Space Lower Bounds for the Signal Detection Problem

Faith Ellen
University of Toronto, Canada
faith@cs.toronto.edu

Rati Gelashvili
University of Toronto, Canada
gelash@cs.toronto.edu

Philipp Woelfel
University of Calgary, Canada
woelfel@cpsc.ucalgary.ca

Leqi Zhu
University of Toronto, Canada
lezhu@cs.toronto.edu

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. We prove a lower bound of $m \geq n^2$, as well as a tight lower bound of $m \geq 2^n$ for two restricted versions of the problem, where the processes are oblivious or where the signaller always resets the blackboard to the same fixed value. We also consider a one-shot version of the problem, where each reader takes at most two steps. In this case, we prove that it is necessary and sufficient that the blackboard can store $m = n+1$ values.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Signal detection, ABA problem, space complexity, lower bound

Digital Object Identifier 10.4230/LIPIcs.STACS.2019.26

Funding We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). This research was undertaken, in part, thanks to funding from the Canada Research Chairs program. Rati Gelashvili was supported by the University of Toronto Faculty of Arts & Science Postdoctoral Fellowship.

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

36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019).
Editors: Rolf Niedermeier and Christophe Paul; Article No. 26; pp. 26:1–26:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

scheduler. Whenever a process takes a step, it atomically reads the blackboard and can modify its contents arbitrarily, i.e. 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 even unbounded values of m, there are simple solutions to the signal detection problem. For example, the board 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 number of values the blackboard needs to store grows with the number of signals that occur, which can be unbounded.

The following simple protocol works for all executions and needs only to store an n-bit string $(b_1, \dots, b_n)$ on the blackboard. Initially, $b_1 = \cdots = b_n = 0$, and whenever the signaller takes a step, it sets all bits to 1. For each $j \in \{1, \dots, n\}$, reader r_j resets bit b_j to 0, returns *false* if this is r_j's first step, and returns the old value of b_j otherwise.

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 [6], 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 [8].

In some cases, it is possible to store the tag in an unrelated memory location [7], 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, handshaking bits 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 [10], and "offer little or no help with preventing the ABA problem" [9].

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 read/write registers or 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 on 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. Therefore, if m^* is the smallest value of m (the number of values stored on the blackboard) for which signal detection can be solved, 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 $m \geq 2^n$. This simply defined combinatorial problem does not seem to have a simple solution and a proof of the conjecture has eluded us so far. Even a proof of a polynomial lower bound is surprisingly non-trivial. We show the following.

▶ **Theorem 1.** *In any algorithm for the signal detection problem, the blackboard stores* $m = \Omega(n^2)$ *different values.*

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 a one-shot version of signal detection, where no reader takes more than two steps (but the signaller can take arbitrarily many steps). We show that this problem is strictly easier than the unrestricted version of the problem by showing that one-shot signal detection can be solved with $n + 1$ different blackboard values, which is optimal.

▶ **Theorem 2.** *The minimum number of different values that the blackboard stores in an algorithm that solves the one-shot signal detection problem is* $m = n + 1$*.*

Then we consider the case of *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 process' internal state (but the return value of a reader's step may not be). In the simple algorithm above, which uses $m = 2^n$ blackboard values, processes are oblivious. In fact, what a reader returns also only depends on the contents of the blackboard and not on its internal state. We prove that when processes are oblivious, no better algorithm exists.

▶ **Theorem 3.** *In any algorithm for the signal detection problem with oblivious processes, the blackboard stores* $m \geq 2^n$ *different values.*

The signal detection problem with 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 signaler 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 3.

We also consider signal detection with *identical signals*, where the signaller always resets the blackboard to the same value. Note that the simple algorithm above with $m = 2^n$ uses identical signals. Studying this restricted problem has another motivation: Consider a shared memory system, where shared memory objects may be reset to their initial states at arbitrary times. For example, this can happen due to power outages if volatile memory is used. A solution to signal detection with identical signals corresponds to an algorithm where processes can detect that such faults have happened. This may allow them to start a recovery procedure. This is dual to the recently introduced notion of *recoverable* algorithms [5, 4, 3], which tolerate power outages when the local variables of processes are stored on volatile memory, but shared memory is non-volatile.

▶ **Theorem 4.** *In any algorithm for the signal detection problem with identical signals, the blackboard stores $m \geq 2^n$ different values.*

The lower bound proofs of $m \geq 2^n$ for signal detection with either oblivious readers or identical signals have one interesting aspect in common. We show 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 $(r_1, \ldots, r_n)$. For our simple algorithm, each execution that ends with the signaller taking a step results in a configuration with this property. We show that a lower bound proof for the unrestricted signal detection problem cannot rely on this property. In particular, we present an algorithm for two readers, r_1 and r_2, which uses a bounded number of blackboard values, such that every reachable configuration C satisfies the following: the schedules sr_1 and sr_2 performed starting from C result in configurations with the same blackboard contents. Hence, in contrast to our earlier intuition, it is not necessary for the blackboard to store information about which processes have taken steps since the signaller last took a step: Csr_1 and Csr_2 are indistinguishable to the signaller. This algorithm uses $m = 16$ blackboard values, so it does not contradict our conjecture. However, it has interesting implications for lower bound proof techniques - for example, the approach that we used to prove Theorem 4 does not apply to this particular algorithm.

