# A tight RMR lower bound for randomized mutual exclusion

George Giakkoupis, Philipp Woelfel

# A Tight RMR Lower Bound for Randomized Mutual Exclusion

George Giakkoupis*
INRIA Rennes-Bretagne Atlantique
Rennes, France
george.giakkoupis@inria.fr

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

## ABSTRACT

The Cache Coherent (CC) and the Distributed Shared Memory (DSM) models are standard shared memory models, and the Remote Memory Reference (RMR) complexity is considered to accurately predict the actual performance of mutual exclusion algorithms in shared memory systems. In this paper we prove a tight lower bound for the RMR complexity of deadlock-free randomized mutual exclusion algorithms in both the CC and the DSM model with atomic registers and compare&swap objects and an adaptive adversary. Our lower bound establishes that an adaptive adversary can schedule $n$ processes in such a way that each enters the critical section once, and the total number of RMRs is $\Omega(n \log n / \log \log n)$ in expectation. This matches an upper bound of Hendler and Woelfel [16].

## Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed programming*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

## General Terms

Algorithms, Theory

## Keywords

Mutual exclusion, Lower bound, Remote memory references, RMRs, Strong adversary, Randomization

*Part of this work was performed while G. Giakkoupis was a Postdoctoral Fellow at the University of Calgary, and he was supported by the Pacific Institute for the Mathematical Sciences (PIMS), and the Natural Sciences and Engineering Research Council of Canada (NSERC).

†Supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).


*STOC'12,* May 19–22, 2012, New York, New York, USA.


## 1. INTRODUCTION

The *mutual exclusion* problem, introduced by Dijkstra in 1965 [10], is a fundamental and well-studied problem in asynchronous computing. Processes coordinate their access to a shared resource by serializing the execution of a piece of code, called *critical section*.

In this paper we consider the mutual exclusion problem in asynchronous shared memory models that provide atomic registers and compare&swap (CAS) objects. In such models, the number of steps a process executes can be unbounded, since processes may have to wait for other processes to leave the critical section. Therefore, the classical measure of efficiency, step complexity, is meaningless.

In shared memory systems, some of the memory is local to each process, while the rest of the memory is located in other processing units or in dedicated storage. For example, in *cache-coherent (CC)* systems, each processor keeps local copies of (remote) shared variables in its cache; the consistency of copies in different caches is maintained by a *coherence protocol*. In *distributed shared-memory (DSM)* systems, on the other hand, each shared variable is permanently locally accessible to a single processor and remote to all other processors.

References to remote memory (short RMRs) are orders of magnitude slower than accesses to local memory. Hence, the performance of many algorithms for shared memory multiprocessor systems depends critically on the number of RMRs they incur [4, 22], and in particular the efficiency of mutual exclusion algorithms is usually measured in terms of the number of RMRs incurred by processes entering and exiting the critical section. *Local-spin* algorithms, which perform busy-waiting by repeatedly reading locally accessible shared variables, achieve bounded RMR complexity and have practical performance benefits [4]. In fact, recent research on mutual exclusion has almost entirely focused on the RMR complexity of the problem (see, e.g., [3, 2, 21, 6, 9, 18, 19, 20, 7, 15, 16]).

Using strong primitives, such as fetch&increment objects, it is possible to implement mutual exclusion so that every process incurs only a constant number of RMRs per passage through the critical section. A prominent example is the MCS lock [23], which uses an object that allows both compare&swap and swap operations. Other examples can be found in standard textbooks, such as [17]. If the system provides only atomic registers, then the RMR complexity of the mutual exclusion problem is higher. Since objects such as compare&swap can be simulated in $O(1)$ RMRs from atomic

registers [13], they don't affect the RMR complexity of the mutual exclusion problem.

Yang and Anderson [24] presented the first *deterministic* mutual exclusion algorithm for $n$ processes (using atomic registers) in which every process that enters the critical section incurs at most $O(\log n)$ RMRs. Anderson and Kim [1] then conjectured that this is best possible. Following several lower bound proofs [8, 20, 11], Attiya, Hendler, and Woelfel [6] finally proved this conjecture true.

More recently, *randomized* techniques have been employed to improve the efficiency of mutual exclusion algorithms. Hendler and Woelfel [16] presented a randomized algorithm, where each process incurs an expected number of $O(\log n/\log\log n)$ RMRs per passage through the critical section. The algorithm works for the strong adaptive adversary model, where scheduling decisions can depend on all past events, including local coin flips.

Recently, Bender and Gilbert [7] presented a very different approach to solving mutual exclusion. Their algorithm employs approximate counting techniques to guarantee with high probability an amortized RMR complexity of $O(\log^2 \log n)$ per passage through the critical section on the CC model. (However, processes can deadlock with a small probability.) This upper bound bound was shown for a weak, oblivious adversary model, in which the schedule is independent of the random decisions made by processes.

### *Our Results*

In reality, the speed of operations can depend on the random decisions of processes. E.g., the location of register accesses may be decided at random, but due to the memory hierarchy and architecture, the speed of such accesses is not uniform. It would therefore be desirable to achieve a similar RMR complexity as in the algorithm of Bender and Gilbert [7], for stronger adversaries. The strongest "reasonable" adversary is the adaptive adversary, and thus, an algorithm with low RMR complexity for this adversary would guarantee efficiency independent of the system behavior. However, in the face of the fact that the best known algorithm for the adaptive adversary [16] has an $O(\log n/\log\log n)$ expected RMR complexity per passage through the critical section, Bender and Gilbert [7] noted that their "choice of a weaker adversary seems fundamental." We show that this is indeed the case, by proving that any deadlock-free mutual exclusion algorithm for the $n$-process CC or DSM model has an expected RMR complexity of $\Omega(\log n/\log\log n)$ per passage through the critical section, against an adaptive adversary. This lower bound holds even for one-time mutual exclusion algorithms. Specifically, there is an adaptive adversary that schedules $n$ processes in such a way that every process enters the critical section once, and the expectation of the total number of RMRs is $\Omega(n \log n/\log\log n)$. This is the first non-trivial lower bound for the RMR complexity of randomized mutual exclusion, against an adaptive adversary; for weaker adversary models no lower bounds are known.

