller's label and the larger (with respect to $\prec$) of the two readers' labels. This results in a blackboard with domain size 81, which is much larger than the number of labels used in the optimal algorithm. However, this algorithm has the property that, starting from any reachable configuration, at most three different blackboard configurations can be reached by executions in which only readers take steps.

Let $B = \{0, 1, 2\}$. The set of labels is $B^2 = B \times B$. We define the dominance relation $\prec$ as follows: For $(a_1, a_0), (b_1, b_0) \in B^2$, $(a_1, a_0) \prec (b_1, b_0)$ if and only if

1. $b_1 \equiv a_1 + 1 \pmod 3$ or
2. $b_1 = a_1$ and $b_0 \equiv a_0 + 1 \pmod 3$.

Note that, for any two distinct labels $a, b \in B^2$, either $a \prec b$ or $b \prec a$. Israeli and Li view the set of all labels with the same first component as being the labels of a separate copy of a bounded sequential timestamp for 2 processes (which uses labels in B).

The algorithm maintains the following *label invariants*: In each reachable configuration, each process $p \in \{s, r_0, r_1\}$ has a different label $\ell(p) = (\ell_1(p), \ell_0(p)) \in B^2$. In addition, each reader has a label from a different copy (i.e. $\ell_1(r_0) \neq \ell_1(r_1)$) and the signaller has a label from one of these copies (i.e. $\ell_1(s) \in \{\ell_1(r_0), \ell_1(r_1)\}$). In the initial configuration,

$$\ell(r_0) = (0, 0),\ \ell(r_1) = (1, 0),\ \text{and}\ \ell(s) = (1, 1),$$

so the label invariants are satisfied.

In each reachable configuration, the readers have different labels, so one of the reader's labels dominates the other reader's label. We call the reader with this label the *dominating reader*.

$$\begin{aligned}&\text{The blackboard stores the pair } (\ell(s), \ell(r_d)),\\ &\text{where } d \in \{0, 1\} \text{ and } \ell(r_{1-d}) \prec \ell(r_d).\end{aligned} \tag{B}$$

In other words, the signaller's label as well as the dominating reader's label are stored on the blackboard. Thus, initially, the blackboard contains ((1,1),(1,0)).

When a process takes a step, it applies the rules described below, which might change its label. We prove that the label invariants remain satisfied. If its label is changed, the process updates the blackboard so that (B) remains true. We will show that, in addition,

$$\begin{aligned}&\text{immediately after each step by } s,\ \text{its label } \ell(s) \text{ dominates both } \ell(r_0) \text{ and } \ell(r_1) \text{ and}\\ &\text{immediately after each step by a reader } r_i,\ \text{its label } \ell(r_i) \text{ dominates } \ell(s).\end{aligned} \tag{S}$$

Therefore, a reader can easily provide the proper response when it takes a step: If a reader's label dominates $\ell(s)$ at the beginning of its step, then the reader returns *false*. Otherwise it returns *true*.

Consider any configuration that satisfies (B) and the label invariants. Let $d \in \{0, 1\}$ be such that $\ell(r_{1-d}) \prec \ell(r_d)$. Since $\ell_1(r_{1-d}) \neq \ell_1(r_d)$, it follows that $\ell_1(r_d) \equiv \ell_1(r_{1-d}) + 1 \pmod 3$.

The Signaller's Step. When the signaller s takes a step, it compares its label with $\ell(r_d)$, which is stored on the blackboard. If $\ell(r_d) \prec \ell(s)$, then s retains its old label and does not update the blackboard, so (B) and the label invariants remain satisfied. In this case, $\ell_1(s) \neq \ell_1(r_{1-d})$, so, by the label invariants, $\ell_1(s) = \ell_1(r_d)$ and, hence, $\ell(r_{1-d}) \prec \ell(s)$. Thus, (S) is true.

Now consider the case when $\ell(s) \prec \ell(r_d)$. The new label of s is

$$\ell'(s) = (\ell_1(r_d), (\ell_0(r_d) + 1) \bmod 3).$$

By definition, $\ell(r_d) \prec \ell'(s)$. Since $\ell'_1(s) = \ell_1(r_d) \equiv \ell_1(r_{1-d}) + 1 \pmod 3$, we have $\ell(r_{1-d}) \prec \ell'(s)$. Thus (S) is satisfied and the labels of all three processes are different. By definition, $\ell'_1(s) = \ell_1(r_d) \in \{\ell_1(r_0), \ell_1(r_1)\}$. Since the labels of r_0 and r_1 do not change, $\ell_1(r_0) \neq \ell_1(r_1)$. Hence, the label invariants remain satisfied.