2 Preliminaries

We consider a deterministic, asynchronous system in which $n + 1$ processes with unique IDs in $\{s, r_1, \ldots, r_n\}$ 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. If C is a configuration and α is a finite execution starting from C, then $C\alpha$ denotes the configuration at the end of α. 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). For any (deterministic) algorithm and for any configuration C, a schedule α determines a unique execution starting from C in which the processes take steps in the order specified by the schedule. The configuration at the end of this execution is called $C\alpha$.

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 execution from C, then it is also an execution from C', and $C\alpha$ and $C'\alpha$ are also indistinguishable to P.

3 One-Shot Signal Detection

Recall that in the one-shot signal detection problem, no reader takes more than two steps, but the signaller can take arbitrarily many steps. Consider the following algorithm that solves this problem using $m = n + 1$ values:

- 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 *false*. Let $v_i \neq 0$ denote the value of the blackboard immediately after this step.
- 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 a reader only changes 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*.

There is also a matching lower bound. In both of the following proofs, it is sufficient to restrict attention to executions in which each reader takes at most two steps.

▶ **Lemma 5.** *Let C be a configuration and let r be a reader. If α is a $(\{r_1, \ldots, r_n\} - \{r\})$-only execution from $C' = Cr$ and β is a $(\{s, r_1, \ldots, r_n\} - \{r\})$-only execution from $C'\alpha s$, then, for every configuration D in α and every configuration E in β, $\mathsf{v}(D) \neq \mathsf{v}(E)$.*

Proof. Suppose not. Then there is some configuration D in α and some configuration E in β such that $\mathsf{v}(D) = \mathsf{v}(E)$. Since r takes no steps in $\alpha s \beta$, 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 2, restated for convenience:

▶ **Theorem 2.** *The minimum number of different values that the blackboard stores in an algorithm that solves the one-shot signal detection problem is $m = n + 1$.*

Proof. In the beginning of this section, we gave an algorithm for one-shot signal detection in which blackboard stores $n + 1$ values. In the following we show that, in any algorithm for one-shot signal detection, the blackboard stores at least $n + 1$ different values.

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

For $1 \leq i < n$, consider the empty execution α from C_i and the execution β from $C_i s$ with schedule $r_{i+1} \cdots s r_n s$. By Lemma 5 with $C' = C_i$ and $r = r_i$, $\mathsf{v}(C_i) \neq \mathsf{v}(E)$ for all configurations E in β. In particular, $\mathsf{v}(C_i) \neq \mathsf{v}(C_j)$ for $i + 1 \leq j \leq n + 1$.

For $i = n$, consider the empty execution α from C_n and the empty execution β from $C_n s = C_{n+1}$ By Lemma 5 with $C' = C_n$ and $r = r_n$, $\mathsf{v}(C_n) \neq \mathsf{v}(C_{n+1})$.

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

There is a simple generalization of the algorithm for one-shot signal detection using $m = n + 1$ values to an algorithm for signal detection using $m = bn + 1$ values when each reader can perform at most $b + 1$ steps: When a reader r_i reads a 0 from the blackboard in its j'th step, for $1 \leq j \leq b$, it changes the blackboard contents to (i, j), instead of i, and stores the value of the blackboard in v_i. When r_i takes its first step, it always returns *false*. When r_i takes subsequent steps, it returns *false* if it reads v_i from the blackboard; otherwise it returns *true*.

4 Identical Signals

Suppose that the signaller always resets the contents of the blackboard to a fixed value, say 0. We show that the blackboard must be able to store at least 2^n values.

Given a set of readers, R, let $\vec{R}$ 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 6.** *If the blackboard can only store a finite number of different values, then it is possible to reach a configuration D such that, for every set of readers, T, there is a $(M(T) \cup \{s\})$-only execution β from $D\vec{T}s$ such that $\mathsf{v}(D\vec{T}s\beta) = \mathsf{v}(D\vec{T})$.*

Proof. Assume that, for all reachable configurations C, there is a set of readers, T, such that, for all $(M(T) \cup \{s\})$-only executions β from $C\vec{T}s$, $\mathsf{v}(C\vec{T}s\beta) \neq \mathsf{v}(C\vec{T})$. We define an infinite sequence $(C_i)_{i \geq 0}$ of reachable configurations as follows. Let C_0 be the initial configuration. For $j \geq 1$, let T_j be a set of readers such that for all $(M(T_j) \cup \{s\})$-only executions β from $C_{j-1}\vec{T}_j s$, $\mathsf{v}(C_{j-1}\vec{T}_j s\beta) \neq \mathsf{v}(C_{j-1}\vec{T}_j)$. The existence of T_j follows from the assumption, since C_{j-1} is reachable. Let $C_j = C_{j-1}\vec{T}_j s$.