### *Techniques*

We define a *randomized adaptive* adversary, i.e., the adversary makes random scheduling decisions but the distribution over these decisions is independent of processes' future coin flips. We prove our lower bound on the expected RMR complexity of *any deterministic* mutual exclusion algorithm scheduled by the above randomized adversary. Our result then follows from Yao's Principle [25].

Our randomized adversary schedules processes in rounds, but in every round only a small, randomly chosen fraction of the processes takes steps. This way, it is difficult for processes to "find" other processes. With every process we associate a potential such that the difference between the total potential initially and after all processes have finished is $\Omega(n \log n/\log\log n)$, and we argue that the expected decrease in potential per round is proportional to the expected number of RMRs executed in that round.

Our lower bound proof is very different from previous lower bounds for deterministic mutual exclusion algorithms [8, 20, 11, 6]. Potential functions have been used before to show lower bounds for *deterministic* shared memory algorithm for various problems (see, e.g., [12]). However, we are not aware of any lower bound proofs for *randomized* shared memory algorithms that use potential function techniques.

## 2. MODEL

Our model is an asynchronous shared memory system where a set $\mathcal{P}$ of $n$ randomized processes with unique IDs communicate by executing operations on a (possibly unbounded) set $\mathcal{R}$ of shared atomic registers. Each process is a probabilistic automaton (with a possibly unbounded number of states) that performs a sequence of steps. A step is an atomic read or write operation on a single shared register, followed, optionally, by a coin flip that returns a uniformly random value over some fixed, bounded domain $\Omega$.

Note that there are implementations of linearizable compare&swap primitives from registers [13]. It follows from [14], that these implementations can be used in a randomized adaptive adversary model in place of atomic compare&swap objects without increasing the expected RMR complexity. Therefore, our lower bound holds even when the system provides atomic compare&swap objects.

Our result holds for the *cache-coherent* (CC) model with a write-through cache[1] as well as for the *distributed shared memory* (DSM) model. In fact, we consider a hybrid of both models (similar to a NUMA with caches), where each process has its own *memory segment* of shared registers, which are local to the process. In addition, whenever a process reads a register $r$ that is not in its own segment, it keeps a copy of $r$ in its local cache memory. A coherence protocol ensures that if a write to register $r$ occurs, all cached copies of $r$ get invalidated. In the proof of the lower bound, we assume that even by writing to a register a process obtains a cache-copy of that register. More precisely, we say that process $p$ has a *valid* cached copy of $r$, if $p$ has *accessed* $r$ and no process has written to $r$ since $p$'s last access of $r$. A read operation by $p$ on register $r$ incurs an RMR, if

- $r$ is not in $p$'s memory segment, and
- at the time of the operation $p$ has no valid cached copy of $r$.

A write operation on $r$ incurs an RMR, whenever $r$ is not in $p$'s memory segment. If $p$ has a valid cached copy of $r$ when some process $q \neq p$ writes to $r$, then we say that $q$'s write *invalidates* $p$'s cached copy.

[1]We believe that the arguments used in our proof can be extended to hold for a write-back cache as well.

# 3. DEFINITIONS AND NOTATION

### *Executions*

An *execution* is a (possibly infinite) sequence $E = (op_1, op_2, \dots)$ of operations, where $op_i$ identifies the process $p \in \mathcal{P}$ that invokes the operation, the type of the operation (write or read), the register $r \in \mathcal{R}$ on which it is applied, the value that this operation writes or reads, and finally, if $p$ executes a coin flip after this write/read operation, the outcome of that coin flip. The length of the sequence $E$ is denoted $|E|$.

A *schedule* is a sequence $\sigma = (p_1, p_2, \dots)$ of process IDs. For a deterministic algorithm (i.e., one involving no coin flips), a schedule $\sigma$ yields a unique execution $E(\sigma)$, in which processes take steps in the order determined by $\sigma$, starting from the (unique) initial configuration. (If during execution $E(\sigma)$ a process reaches a final state, i.e., it cannot take any further steps, then we assume it simply passes whenever it is scheduled again.) For a randomized algorithm, in order to obtain a unique execution we also need $n$ (sufficiently large) coin-flip vectors, a vector $\vec{c}_p = (c_{p,1}, c_{p,2}, \dots)$, $c_{p,i} \in \Omega$, for each $p \in \mathcal{P}$. A schedule $\sigma$ together such a collection of coin-flip vectors $\{\vec{c}_p\}_{p\in\mathcal{P}}$ yields a unique execution $E(\sigma, \{\vec{c}_p\})$, in which the $i$-th operation is executed by process $p_i$, and if that operation involves the $j$-th coin flip by process $p$ then the coin flip returns value $c_{p,j}$.

A *configuration* is a description of the states of all processes and the values of all registers. The configuration after the last operation in an execution $E$ is denoted $C(E)$. The concatenation of two executions $E_1$ and $E_2$ is denoted $E_1 \circ E_2$.

### *Adaptive Adversary*

We assume that schedules are generated by an *adaptive adversary* (see, e.g., [5, 14]), and thus can depend on the random values generated by the processes. In the adaptive adversary model, after each step the adversary decides which process takes the next step, and in order to make this decision it can take all preceding events into account, including the results of past coin flips, but not the results of any of the future coin flips. For determinability algorithms, adaptive adversaries are clearly no more powerful than non-adaptive (oblivious), which generate the schedule before the execution starts.

### *Randomized One-Time Mutual Exclusion*

We prove a lower bound for every randomized *one-time* mutual exclusion algorithm. In such an algorithm, every process executes three pieces of code, an *entry section*, a *critical section*, and an *exit section* (in this order). The algorithm satisfies *mutual exclusion* if at any time at most one process is in its critical section.

We assume (w.l.o.g.) that in its critical section a process first reads a shared register $r_{crit}$ and then it writes its ID to this register. No process accesses $r_{crit}$ outside its critical section. Processes that have finished their exit section take no further steps. (Whenever they are scheduled again they simply pass.) We say that a process is *active*, if it has not finished its exit section; otherwise it is *inactive*. For an execution $E$, we let $\mathcal{I}(E)$ denote the set of processes that are inactive in configuration $C(E)$.