After determining its new label $\ell'(s)$, the signaller updates the first component of the pair stored on the blackboard with $\ell'(s)$, so (B) remains satisfied.

The Reader's Step. When reader r_i, $i \in \{0, 1\}$, takes a step, it compares its label with $\ell(s)$, which is stored on the blackboard. If $\ell(s) \prec \ell(r_i)$, then r_i does not change its label and (S) is true. In this case, none of the label invariants are affected, the reader does not change the blackboard, and (B) remains satisfied.

So suppose that $\ell(r_i) \prec \ell(s)$. Then r_i's new label is

$$\ell'(r_i) = \begin{cases} (\ell_1(s), (\ell_0(s)+1) \bmod 3) & \text{if } \ell_1(r_i) = \ell_1(s) \\ ((\ell_1(s)+1) \bmod 3, 0) & \text{otherwise.} \end{cases}$$

In both cases, $\ell(s) \prec \ell'(r_i)$, so (S) is satisfied and $\ell'(r_i) \neq \ell(s)$. If $\ell_1(r_i) = \ell_1(s)$, then, by construction, $\ell'_1(r_i) = \ell_1(s) = \ell_1(r_i)$. By the label invariants, $\ell_1(r_i) \neq \ell_1(r_{1-i})$, so $\ell'_1(r_i) \neq \ell_1(r_{1-i})$. If $\ell_1(r_i) \neq \ell_1(s)$, then, since $\ell_1(s) \in \{\ell_1(r_0), \ell_1(r_1)\}$, we have $\ell_1(s) = \ell_1(r_{1-i})$. By construction, $\ell'_1(r_i) \neq \ell_1(s)$, so $\ell'_1(r_i) \neq \ell_1(r_{1-i})$. In both cases, $\ell'(r_i) \neq \ell(r_{1-i})$. Since $\ell(r_{1-i}) \neq \ell(s)$, the labels of all three processes are different. Hence, the label invariants remain satisfied.

If $\ell_1(r_i) = \ell_1(s)$ and $d = 1-i$ (i.e., the second label on the blackboard is not equal to $\ell(r_i)$), then r_i does not change the blackboard. In this case, by (B), $\ell(r_i) \prec \ell(r_{1-i})$ and, by the label invariants, $\ell_1(r_i) \neq \ell_1(r_{1-i})$. Therefore, it follows that $\ell_1(r_{1-i}) \equiv (\ell_1(r_i)+1) \pmod 3$. By construction, $\ell'_1(r_i) = \ell_1(s)$, so $\ell_1(r_{1-i}) \equiv (\ell'_1(r_i)+1) \pmod 3$ and, hence, $\ell'(r_i) \prec \ell(r_{1-i})$. Thus (B) remains satisfied.

If $\ell_1(r_i) = \ell_1(s)$ and $d = i$, then r_i changes the second component of the blackboard to $\ell'(r_i) = (\ell_1(s), (\ell_0(s)+1) \bmod 3)$. Since $\ell(r_{1-i}) \prec \ell(r_i)$ and, by the label invariants, $\ell_1(r_{1-i}) \neq \ell_1(r_i)$, it follows that $\ell'_1(r_i) \equiv \ell_1(r_{1-i}) + 1 \pmod 3$. Hence $\ell(r_{1-i}) \prec \ell'(r_i)$ and (B) remains satisfied.

The last case is when $\ell_1(r_i) \neq \ell_1(s)$. Then r_i changes the second component of the blackboard to $\ell'(r_i) = ((\ell_1(s)+1) \bmod 3, 0)$. By the label invariants, $\ell_1(s) = \ell_1(r_{1-i})$, so $\ell'_1(r_i) = (\ell_1(r_{1-i})+1) \bmod 3$. Hence $\ell(r_{1-i}) \prec \ell'(r_i)$ and (B) remains satisfied.

Lemma 6 *For any reachable configuration C,*

$$\left|\left\{\mathsf{v}(C\alpha) : \alpha \in \{r_0, r_1\}^*\right\}\right| \leq 3.$$

Proof Without loss of generality, we may assume that either C is the initial configuration or the last step in the execution leading to C was taken by the signaller. Let $\ell^*(p) = \left(\ell^*_1(p), \ell^*_0(p)\right)$ denote the label of process $p \in \{s, r_0, r_1\}$ in configuration C. By (S) and the definition of the initial configuration, the label of the signaller, $\ell^*(s)$, dominates the labels, $\ell^*(r_0)$ and $\ell^*(r_1)$, of the readers. By the label invariants, there is an index $d \in \{0, 1\}$ such that $\ell^*_1(s) = \ell^*_1(r_d) \neq \ell^*_1(r_{1-d})$. Since $\ell^*(r_{1-d})$ is dominated by $\ell^*(s)$, it is also dominated by $\ell^*(r_d)$. Thus, by (B), $\mathsf{v}(C) = (\ell^*(s), \ell^*(r_d))$.