Consider the infinite sequence $(M(T_j))_{j \geq 1}$. Since $|\{M(R) : R \subseteq \{r_1, \ldots, r_n\}\}| = n + 1$, some set of readers occurs in the sequence infinitely often. Let M be the largest such set, let $J = \{j \geq 1 : M(T_j) = M\}$, 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 $\vec{T}_{k+1} s \vec{T}_{k+2} s \cdots \vec{T}_\ell$ is $(M \cup \{s\})$-only. Thus, by definition of T_k, $\mathsf{v}(C_k\vec{T}_k) \neq \mathsf{v}(C_k\vec{T}_k s\vec{T}_{k+1} s \cdots \vec{T}_\ell) = \mathsf{v}(C_\ell\vec{T}_\ell)$. Therefore, the blackboard can store an infinite number of values. ◀

This allows us to prove Theorem 4 (restated):

▶ **Theorem 4.** *In any algorithm for the signal detection problem with identical signals, the blackboard stores $m \geq 2^n$ different values.*

Proof. Suppose the blackboard can only store a finite number of values. Then, by Lemma 6, it is possible to reach a configuration D such that, for any set of readers T, there is a $(M(T) \cup \{s\})$-only execution β from $D\vec{T}s$ such that $\mathsf{v}(D\vec{T}s\beta) = \mathsf{v}(D\vec{T})$.

Suppose there exist two different sets of readers $R, R' \subseteq \{r_1, \ldots, r_n\}$ such that $\mathsf{v}(D\vec{R}) = \mathsf{v}(D\vec{R'})$. Without loss of generality, $\vec{R} = \vec{T}x\vec{X}$ and $\vec{R'} = \vec{T}\vec{X'}$, where $x \in R - R'$ and $\vec{T}$ is the longest common prefix of $\vec{R}$ and $\vec{R'}$. Note that $M(T) \cap (\{x\} \cup X \cup X') = \emptyset$ since $\vec{R}$ and $\vec{R'}$ are sorted. By definition of D, there is a $(M(T) \cup \{s\})$-only execution β from $D\vec{T}s$ such that $\mathsf{v}(D\vec{T}s\beta) = \mathsf{v}(D\vec{T})$. Consider the execution β' from $D\vec{T}xs$ which has the same schedule as β. Since $D\vec{T}s$ and $D\vec{T}xs$ are indistinguishable to $M(T)$ and s always sets the blackboard to 0, the corresponding configurations in β and β' are indistinguishable to $M(T)$. In particular,

$\mathsf{v}(D\vec{T}xs\beta) = \mathsf{v}(D\vec{T})$. Since $(M(T) \cup \{s, x\}) \cap X' = \emptyset$, configurations $D\vec{T}xs\beta$ and $D\vec{T}$ are indistinguishable to the set of readers X'. Thus $\mathsf{v}(D\vec{T}xs\beta\vec{X}') = \mathsf{v}(D\vec{T}\vec{X}') = \mathsf{v}(D\vec{R}') = \mathsf{v}(D\vec{R}) = \mathsf{v}(D\vec{T}x\vec{X})$. Since $x \notin M(T) \cup X \cup X' \cup \{s\}$, it follows that $D\vec{T}xs\beta\vec{X}'$ and $D\vec{T}x\vec{X}$ are indistinguishable to x. Note that x must return *false* if it takes a step in configuration $D\vec{T}x\vec{X}$, 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 $D\vec{T}xs\beta\vec{X}'$, because s has taken a step since x last took a step. This is impossible, because these two configurations are indistinguishable to x.

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

If the signaller can only read from the blackboard and write to the blackboard, but cannot perform atomic read-modify-write operations, the blackboard must also store at least 2^n different values. The same proof works, provided the scheduler only lets the signaller write to the blackboard in solo executions that begin with a read of the blackboard. In such executions, the signaller writes a fixed sequence of values, that does not depend on the steps taken by the readers. This is all that is necessary to prove that the corresponding configurations in β and β' are indistinguishable to $M(T)$ and, therefore, $\mathsf{v}(D\vec{T}xs\beta) = \mathsf{v}(D\vec{T})$.