Our lower bound holds for any randomized one-time mutual exclusion algorithm $\mathcal{G}$ that satisfies the following progress-condition, which we call *expected deadlock freedom*. Let $C$ be an arbitrary reachable configuration, and let $\mathcal{Q} \subseteq \mathcal{P}$ be the set of processes that have taken at least one step and are still active in $C$. There is an integer $T = T(\mathcal{G}, n) > 0$ such that if we start from configuration $C$ and all processes in $\mathcal{Q}$ take steps in a round-robin fashion (while the remaining processes take no steps), then the expected total number of steps until some process from $\mathcal{Q}$ finishes is at most $T$. We say then that the algorithm satisfies the deadlock freedom property for *step-bound* $T$.

# 4. AN RMR LOWER BOUND FOR A RANDOMIZED ADVERSARY

We describe a *randomized* adaptive adversary $\mathcal{D}$ for an arbitrary *deterministic* mutual exclusion algorithm. Then we prove a lower bound on the RMR complexity of any such deterministic algorithm $L$ scheduled by the randomized adversary $\mathcal{D}$, provided that $L$ finishes in a bounded expected number of step (Lemma 4.2). Further, we prove that any randomized mutual exclusion algorithm $\mathcal{G}$ that satisfies expected deadlock freedom finishes in a bounded expected number of steps when scheduled by a deterministic adversary obtained by fixing the random choices of $\mathcal{D}$ (Lemma 4.3).

## 4.1 The Randomized Adversary

W.l.o.g. we assume that whenever a processes $p$ writes some value $x$ to a register it also writes its ID to the register, i.e., it writes the pair $(x, p)$. We say that $p$ is *visible* on register $r$, if the value of $r$ is $(x, p)$ for some arbitrary value $x$. Also, we assume (w.l.o.g.) that $n = k^k$ for some integer $k$, and we define

$$\varepsilon := k^{-4}.$$

We now define randomized adaptive adversary $\mathcal{D}$. The adversary schedules processes in batches as follows. We describe below a randomized adversary $\mathcal{D}(\mathcal{P}')$, for any $\mathcal{P}' \subseteq \mathcal{P}$, which schedules only a small random subset $\mathcal{P}''$ of the processes in $\mathcal{P}'$, until all processes in $\mathcal{P}''$ finish. The expected size of $\mathcal{P}''$ is $\varepsilon \cdot |\mathcal{P}'|$. Adversary $\mathcal{D}$ now is composed of $1/\varepsilon$ adversaries $\mathcal{D}(\mathcal{P}'_i)$, $1 \le i \le 1/\varepsilon$, where

$$\mathcal{P}'_i = \begin{cases} \mathcal{P} & \text{if } i = 1; \\ \mathcal{P}'_{i-1} - \mathcal{P}''_{i-1} & \text{if } 1 < i \le 1/\varepsilon, \end{cases}$$

and $\mathcal{P}''_i$ is the set of processes scheduled by $\mathcal{D}(\mathcal{P}'_i)$. First, processes are scheduled by $\mathcal{D}(\mathcal{P}'_1)$ until all processes in $\mathcal{P}''_1$ finish, then remaining processes are scheduled by $\mathcal{D}(\mathcal{P}'_2)$ until the processes in $\mathcal{P}''_2$ finish, and so on. Finally, any processes remaining are scheduled deterministically in a round-robin fashion.

In the rest of this section we describe the randomized adversary $\mathcal{D}(\mathcal{P}')$, for an arbitrary set $\mathcal{P}' \subseteq \mathcal{P}$. The adversary schedules processes in $n^2$ rounds; in each round zero or more processes take steps. Every round consists of three phases, an *RMR Phase*, a *Roll-Forward Phase*, and a *Local Step Phase*. We use a stack to keep track of the sets of processes to schedule in rounds. A stack element is a set of process IDs.

At the beginning (in round 0), a subset $\mathcal{P}''$ of the processes in $\mathcal{P}'$ is selected at random: each process from $\mathcal{P}'$ is added to $\mathcal{P}''$ with probability $\varepsilon$ (independently). Next we schedule each process in set $\mathcal{P}''$ to take steps until it is poised to

execute an RMR, and we push the set $\mathcal{P}''$ onto the stack. Throughout the execution, only processes in $\mathcal{P}''$ will take steps.

We now describe round $i$, for $1 \leq i \leq n^2$. We maintain the invariant that at the beginning of each round, every process in $\mathcal{P}''$ that is still active (i.e., it has not finished its exit section) is poised to perform an RMR. Clearly this invariant is true at the beginning of the first round.

### *RMR Phase*

Round $i$ starts with an RMR phase. Suppose that the stack is not empty at the beginning of the round. (If the stack is empty, then all processes in $\mathcal{P}''$ must have finished their exit section, as will become clear, and thus we can stop scheduling processes.) We pop the topmost set of processes from the stack, and let $G_i$ be the subset of the processes from that set that are still active. If $G_i = \emptyset$, the round ends immediately, and we proceed to the next round, $i + 1$, if $i < n^2$.

Suppose that $G_i \neq \emptyset$. For each register $r \in \mathcal{R}$, we flip a biased random coin with one green side and the other red. All coin flips are independent, and the coin shows the green side with probability $\varepsilon$ and the red side with probability $1 - \varepsilon$. If the coin flipped for some register $r$ is green, then we say that register $r$ is green in round $i$; otherwise we say it is red. Based on the outcome of the coin flips, we partition $G_i$ into processes that *participate* in the RMR Phase of the current round, and processes that are *halted* in this round. A process $p \in G_i$ participates in the RMR Phase, if it is poised to execute its next operation on a green register; otherwise (i.e., if poised to execute its operation on a red register), process $p$ is halted in the current round. (Note that the adaptive adversary always knows on which register what operation a process is poised to execute.) Halted processes don't take any steps in the current round, unless the get *rolled forward* (see the description of the Roll-Forward Phase). All processes that participate in the RMR Phase execute exactly one step in that phase; first the processes that are poised to read (in the order of increasing process IDs), and then the processes that are poised to write (again in the order of increasing process IDs). For each register $r$, the last process that writes to $r$ in this phase is called the *top-writer* of register $r$ in round $i$. If process $q$ accesses (reads or writes) register $r$ in the RMR Phase of the round $i$ and process $p \neq q$ is the top-writer of $r$ in round $i$, then we say that $p$ *covers* $q$ in round $i$.