The signaller takes no steps in α, so its label remains unchanged. If r_d takes at least one step in α, its label changes to $\ell'(r_d) = \left(\ell^*_1(s), (\ell^*_0(s)+1) \bmod 3\right)$ and, if r_{1-d} takes at least one step in α, its label changes to $\ell'(r_{1-d}) = \left((\ell^*_1(s)+1) \bmod 3, 0\right)$.

By (S), after a reader has taken a step, its new label dominates $\ell^*(s)$, so its label does not change when it takes additional steps in α.

When a reader takes a step, either it does not change the blackboard, or it replaces the second component of the blackboard with its new label. Thus, for all $\alpha \in \{r_0, r_1\}^*$,

$$\mathsf{v}(C\alpha) = (\ell^*(s), \ell), \text{ where } \ell \in \left\{\ell^*(r_d), \ell'(r_d), \ell'(r_{1-d})\right\}.$$

Hence, $|\{\mathsf{v}(C\alpha) : \alpha \in \{r_0, r_1\}^*\}| \leq 3$. □

8 Discussion

This paper introduces the signal detection problem, gives some simple algorithms for solving it, proves that domain size at least $n(n+1)/2$ is necessary, and proves that domain size 2^n is necessary and sufficient for three restricted versions of the problem: when the processes are write oblivious, when the readers are response oblivious, or when the signaller writes a fixed sequence of values to the blackboard. We conjecture that domain size must be at least 2^n for the unrestricted version of the problem. It would also be interesting to prove this bound for two other restricted versions of the problem: when only the signaller is write oblivious or only the readers are write oblivious.

We also consider another restricted version of the problem, where each reader takes no more than $b \geq 2$ steps. We give an algorithm with domain size $(b-1)n+1$ and prove this is optimal when $b = 2$. It remains open whether it is optimal when $b > 2$. If the signaller takes no more that B steps, then one of our simple algorithms uses a domain of size $B+1$. It is unclear whether it is possible to have an algorithm for this restricted version that has a smaller domain.

Acknowledgements Support is gratefully acknowledged from the Natural Science and Engineering Research Council of Canada (NSERC) under Discovery Grant RGPIN-2015-05080, Discovery Grant RGPIN-2019-04852, the Canada Research Chairs program, and a Postgraduate Scholarship, and from a University of Toronto Faculty of Arts and Science Postdoctoral Fellowship.

A preliminary version of this paper appeared in [3].

References

1. Afek, Y., Attiya, H., Dolev, D., Gafni, E., Merritt, M., Shavit, N.: Atomic snapshots of shared memory. J. ACM **40**(4), 873–890 (1993)

2. Aghazadeh, Z., Woelfel, P.: On the time and space complexity of ABA prevention and detection. In: Proceedings of the 34th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), pp. 193–202 (2015)
3. Ellen, F., Gelashvili, R., Woelfel, P., Zhu, L.: Space lower bounds for the signal detection problem. In: Proceedings of the 36th International Symposium on Theoretical Aspects of Computer Science, (STACS), pp. 26:1–26:13 (2019)
4. IBM system/370 extended architecture, principles of operation. Tech. rep. (1983). Publication No. SA22-7085
5. Israeli, A., Li, M.: Bounded time-stamps. Distrib. Comput. **6**(4), 205–209 (1993)
6. Jayanti, P., Petrovic, S.: Efficient and practical constructions of LL/SC variables. In: Proceedings of the 2nd SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), pp. 285–294 (2003)
7. Michael, M.: ABA Prevention Using Single-Word Instructions. Tech. rep., IBM T. J. Watson Research Center (2004)
8. Michael, M.: Practical lock-free and wait-free LL/SC/VL implementations using 64-bit CAS. In: Proceedings of the 18th International Symposium on Distributed Computing (DISC), pp. 144–158 (2004)
9. Moir, M.: Practical implementations of non-blocking synchronization primitives. In: Proceedings of the 16th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), pp. 219–228 (1997)

Affiliations

Faith Ellen[1] · Rati Gelashvili[1] · Philipp Woelfel[2] · Leqi Zhu[1]

Faith Ellen
faith@cs.toronto.edu

Rati Gelashvili
gelash@cs.toronto.edu

Leqi Zhu
lezhu@cs.toronto.edu

[1] Department of Computer Science, University of Toronto, Toronto, ON, Canada

[2] Department of Computer Science, University of Calgary, Calgary, AB, Canada