5 Oblivious Processes

Recall that a process is 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. In this section we prove Theorem 3, which we restate for convenience:

▶ **Theorem 3.** *In any algorithm for the signal detection problem with oblivious processes, the blackboard stores $m \geq 2^n$ different values.*

Proof. Suppose the blackboard stores fewer than 2^n different values. For every (possibly empty) set of readers R and every positive integer i, consider the schedule $\rho_i(R)$, which consists of $s\vec{R}$ repeated i times. Because the blackboard stores fewer than 2^n different values, the blackboard contents will repeat when schedule $\rho_{2^n}(R)$ is applied starting from the initial configuration, C_0. Let $L(R) = \mathsf{v}(C_0\rho_\ell(R))$, where ℓ is the index of the first repetition in the sequence $\mathsf{v}(C_0\rho_i(R))_{i\geq 1}$.

Let R and R' be any two different sets of readers. Without loss of generality, suppose there is a reader $r_k \in R' \setminus R$. To obtain a contradiction, assume that $L(R) = L(R')$. Let $0 < i < j$ and $0 < i' < j'$ be such that $L(R) = \mathsf{v}(C_0\rho_i(R)) = \mathsf{v}(C_0\rho_j(R))$ and $L(R') = \mathsf{v}(C_0\rho_{i'}(R')) = \mathsf{v}(C_0\rho_{j'}(R'))$. Since processes are oblivious, and $\mathsf{v}(C_0\rho_{i'}(R')) = L(R') = L(R) = \mathsf{v}(C_0\rho_i(R))$, it follows that $\mathsf{v}(C_0\rho_{i'}(R')\rho_{j-i}(R)) = \mathsf{v}(C_0\rho_i(R)\rho_{j-i}(R)) = \mathsf{v}(C_0\rho_j(R)) = \mathsf{v}(C_0\rho_{i'}(R'))$. Since r_k takes no steps in $\rho_{j-i}(R)$, configurations $C_0\rho_{i'}(R')\rho_{j-i}(R)$ and $C_0\rho_{i'}(R')$ are indistinguishable to r_k. This is impossible, as the signaller has taken a step after r_k's last step in $C_0\rho_{i'}(R')\rho_{j-i}(R)$, but not in $C_0\rho_{i'}(R')$, so r_k would have to return different responses if it takes the next step. Thus, if $R \neq R'$, then $L(R) \neq L(R')$.

However, since there are 2^n different sets of readers and the blackboard stores fewer than 2^n different values, this contradicts the pigeon-hole principle. ◀

6 The General Setting

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

▶ **Lemma 7.** *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 pair of executions α and β from D, there exists an $(M(\alpha) \cup M(\beta) \cup \{s\})$-only execution γ from $D\alpha$ such that $\mathsf{v}(D\alpha\gamma) = \mathsf{v}(D\beta)$.*

Proof. Let C_0 be an arbitrary configuration. To obtain a contradiction, suppose that, for all configurations C reachable from C_0, there are two executions, α and β from C such that for all $(M(\alpha) \cup M(\beta) \cup \{s\})$-only executions γ from $C\alpha$, $\mathsf{v}(C\alpha\gamma) \neq \mathsf{v}(C\beta)$.

We inductively define an infinite execution δ starting from C_0 and an infinite sequence of configurations C_j, for $j \geq 0$, in this execution such that C_j precedes C_{j+1}. In particular, C_j is reachable from C_0, so there exist two executions, α_{j+1} and β_{j+1} from C_j such that for all $(M(\alpha_{j+1}) \cup M(\beta_{j+1}) \cup \{s\})$-only executions γ from $C_j\alpha_{j+1}$, $\mathsf{v}(C_j\alpha_{j+1}\gamma) \neq \mathsf{v}(C_j\beta_{j+1})$. Let $C_{j+1} = C_j\alpha_{j+1}$ and let $\delta = \alpha_1\alpha_2\cdots$.

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 in $\mathcal{M}$ that occurs an infinite number of times in $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \ldots$. 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 such 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 execution from $C_{k-1}\alpha_k$. 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. This contradicts the assumption that the blackboard can only store finite number of different values. ◀

Let D be a configuration such that, for any pair of executions α and β from D, there exists an $(M(\alpha) \cup M(\beta) \cup \{s\})$-only execution γ from $D\alpha$ such that $\mathsf{v}(D\alpha\gamma) = \mathsf{v}(D\beta)$. 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 $\delta(2,3) = r_1sr_2sr_3$.

▶ **Lemma 8.** *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 $j < j'$. Let $\delta' = r_{j+1}\cdots r_{j'}$, so $\delta(i',j') = \delta(i,j)\delta'$. By Lemma 7, where α is the execution of schedule $\delta(i,j)s$ starting from D and β is the execution of schedule $\delta(i,j)$ starting from D, there exists an $\{r_1,\ldots,r_j,s\}$-only execution 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 $r_1,\ldots,r_j$, and s, since the signaller and these readers take no steps in δ'. Let g be the schedule of execution γ. Then $\mathsf{v}(D\delta(i',j')sg) = \mathsf{v}(D\delta(i,j)sg) = \mathsf{v}(D\delta(i,j)s\gamma) = \mathsf{v}(D\delta(i,j)) = \mathsf{v}(D\delta(i',j'))$. Since r_{j+1} does not appear in sg, configurations $D\delta(i',j')sg$ 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 $D\delta(i',j')sg$, because s has taken a step since r_{j+1} last took a step. This is impossible, because $D\delta(i',j')sg$ and $D\delta(i',j')$ are indistinguishable to r_{j+1}. ◀