### *Roll-Forward Phase*

After every process that participates in the RMR Phase has executed its (single) step in that phase, the Roll-Forward Phase starts. During this phase we keep track of a set of processes that are being "rolled forward". For every execution $E$ we define a *roll-forward set* $\mathcal{F}(E)$, which is a superset of the set $\mathcal{I}(E)$ of inactive processes. If $E'$ is the longest prefix of $E$ s.t. a process $p \in \mathcal{F}(E) - \mathcal{I}(E)$ is not in $\mathcal{F}(E') - \mathcal{I}(E')$, then we say that $p$ *gets rolled forward* in the first step in $E$ that follows $E'$. (Note that a process $p$ can be added to the roll-forward set because it becomes inactive, but in this case we don't say that $p$ gets rolled forward.) Suppose our schedule has generated an execution $E$ that ends during a Roll-Forward Phase. We now choose the process whose last step during the current Roll-Forward Phase is longest ago (giving preference to processes that have not taken any steps during the current Roll-Forward Phase). If multiple such processes exist, we choose among those the one with the smallest ID. Then we let that process take one step. This yields a longer execution $E'$ and a new roll-forward set $\mathcal{F}(E')$. We let $E = E'$ and if $\mathcal{F}(E) = \mathcal{I}(E)$, we repeat the above to determine a new process from the new set $\mathcal{F}(E) - \mathcal{I}(E)$, and we schedule that process next. This is repeated until we have obtained an execution $E$ such that $\mathcal{F}(E) = \mathcal{I}(E)$, i.e., all processes in the roll-forward set are inactive. When this happens, the Roll-Forward Phase ends.

To complete the description of the Roll-Forward Phase it remains to give the definition of $\mathcal{F}(E)$, which we do next. We say that process $p$ *finds* a process $q \neq p$ on register $r$ when one of the following two events happens:

1. $p$ accesses $r$ during an RMR Phase at a point when $q$ is still active, and either $r$ is in $q$'s local memory segment or $q$ was visible on $r$ at the beginning of the RMR Phase; or

2. $p$ accesses $r$ during a Roll-Forward Phase at a point when $q$ is not in the roll-forward set, and either $r$ is in $q$'s local memory segment or $q$ was visible on $r$ just before $p$ executed its operation on $r$.

(Note that when multiple processes $p_1, \ldots, p_\ell$ access the same register $r$ in an RMR phase and one of them finds a process $q$, then all processes $p_1, \ldots, p_\ell \neq q$ find $q$.)

We say that process $p$ *spoils* process $q$ on register $r$ in round $i$, if $p$ and $q$ both top-write in the RMR Phase of round $i$, $p$ top-writes to $r$, and $q$ had a valid cached copy of $r$ at the beginning of round $i$. We define the set $\mathcal{F}(E)$ by the following five rules:

(I0) If $p \in \mathcal{I}(E)$, then $p \in \mathcal{F}(E)$, i.e., all inactive processes are in the roll-forward set.

(F1) If process $p$ finds process $q \neq p$ during $E$, then both $p$ and $q$ are in $\mathcal{F}(E)$.

(F2) If process $p$ spoils processes $q_1, \ldots, q_t$ in some round during $E$, then $p$ and the process in $\{q_1, \ldots, q_t\}$ with the smallest ID are both in $\mathcal{F}(E)$.

(F3) If $p$ incurs at least $k$ RMRs in $E$, then $p \in \mathcal{F}(E)$.

(F4) If $p \in \mathcal{F}(E)$ and process $q$ covers $p$ in some round during $E$, then $q \in \mathcal{F}(E)$.

### *Local Step Phase and Preparation of the Next Round*

After the Roll-Forward Phase has ended, we finish the round by letting all active processes take steps as long as these steps do not incur RMRs. We do this in the order of process IDs, i.e, first we let the process with the smallest ID take steps until either it becomes inactive or its next step is poised to incur an RMR; then we do the same for the process with the second-smallest ID, and so on. This is the Local Step Phase of round $i$.

If we are not in the last round, i.e., $i < n^2$, then after the Local Step Phase is finished we prepare the next round by pushing appropriate sets of processes onto the stack. Let $A_i$ be the subset of processes in $G_i$ that are still active at the end of round $i$. We partition $A_i$ into three sets $G_{i,j}$, $j \in \{1, 2, 3\}$, as follows:

- $G_{i,1}$ is the subset of processes in $A_i$ that top-wrote to some green register $r$ during the RMR Phase of round $i$ (and thus are now visible on $r$);
- $G_{i,2}$ is the subset of processes in $A_i - G_{i,1}$ that took at least one step during round $i$; and
- $G_{i,3} = A_i - (G_{i,1} \cup G_{i,2})$.

Then we push onto the stack all of the sets $G_{i,3}$, $G_{i,2}$, and $G_{i,1}$ that are not empty (in this order), and proceed to round $i+1$.

If now $i = n^2$, our adversary becomes deterministic and it schedules all processes in $\mathcal{P}''$ that are still active simply in a round-robin fashion.

## 4.2 The RMR Lower Bound

An analysis of our randomized algorithm, in Sections 6 and 7, yields the following lower bound on the RMR complexity of deterministic one-time mutual exclusion algorithms scheduled by that adversary.

LEMMA 4.1. *Let $\mathcal{P}'$ be any subset of processes with $|\mathcal{P}'| \geq n/2$, and $L$ be any deterministic one-time mutual exclusion algorithm scheduled by randomized adversary $\mathcal{D}(\mathcal{P}')$. If all processes that take at least one step finish in a bounded expected number of steps, then the expected number of RMRs incurred is $\Omega(\varepsilon \log n / \log\log n)$.*

Recall that $\varepsilon = k^{-4} = \Theta\left((\log n/\log\log n)^4\right)$.

Using this lemma we can easily show the next result.

LEMMA 4.2. *Let $L$ be an arbitrary deterministic one-time mutual exclusion algorithm scheduled by $\mathcal{D}$. If all processes finish in a bounded expected number of steps, then the expected number of RMRs incurred is $\Omega(\log n/\log\log n)$.*

PROOF. Let $R_i$, for $1 \leq i \leq 1/\varepsilon$, be the number of RMRs incurred during the schedule by $\mathcal{D}(\mathcal{P}'_i)$. To prove the claim it suffices to show that $\mathrm{Exp}\left[\sum_{i=1}^{1/\varepsilon} R_i\right] = \Omega(\varepsilon \log n/\log\log n)$.

We have

$$\begin{aligned}\mathrm{Exp}\left[\sum_{i=1}^{1/\varepsilon} R_i\right] &= \sum_{i=1}^{1/\varepsilon} \mathrm{Exp}[R_i] \\ &\geq \sum_{i=1}^{1/\varepsilon} \mathrm{Exp}\left[R_i \,\middle|\, |\mathcal{P}'_i| \geq n/2\right] \cdot \mathrm{Prob}\left(|\mathcal{P}'_i| \geq n/2\right).\end{aligned}$$

By Lemma 4.1, $\mathrm{Exp}\left[R_i \,\middle|\, |\mathcal{P}'_i| \geq n/2\right] = \Omega(\varepsilon \log n/\log\log n)$. We now bound $\mathrm{Prob}(|\mathcal{P}'_i| \geq n/2)$. By a simple induction argument we obtain that $\mathrm{Exp}[|\mathcal{P}'_i|] = (1-\varepsilon)^{i-1}n$, and thus by Markov's inequality,

$$\begin{aligned}\mathrm{Prob}\left(|\mathcal{P}'_i| \leq n/2\right) &= \mathrm{Prob}\left(n - |\mathcal{P}'_i| \geq n/2\right) \\ &\leq \frac{n - \mathrm{Exp}[|\mathcal{P}'_i|]}{n/2} \\ &= 2 - 2(1-\varepsilon)^{i-1}.\end{aligned}$$

Thus, $\mathrm{Prob}(|\mathcal{P}'_i| \leq n/2) \geq 2(1-\varepsilon)^{i-1} - 1$.

Combining the above yields

$$\begin{aligned}\mathrm{Exp}\left[\sum_{i=1}^{1/\varepsilon} R_i\right] &= \Omega\left(\sum_{i=1}^{1/\varepsilon} \frac{\varepsilon \log n}{\log\log n} \cdot \left(2(1-\varepsilon)^{i-1} - 1\right)\right) \\ &= \Omega\left(\frac{\varepsilon \log n}{\log\log n} \cdot \left(2\sum_{i=1}^{1/\varepsilon}(1-\varepsilon)^{i-1} - 1/\varepsilon\right)\right).\end{aligned}$$

And since

$$2\sum_{i=1}^{1/\varepsilon}(1-\varepsilon)^{i-1} = 2\frac{1-(1-\varepsilon)^{1/\varepsilon}}{\varepsilon} \geq 2\frac{1-1/e}{\varepsilon} \geq 1.2/\varepsilon,$$

the claim follows. ∎

## 4.3 An Upper Bound on the Number of Steps.

The RMR bounds in Section 4.2, hold only for (deterministic) algorithms that finish in a bounded expected number of steps when scheduled by the randomized adversary. Below we show that any randomized algorithm that satisfies expected deadlock freedom is guaranteed to finish in a bounded expected number of steps, when scheduled by any deterministic adversary that is obtained by fixing the coin flips of the randomized adversary. We also argue that the randomized adversary corresponds to a distribution over a *finite* collection of deterministic adversaries.

LEMMA 4.3. *Let $\mathcal{G}$ be any randomized one-time mutual exclusion algorithm that satisfies expected deadlock freedom for step-bound $T$, and let $A$ be any deterministic adaptive adversary obtained by fixing the random choices of $\mathcal{D}$. In an execution of $\mathcal{G}$ scheduled by $A$, the total number of steps until all processes finish is upper-bounded by $18Tn^3$ with probability at least $1 - e^{-n^3}$.*

PROOF. Denote by $A(\mathcal{P}'_i)$, for $1 \leq i \leq 1/\varepsilon$, the part of adversary $A$ that corresponds to the part $\mathcal{D}(\mathcal{P}'_i)$ of $\mathcal{D}$. In every RMR Phase scheduled by $A(\mathcal{P}'_i)$, each process in $\mathcal{P}''_i$ takes at most one step, while the remaining processes take no steps. Thus, the total number of steps in all RMR Phases scheduled by $A$ is upper-bounded by $n^2 \cdot \sum_{i=1}^{1/\varepsilon} |\mathcal{P}''_i| \leq n^3$.

To bound the steps in the rest of the execution we use the following result, which we prove below.

CLAIM 4.4. *Let $E'$ be a (proper) prefix of an execution of $\mathcal{G}$ scheduled by $A$, such that the first step after $E'$ ends is not part of an RMR Phase. The expected number of steps from the end of $E'$ until any one of the following events occurs is at most $T$: (a) a process gets rolled forward; (b) a process becomes inactive; (c) a process is stopped because it is poised to incur an RMR in a Local Step Phase.*

Let $E$ denote the execution of $\mathcal{G}$ scheduled by $A$, and let $S$ be the subset of the steps in $E$ that are not executed during RMR Phases. We say that a process becomes inactive in step $t$, if it becomes inactive after executing step $t-1$. We will refer to the parts of the execution that are not regular phases (and in which processes are scheduled deterministically in a round-robin fashion) as Complementary Phases. Let $t_i$, for $i \geq 1$, be the $i$-th smallest step in $S$ in which one of the events (a)–(c) occurs, or some phase starts (clearly, this phase is not an RMR Phase, but it can be a Complementary Phase). Further, let $t'_i > t_i$ be the smallest step in which one of the events (a)–(c) occurs; this step may or may not belong to $S$. We denote by $i^*$ be the largest $i$ for which $t_i$ is defined. Finally, for each $1 \leq i \leq i^*$, we define $X_i = t'_i - t_i$.[2]

Clearly, $|S| = \sum_{i=1}^{i^*} X_i$, because at the end of every phase other than RMR Phases event (b) or (c) occurs. Now if $i \leq i^*$ and we have fixed the first $t_i - 1$ steps, we can apply Claim 4.4 to bound $X_i = t'_i - t_i$: From the claim

[2]Note that if $E$ is unbounded then $t'_{i^*} = \infty$, and thus $X_{i^*} = \infty$.

and Markov's inequality, it follows that $X_i$ is dominated by $2T \cdot Y_i$, where $Y_i$ is a geometrically distributed random variable with expectation 2. Further, we observe that $i^* \leq 2n + n^3$: Each of the $n$ process in $\mathcal{P}$ is rolled forward at most once, it becomes inactive exactly once, and it participates in at most $n^2$ Local Step Phases.