Using this lemma, we obtain Theorem 1 (restated for convenience):

▶ **Theorem 1.** *In any algorithm for the signal detection problem, the blackboard stores $m = \Omega(n^2)$ different values.*

Proof. Consider any algorithm for signal detection in which the blackboard stores a finite number of different values. By Lemma 7, there is a reachable configuration D such that, for any pair of executions α and β from D, there exists an $(M(\alpha) \cup M(\beta) \cup \{s\})$-only execution γ from $D\alpha$ such that $\mathsf{v}(D\alpha\gamma) = \mathsf{v}(D\beta)$. By Lemma 8, 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 Process Algorithm

We describe an algorithm for signal detection among $n = 2$ readers, r_1 and r_2, using $m = 16$ values. The algorithm has the property that, for every reachable configuration C, $\mathsf{v}(Csr_1) = \mathsf{v}(Csr_2)$. This will allow us to show that, from any reachable configuration C, the number of different blackboard values that can be reached from C using $\{r_1, r_2\}$-only executions is at most 3. Thus, in order to show the existence of 4 different blackboard values from some configuration C, the signaller must also take steps. Note that our proof of the reset case does not do this, so it is unlikely to be generalized.

At all times, the contents of the blackboard is a quadruple $(track, position, both, flag) \in \{0, 1\}^4$. Initially, the blackboard has value $(0, 0, 1, 1)$. The $flag$ is used to indicate whether the last step was taken by the signaller. In particular, the signaller always sets $flag$ to 1 and the readers always set $flag$ to 0. Each reader r_i has 3 local variables, t_i, p_i and $jump_i$. Initially, $(t_i, p_i) = (0, 0)$ and $jump_i = false$. Variables t_i and p_i represent the last values that r_i wrote to the $track$ and $position$ fields of the blackboard, even if it didn't change their values. Readers only change these fields when the signaller sets $flag$ to 1. If $t_1 = t_2$ and $p_1 = p_2$ in some configuration C, then $both = 1$ in C. Otherwise, it is 0. If variable $jump_i$ is $true$, then when r_i takes its next step, it will change the track, provided it sees $track = t_i$, $position = p_i$, $both = 0$, and $flag = 1$ on the blackboard.

Suppose r_i reads (t, p, b, f) from the blackboard. Then, in its next step, r_i does the following:

1. If $f = 1$ and $b = 1$, then r_i changes $track$, sets $position$ to 0, sets $both$ to 0, and sets $jump_i$ to $false$.
2. If $f = 1$, $b = 0$, $t = t_i$, $p = p_i$ and $jump_i = false$, then r_i only changes $position$.
3. If $f = 1$, $b = 0$, $t = t_i$, $p = p_i$, and $jump_i = true$, then r_i changes $track$, sets $position$ to 0, and sets $jump_i$ to $false$.
4. If $f = 1$, $b = 0$, $t = t_i$, and $p \neq p_i$, then r_i changes $track$ and sets $jump_i$ to $false$.
5. If $f = 1$, $b = 0$, and $t \neq t_i$, then r_i changes $position$ and sets $jump_i$ to $true$.
6. If $f = 0$ and $t \neq t_i$ or $p \neq p_i$, then r_i sets $both$ to 1 and sets $jump_i$ to $false$.
7. If $f = 0$, $t = t_i$, and $p = p_i$, then r_i doesn't change anything.

In the first 6 cases, r_i returns $true$. In case 7, r_i returns $false$. Pseudocode appears in Algorithm 1.

Note that consecutive steps by a process do not change its state or the blackboard. Moreover, if s takes a step followed by an $\{r_1, r_2\}$-only execution in which they each take at least one step, then, in the resulting configuration, their t_i and p_i variables will be equal to track and position on the blackboard. From then on, r_1 and r_2 will not change their local variables or the blackboard until the next signaller step. Therefore, we may restrict attention to schedules of the form $\alpha_1 s\alpha_2 \cdots s\alpha_\ell$ and $s\alpha_1 s\alpha_2 \cdots s\alpha_\ell$, where each α_k is an $\{r_1, r_2\}$-only schedule in which r_1 and r_2 each occur at most once and α_k is non-empty for $k < \ell$. Since *flag* is initially 1, a step by the signaller does not change the value of the blackboard. Hence we may assume α begins with s.

Algorithm 1: Pseudocode for reader r_i.