By combining the above we obtain that $|S| = \sum_{i=1}^{i^*} X_i$ is dominated by $2T \cdot \sum_{i=1}^{2n+n^3} Y_i$, where $Y_1, \ldots, Y_{2n+n^3}$ are independent geometrically distributed random variables with expectation 2. And standard Chernoff-bound arguments yield that $2T \cdot \sum_{i=1}^{2n+n^3} Y_i$ is upper-bounded by $16T(2n+n^3)$ with probability at least $1 - e^{-(2n+n^3)}$. Therefore, with this probability, the total number of steps is upper-bounded by $n^3 + 16T(2n + n^3)$.

To compete the proof it remains to show Claim 4.4.

*Proof of Claim 4.4*: If execution $E'$ ends during a Complementary Phase, then all active processes that have taken at least one step up to that point are scheduled in a round-robin fashion, and thus the claim follows from the definition of expected deadlock freedom.

Suppose now that $E'$ ends during a Roll-Forward or Local Step Phase scheduled by $\mathcal{D}(\mathcal{P}'_i)$. The following *invariant* holds for any execution $E''$ that ends during an RMR, Roll-Forward, or Local Step Phase scheduled by $\mathcal{D}(\mathcal{P}'_i)$: At the end of $E''$ no process $p$ knows of some other process $q$ that is not in the roll-forward set $\mathcal{F}(E'')$; or formally, process $p$ cannot distinguish execution $E''$ from execution $E'' \mid \mathcal{F}(E'') \cup \{p\}$, in which processes in $\mathcal{F}(E'') \cup \{p\}$ take steps in the same order as in $E''$, and the other processes take no steps. The invariant holds because of the roll-forward rule (F1), as we explain now. Rule (F1) ensures that no process $q \notin \mathcal{F}(E'')$ is *found* during $E''$, and this implies that no process ever accesses a register $r$ during a phase at the beginning of which $q$ was visible on $r$. It remains to show that no process $q \notin \mathcal{F}(E'')$ becomes visible on a register $r$ *after* the beginning of a phase and subsequently $r$ is *read* by a process $p$ during the *same* phase. This is true for an RMR Phase because all reads precede all writes in that phase. Also it holds for a Roll-Forward Phase because any process that takes steps during that phase is already in $\mathcal{F}(E'')$. Finally, it is true for a Local Step Phase because in order for a process $q$ to write to some register $r$ and for a process $p \neq q$ to subsequently read $r$ at least one RMR must occur, which is impossible since processes are stopped before they incur an RMR in a Local Step Phase.

We can now finish the proof of Claim 4.4. First we consider the case in which $E'$ ends during a Roll-Forward Phase. By definition, after $E'$ ends all active processes in $\mathcal{F}(E')$ are scheduled to take steps in a round-robin fashion until some of these processes becomes inactive or a new process gets rolled forward; denote by $X$ the number of these steps. By the invariant we showed above, the execution until that point (at which a process gets rolled forward or becomes inactive) is indistinguishable to any $p \in \mathcal{F}(E')$ from an execution in which only processes in $\mathcal{F}(E')$ take steps (in the same order). In the latter execution, by expected deadlock freedom, the number $Y$ of steps from configuration $C(E' \mid \mathcal{F}(E'))$ until some process in $\mathcal{F}(E')$ becomes inactive has an expectation of at most $T$. And since by the indistinguishability of the two executions, $Y \geq X$, the claim follows.

Consider now the case where $E'$ ends during a Local Step Phase. Throughout this phase the set of rolled-forward processes is the same as the set of inactive processes: All the roll-forward events described in (F1)–(F3) require that a process incurs an RMR; hence, none of these events happens in the Local Step Phase, and thus neither does the event in (F4). Suppose now that process $p$ is scheduled to take the next step after $E'$ ends. Then $p$ continues to take steps solo until it either becomes inactive or is stopped because it is about to incur an RMR; again let $X$ denote the number of these steps. By the invariant we showed earlier, from $p$'s point of view the execution up to that point is indistinguishable from the one where only the processes in $\mathcal{F}(E') = \mathcal{I}(E')$ and $p$ take steps, and, by expected deadlock freedom, in the latter execution the number $Y$ of steps from $C(E' \mid \mathcal{F}(E') \cup \{p\})$ until $p$ becomes inactive has an expectation of at most $T$. And since $Y \geq X$, the same bound holds for $X$. This completes the proof of Claim 4.4, and of Lemma 4.3. ∎

The randomized decisions that adversary $\mathcal{D}$ must make are that for each $1 \leq i \leq 1/\varepsilon$, (1) it must select the random subset $\mathcal{P}''_i$ of the processes in $\mathcal{P}'_i$ that are scheduled by $\mathcal{D}(\mathcal{P}'_i)$; and (2) it must determine the colours of the shared registers in each of the $n^2$ rounds schedule by $\mathcal{D}(\mathcal{P}'_i)$. At most $n$ independent coin flips are needed for (1), one coin flip for each process in $\mathcal{P}'_i$. But for (2) an infinite number of coin flips may be necessary to colour all registers, since $\mathcal{R}$ can be unbounded. Note, however, that in each round the adversary only needs to colour the registers that are accessed in the RMR Phase of that round, that is, at most $|\mathcal{P}''_i|$ registers. Therefore, the total number of coin flip that $\mathcal{D}$ executes is at most

$$n/\varepsilon + \sum_{i=1}^{1/\varepsilon} |\mathcal{P}''_i| \cdot n^2 \leq n/\varepsilon + n^3.$$

Further, these coin flips are governed by the same distribution, that is, one side has probability $\varepsilon$ and the other $1 - \varepsilon$. Therefore, a vector of $n/\varepsilon + n^3$ independent coin flips yields a unique deterministic adaptive adversary. The next observation now follows.