$(\mathit{track}, \mathit{position}, \mathit{both}, \mathit{flag}) \leftarrow$ **read** from blackboard
if $(\mathit{flag} = 0) \wedge ((\mathit{track}, \mathit{position}) = (t_i, p_i))$ **then**
  return *false*
if $(\mathit{flag} = 0)$ **then**
  write $(\mathit{track}, \mathit{position}, 1, 0)$ to blackboard
else if $(\mathit{both} = 0) \wedge ((\mathit{track} \neq t_i) \vee ((\mathit{position} = p_i) \wedge \neg \mathit{jump}_i))$ **then**
  write $(\mathit{track}, 1 - \mathit{position}, 0, 0)$ to blackboard
else
  write $(1 - \mathit{track}, 0, 0, 0)$ to blackboard
$\mathit{jump}_i \leftarrow (\mathit{flag} = 1) \wedge (\mathit{both} = 0) \wedge (\mathit{track} \neq t_i)$
$(t_i, p_i) \leftarrow$ last track and position written
return *true*

Given a reachable configuration C, let $t_i(C)$, $p_i(C)$, and $\mathit{jump}_i(C)$ denote the value of reader r_i's local variables t_i, p_i, and jump_i, respectively, in C, and let $\mathit{track}(C)$, $\mathit{position}(C)$, $\mathit{both}(C)$, and $\mathit{flag}(C)$ denote the values of the *track*, *position*, *both*, and *flag* fields, respectively, on the blackboard in C.

▶ **Lemma 9.** *For every reachable configuration C and every $i \in \{1, 2\}$,*

$$(t_i(C), p_i(C)) \neq (\mathit{track}(Csr_{3-i}), \mathit{position}(Csr_{3-i}))\,.$$

Proof. Suppose, for a contradiction, that this is not the case. Consider a shortest schedule α such that, in configuration $C = C_0\alpha$, $(t_i(C), p_i(C)) = (\mathit{track}(Csr_{3-i}), \mathit{position}(Csr_{3-i}))$, for some $i \in \{1, 2\}$. As discussed above, $\alpha = s\alpha_1 \cdots s\alpha_\ell$, where each α_k is an $\{r_1, r_2\}$-only schedule in which r_1 and r_2 each occur at most once and α_k is non-empty for $k < \ell$.

Suppose r_i does not occur in α. Then $t_i(C) = t_i(C_0) = 0$. Since $\mathit{flag}(C_0) = 1$, $\mathit{both}(C_0) = 1$, and $\mathit{track}(C_0) = 0$, $\mathit{track}(C_0 sr_{3-i}) = 1$. If only s and r_{3-i} take steps from $C_0 sr_{3-i}$, then neither changes *track*. Since α is a $\{s, r_{3-i}\}$-only schedule, $\mathit{track}(Csr_{3-i}) = 1$. Therefore, $t_i(C) = 0 \neq 1 = \mathit{track}(Csr_{3-i})$, which contradicts the fact that $t_i(C) = \mathit{track}(Csr_{3-i})$.

Now suppose that r_i occurs at least once in α. Suppose it last occurs in α_k. Let $C' = C_0 s\alpha_1 \cdots s\alpha_k$. Since r_i does not occur in the remainder of α, $(t_i(C), p_i(C)) = (t_i(C'), p_i(C')) = (\mathit{track}(C'), \mathit{position}(C'))$. There are 3 cases:

Case 1: $k = \ell$. Then $C = C'$. If r_{3-i} takes a step from Cs, it either changes *track* or *position*. Hence, either $t_i(C) \neq \mathit{track}(Csr_{3-i})$ or $p_i(C) \neq \mathit{position}(Csr_{3-i})$. This contradicts the fact that $(t_i(C), p_i(C)) = (\mathit{track}(Csr_{3-i}), \mathit{position}(Csr_{3-i}))$.

Case 2: $k < \ell$ **and** α_k **contains both** r_1 **and** r_2. Then $1 = \mathit{both}(C') = \mathit{both}(C's)$. Hence, if r_{3-i} takes a step from $C's$, it changes *track* to $1 - \mathit{track}(C')$. As r_i does not take any more steps, r_{3-i} does not subsequently change *track*. Thus, $t_i(C) = \mathit{track}(C') \neq 1 - \mathit{track}(C') = \mathit{track}(Csr_{3-i})$. This contradicts the fact that $t_i(C) = \mathit{track}(Csr_{3-i})$.

Case 3: $k < \ell$ **and** $\alpha_k = r_i$. Since $s\alpha_1 \cdots s\alpha_k$ is strictly shorter than $s\alpha_1 \cdots s\alpha_\ell$, $(t_i(C'), p_i(C')) \neq (\mathit{track}(C'sr_{3-i}), \mathit{position}(C'sr_{3-i}))$. Moreover, if $\alpha = s\alpha_1 \cdots s\alpha_k s$, then $C = C's$, so $(t_i(C), p_i(C)) \neq (\mathit{track}(Csr_{3-i}), \mathit{position}(Csr_{3-i}))$, contrary to the assumption. Hence $\alpha_{k+1} = r_{3-i}$.