Observation 4.5. *Randomized adversary $\mathcal{D}$ can be described as a probability distribution over a collection of at most $2^{n^3+n/\varepsilon}$ deterministic adaptive adversaries.*

# 5. THE MAIN THEOREM

We are now ready to prove the main result of this paper.

Theorem 5.1. *For every randomized one-time mutual exclusion algorithm that satisfies expected deadlock freedom, there is an adaptive adversary that yields an execution where the total number of RMRs incurred has an expectation of $\Omega(n \log n / \log \log n)$.*

Proof. Fix a randomized one-time mutual exclusion algorithm $\mathcal{G}$ that satisfies expected deadlock freedom for step-bound $T$. We can view $\mathcal{G}$ as a probability distribution over a set $\mathcal{L}$ of deterministic algorithms obtained by fixing the coin-flip vector associated with each process. Since these vectors are infinite, $\mathcal{L}$ can be uncountably infinite. Consider now the randomized adaptive adversary $\mathcal{D}$ described in Section 4. We saw in Observation 4.5 that $\mathcal{D}$ can be viewed as a probability distribution over a set $\mathcal{A}$ of $2^{n^3+n/\varepsilon}$ deterministic adaptive adversaries.

For each deterministic adversary $A \in \mathcal{A}$, let $\mathcal{L}(A) \subseteq \mathcal{L}$ be the set of deterministic algorithms such that each algorithm in $\mathcal{L}(A)$ finishes in at most $\lambda = 18Tn^3$ steps when scheduled by $A$. From Lemma 4.3, for any $A \in \mathcal{A}$, the total number of steps in an execution of $\mathcal{G}$ scheduled by $A$ is bounded by $\lambda = 18Tn^3$ with probability at least $1-e^{-n^3} = 1-o(1/|\mathcal{A}|)$. It follows that, for any $A \in \mathcal{A}$, $\mathrm{Prob}_{L\in_{\mathrm{R}}\mathcal{G}}(L \in \mathcal{L}(A)) = 1 - o(1/|\mathcal{A}|)$. Define now

$$\mathcal{L}' = \bigcap_{A\in\mathcal{A}} \mathcal{L}(A).$$

Then every algorithm in $\mathcal{L}'$ finishes in at most $\lambda$ steps when scheduled by and adversary in $\mathcal{A}$, and by the union bound,

$$\mathrm{Prob}_{L\in_{\mathrm{R}}\mathcal{G}}(L \in \mathcal{L}') = 1 - o(1). \tag{5.1}$$

Let $T_{\mathrm{RMR}}(L, A)$ denote the total number of RMRs incurred in the execution of $L \in \mathcal{L}$ scheduled by $A \in \mathcal{A}$. By Lemma 4.2,

$$\min_{L\in\mathcal{L}'} \mathrm{Exp}_{A\in_{\mathrm{R}}\mathcal{D}}\big[T_{\mathrm{RMR}}(L, A)\big] = \Omega\left(\frac{n\log n}{\log\log n}\right). \tag{5.2}$$

We will now show using Yao's Principle that

$$\max_{A\in\mathcal{A}} \mathrm{Exp}_{L\in_{\mathrm{R}}\mathcal{G}}\big[T_{\mathrm{RMR}}(L, A) \,\big|\, L \in \mathcal{L}'\big] = \Omega\left(\frac{n\log n}{\log\log n}\right). \tag{5.3}$$

Recall that the set $\mathcal{A}$ is finite, but $\mathcal{L}'$ may be uncountably infinite (which prevents us from applying Yao's Principle). However, by definition, any algorithm $L \in \mathcal{L}'$ finishes after at most $\lambda$ steps when scheduled by any adversary $A \in \mathcal{A}$. Thus, instead of set $\mathcal{L}'$ we can consider the set $\mathcal{L}''$ of algorithms obtained if we modify each algorithm $L \in \mathcal{L}'$ by forcing every process to stop after it executes its $\lambda$-th step (if it has not finished earlier); we denote by $L(\lambda)$ this bounded version of $L$.[3] Clearly, $T_{\mathrm{RMR}}(L, A) = T_{\mathrm{RMR}}(L(\lambda), A)$ for all $L \in \mathcal{L}'$ and $A \in \mathcal{A}$, and thus (5.2) still holds if we replace $\mathcal{L}'$ by $\mathcal{L}''$. Further, we have that $\mathcal{L}''$ is finite: since for every algorithm in $\mathcal{L}''$ each processes takes at most $\lambda$ steps and thus it uses at most $\lambda$ coin flips, the number of distinct algorithms in $\mathcal{L}''$ is at most $|\Omega|^{n\lambda}$, where $\Omega$ is the domain of the coin flips. Therefore, by Yao's Principle, for any distribution $\mathcal{G}''$ over $\mathcal{L}''$,

$$\begin{aligned}\max_{A\in\mathcal{A}} \mathrm{Exp}_{L\in_{\mathrm{R}}\mathcal{G}''}\big[T_{\mathrm{RMR}}(L, A)\big] &\geq \min_{L\in\mathcal{L}''} \mathrm{Exp}_{A\in_{\mathrm{R}}\mathcal{D}}\big[T_{\mathrm{RMR}}(L, A)\big] \\ &= \Omega\left(\frac{n\log n}{\log\log n}\right).\end{aligned}$$

Equation (5.3) now follows by choosing $\mathcal{G}''$ to be the distribution of $L(\lambda)$, conditional on the event that $L \in \mathcal{L}'$, when $L$ is chosen at random from $\mathcal{L}$ according to $\mathcal{G}$.

Combining (5.1) and (5.3), yields the theorem. ∎

# 6. THE PROBABILITY OF FINDING PROCESSES

One of the main challenges in our analysis is to bound the probability that processes get rolled forward due to roll-forward rule (F1) i.e., because some process finds another one. In this section we show that in every step in which a process incurs an RMR it triggers that roll-forward event only with small probability.