Suppose $\mathit{track}(C'sr_{3-i}) \neq t_i(C')$ or $\mathit{jump}_{3-i}(C'sr_{3-i})$ is *true*. In the first case, r_{3-i} changed from $t_i(C') = \mathit{track}(C')$ to $1 - \mathit{track}(C')$ and it stays on this track. In the second case, $\mathit{track}(C'sr_{3-i}sr_{3-i}) = 1 - \mathit{track}(C')$ and it stays on this track. In either case, this implies that $\mathit{track}(Csr_{3-i}) = 1 - \mathit{track}(C')$. Therefore, $t_i(C) = \mathit{track}(C') \neq \mathit{track}(Csr_{3-i})$, which is a contradiction.

Now suppose that $\mathit{track}(C'sr_{3-i}) = t_i(C')$ and $\mathit{jump}_{3-i}(C'sr_{3-i})$ is false. Since r_{3-i} did not change the track and jump_{3-i} is false, $(t_{3-i}(C'), p_{3-i}(C')) = (\mathit{track}(C'), \mathit{position}(C'))$. Let $C'' = C_0 s\alpha_1 \cdots s\alpha_{k-1}$. Since $\alpha_k = r_i$, $(t_{3-i}(C''), t_{3-i}(C'')) = (t_{3-i}(C'), t_{3-i}(C'))$. Since the latter is $(\mathit{track}(C'), \mathit{position}(C')) = (\mathit{track}(C''sr_i), \mathit{position}(C''sr_i))$, this contradicts the minimality of α.

Therefore, in all cases, we reach the desired contradiction. ◀

The next lemma proves that the algorithm has the desired property.

▶ **Lemma 10.** *For every reachable configuration C, $\mathsf{v}(Csr_1) = \mathsf{v}(Csr_2)$.*

Proof. Suppose, for a contradiction, that this is not the case. Consider a shortest α from the initial configuration C_0 such that $\mathsf{v}(C_0\alpha sr_1) \neq \mathsf{v}(C_0\alpha sr_2)$. Let $C = C_0\alpha$ and write $\alpha = s\alpha_1 \cdots s\alpha_\ell$. Notice that α_ℓ is non-empty as, otherwise, $\alpha' = s\alpha_1 \cdots s\alpha_{\ell-1}$ is a shorter execution such that $\mathsf{v}(C_0\alpha' sr_1) \neq \mathsf{v}(C_0\alpha' sr_2)$. Moreover, $\alpha_\ell \notin \{r_1r_2, r_2r_1\}$ since, otherwise, $\mathit{both}(Cs) = 1$ and both r_1 and r_2 set the blackboard to $(1 - \mathit{track}(C), 0, 0, 0)$ from Cs. Hence, $\alpha_\ell = r_i$, for some $i \in \{1, 2\}$. Then $(t_i(C), p_i(C)) = (\mathit{track}(C), \mathit{position}(C))$. Let $C' = C_0 s\alpha_1 \cdots s\alpha_{\ell-1}$. There are two cases to consider.

Case 1: $\mathit{jump}_i(C)$ is true. By the pseudocode, since $\mathit{jump}_i(C)$ is true, r_i saw that $\mathit{both}(C's) = 0$, $\mathit{track}(C's) \neq t_i(C's)$, and it wrote $t_i(C) = \mathit{track}(C')$ and $p_i(C) = 1 - \mathit{position}(C')$. Since $\mathit{track}(C') \neq t_i(C's)$, it must be that $\mathit{track}(C's) = t_{3-i}(C's)$. It follows that the next step of r_{3-i} in Cs is to write $(1 - \mathit{track}(C), 0, 0, 0)$. However, this is precisely what r_i does in its next step from Cs as well since jump_i is true and $(t_i(C), p_i(C)) = (\mathit{track}(C), \mathit{position}(C))$. This is a contradiction.

Case 2: $\mathit{jump}_i(C)$ is false. Then $\mathit{both}(C's) = 0$ and $\mathit{track}(C's) = t_i(C's)$. There are two subcases to consider:

- Suppose that either $\mathit{position}(C's) \neq p_i(C's)$ or $\mathit{jump}_i(C's)$ is true. Then r_{3-i} was the last reader to take a step before C', $t_{3-i}(C's) = \mathit{track}(C's)$, and $p_{3-i}(C's) = \mathit{position}(C's)$. Moreover, the next step of r_i from $C's$ is to write $(1 - \mathit{track}(C's), 0, 0, 0)$. Thus, $\mathit{track}(C) \neq \mathit{track}(C')$. It follows that the next step of r_{3-i} from Cs is to write $(\mathit{track}(C), 1 - \mathit{position}(C), 0, 0)$. This is a contradiction.
- Suppose that $\mathit{position}(C's) = p_i(C's)$ and $\mathit{jump}_i(C's)$ is false. Since $\mathit{both}(C's) = 0$, C' cannot be the initial configuration and, thus, r_i was the last reader to take a step before C' and $\alpha_{\ell-1} = r_i$. By Lemma 9, $(t_{3-i}(C'), p_{3-i}(C')) \neq (\mathit{track}(C'), \mathit{position}(C'))$. By definition of α, $\mathsf{v}(C'sr_i) = \mathsf{v}(C'sr_{3-i})$. Thus, as r_i did not change the track, it must be that $\mathit{track}(C's) \neq t_{3-i}(C's)$ or ($\mathit{position}(C's) = p_{3-i}(C's)$ and $\mathit{jump}_{3-i}(C's)$ is false). Since $(t_{3-i}(C'), p_{3-i}(C')) \neq (\mathit{track}(C'), \mathit{position}(C'))$, it must be the former. Thus, r_{3-i}'s next step from Cs is to write $(\mathit{track}(C), 1 - \mathit{position}(C), 0, 0)$, which is exactly what r_i does. This is a contradiction.

In all cases, we reach the desired contradiction. ◀

▶ **Lemma 11.** *For any configuration C, $|\{\mathsf{v}(C\alpha) : \alpha \text{ is a } \{r_1, r_2\}\text{-only execution}\}| \leq 3$.*

Proof. For any configuration C, let α' be the longest $\{r_1, r_2\}$-only execution such that $C = C'\alpha'$. Notice that it suffices to prove that the claim holds for configuration C'. By our earlier assumption that s takes a step immediately after the initial configuration, we may assume that s took the last step before C'.

By Lemma 10, $\mathsf{v}(C'r_1) = \mathsf{v}(C'r_2)$. Subsequently, from $C'r_i$, the steps by r_i do not change the blackboard value. If r_{3-i} takes a step, then it sets *both* to 1 and $\mathsf{v}(C'r_1r_2) = \mathsf{v}(C'r_2r_1)$. After this, neither r_i or r_{3-i} can change the blackboard. Hence $\{\mathsf{v}(C'\alpha) : \alpha \text{ is a } \{r_1, r_2\}\text{-only execution}\} = \{\mathsf{v}(C'), \mathsf{v}(C'r_1), \mathsf{v}(C'r_1r_2)\}$. ◀

Finally, we prove that the algorithm is correct.

▶ **Lemma 12.** *In every execution, each reader returns the correct responses.*

Proof. Suppose not, so that there is some execution α from the initial configuration C_0 such that, in $C = C_0\alpha$, some r_i returns an incorrect response in its next step from C. Write $\alpha = \alpha' r_i \alpha''$, where α'' is a $\{s, r_{3-i}\}$-only execution. If s does not take any steps in α'', then r_i returns $false$, which is correct. So, s takes at least one step in α'' and r_i returns $false$ in its next step from C. By the pseudocode, this implies that $flag(C) = 0$ and $(t_i(C), p_i(C)) = (track(C), position(C))$. It follows that we may write $\alpha'' = \alpha''' s r_{3-i}$. However, this contradicts Lemma 9 from $C' = C_0\alpha'''$ with r_i. In particular, r_i returns *true* in its next step from $C = C' s r_{3-i}$. ◀

References

1 Yehuda Afek, Hagit Attiya, Danny Dolev, Eli Gafni, Michael Merritt, and Nir Shavit. Atomic Snapshots of Shared Memory. *Journal of the ACM*, 40(4):873–890, 1993.

2 Zahra Aghazadeh and Philipp Woelfel. 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)*, pages 193–202, 2015.

3 Wojciech Golab. Recoverable Consensus in Shared Memory. arXiv preprint, 2018. `arXiv:1804.10597`.

4 Wojciech Golab and Danny Hendler. Recoverable Mutual Exclusion in Sub-logarithmic Time. In *Proceedings of the 36th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 211–220, 2017. `doi:10.1145/3087801.3087819`.

5 Wojciech Golab and Aditya Ramaraju. Recoverable mutual exclusion. In *Proceedings of the 35th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 65–74, 2016.

6 IBM. IBM system/370 extended architecture, principles of operation, 1983. Publication No. SA22-7085.

7 Prasad Jayanti and Srdjan Petrovic. Efficient and practical constructions of LL/SC variables. In *Proceedings of the 2nd SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 285–294, 2003.

8 Maged Michael. ABA prevention using single-word instructions. Technical report, IBM T. J. Watson Research Center, 2004.

9 Maged Michael. Practical Lock-Free and Wait-Free LL/SC/VL Implementations Using 64-Bit CAS. In Rachid Guerraoui, editor, *Proceedings of the 18th International Symposium on Distributed Computing (DISC)*, pages 144–158, 2004.

10 Mark Moir. Practical Implementations of Non-Blocking Synchronization Primitives. In *Proceedings of the 16th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 219–228, 1997.