As discussed earlier, we assume that processes are deterministic. Hence, an execution is uniquely determined by a schedule provided by the randomized adversary. Further, the schedule is uniquely determined by the colour (red or green) of each register in each of the first $n^2$ rounds. (Recall that after $n^2$ rounds, the adversary switches to a deterministic round-robin schedule.) A *colour schedule* is a binary matrix $(M_{i,r})_{1\leq i\leq n^2, r\in\mathcal{R}}$ that corresponds to the schedule constructed by our adversary: $M_{i,r} = 1$ if register $r$ is green in round $i$, and $M_{i,r} = 0$ if $r$ is red in round $i$. Let $E(M)$ be the execution resulting from the colour schedule $M$. We sometimes say $M_{i,r}$ is *green* (*red*), if $M_{i,r} = 1$ (resp., $M_{i,r} = 0$). We say that a colour schedule $M'$ *dominates* a colour schedule $M$, if $M'_{i,r} \geq M_{i,r}$ for all $(i, r) \in \{1, \ldots, n^2\} \times \mathcal{R}$. Recall that a register is green in round $i$ with probability $\varepsilon$.

In the remainder of this section we prove the following statement, which is the core of the analysis of the randomized adversary.

LEMMA 6.1. *Let $M$ be a random colour schedule. For a process $b$, a round number $j$, and an integer $\zeta \geq 0$, define the events $\mathcal{E}_{b,j,\zeta}$ and $\mathcal{E}^{find}_{b,j,\zeta}(M)$ as follows:*

- *If $\zeta = 0$, then $\mathcal{E}_{b,j,\zeta}(M)$ is the event that $b$ participates in the RMR Phase of round $j$ of $E(M)$, and $\mathcal{E}^{find}_{b,j,\zeta}(M)$ is the event that $b$ finds a process in that phase.*
- *If $\zeta > 0$, then $\mathcal{E}_{b,j,\zeta}(M)$ is the event that in the Roll-Forward Phase of round $j$ of $E(M)$ process $b$ incurs the $\zeta$-th RMR of that phase, and $\mathcal{E}^{find}_{b,j,\zeta}(M)$ is the event that $b$ finds a process when it incurs the $\zeta$-th RMR.*

*Then,*

$$\mathrm{Prob}\Big(\mathcal{E}^{find}_{b,j,\zeta}(M) \;\Big|\; \mathcal{E}_{b,j,\zeta}(M)\Big) \leq \frac{\varepsilon}{1-\varepsilon}\cdot k(k+2).$$

The idea is the following: Consider a colour schedule $M$ that results in an execution $E(M)$ in which a process $b$ finds some other active process $a$ in a step $s$ (where $s$ is the $\zeta$-th RMR process $b$ incurs in some round $j$). We flip some array entries of $M$ from green to red to obtain a new colour schedule $N$. In the resulting execution $D = E(N)$ process $b$ does not find any other process in step $s$. We have to be careful, though to not change $E$ too much. In particular we must ensure that in $D$ there are not any more roll-forward events than in $E$. In fact, the execution $D$ will be the same as $E$ for almost all processes.

We then evaluate the relative probability of the two colour schedules $M$ and $N$. Since we only flip some registers from green to red in order to obtain $N$ from $M$, colour schedule $N$ is significantly less likely than colour schedule $M$. However, we also have to take into account the number of distinct colour schedules $M'$ that get mapped to the same $N$ in order to avoid that process $b$ finds some other process in step $s$.

## 6.1 Preliminaries

We start with some observations regarding the roll-forward rules. Roll forward rule (F4) implies that there can be a chain of roll-forward events in some step: If process $a$ covers $b$ and $b$ covers $c$, then when $c$ gets rolled forward, $a$ and $b$ have to get rolled forward, too. Part (a) of the following claim implies that such chains have length at most

[3]Algorithm $L(\lambda)$ could suffer a deadlock if scheduled by an adversary that is not in $\mathcal{A}$, but we are only interested in schedules by the adversaries in $\mathcal{A}$.

$k+1$. In part (b) of the claim, we show that event (F3) cannot trigger event (F4). Part (c) states that if in the RMR Phase of some round a process gets rolled forward due to (F3), then all processes that participate in that RMR Phase get rolled forward because they execute their $k$-th RMR. Finally, in part (d) we show that at most one process may become inactive in a round without getting rolled forward.

CLAIM 6.2. *Let $E$ be an execution obtained by the randomized adversary.*

*(a) If $p_1, p_2, \ldots, p_\ell \in \overline{\mathcal{F}(E)}$, and for all $1 \leq j < \ell$ process $p_j$ covers $p_{j+1}$ at some point during $E$, then $\ell \leq k$.*

*(b) If $p, q \in \overline{\mathcal{F}(E)}$ and $E$ ends at the beginning of a round in which $p$ will get popped from the stack, then $q$ does not cover $p$ during $E$.*

*(c) If processes $p$ and $q$ both participate in round $i$, then in every round $j < i$ either both, $p$ and $q$, or none of them participated. (In particular, if a process incurs its $k$-th RMR in the RMR Phase of round $i$, then all processes that participate in this RMR Phase are rolled forward in the following Roll-Forward Phase.)*

*(d) In each round, at most one process becomes inactive among the processes that do not get rolled forward.*

PROOF. **(a):** For the purpose of a contradiction, suppose $\ell \geq k+1$. Let $i_1, \ldots, i_{\ell-1}$ be the round numbers, such that $p_j$ covers $p_{j+1}$ in round $i_j$, for all $1 \leq j < \ell$. We show that $p_\ell$ participates in all rounds $i_1, \ldots, i_{\ell-1}$. I.e., $p_\ell$ takes a step in the RMR phase of each of those rounds, and thus it incurs $\ell - 1 \geq k$ RMRs during $E$. Thus, by rule (F3) it is in $\mathcal{F}(E)$, contradicting the assumption.

Clearly, $p_\ell$ participates in round $i_{\ell-1}$, as it gets covered by $p_{\ell-1}$ in that round. Suppose $p_\ell$ does not participate in some round $i_j$, $1 \leq j \leq \ell-2$. Then either $p_\ell$ is still in a set on the stack when the set containing $p_j$ and $p_{j+1}$ was popped from the stack at the beginning of round $i_j$, or $p_\ell$ gets halted in round $i_j$. In either case, since $p_j$ is a top-writer in round $i_j$, process $p_\ell$ ends up in a set on the stack that is below $p_j$'s set. Hence, $p_\ell$ cannot participate again until it gets rolled forward, or until $p_j$ has become inactive. Neither of that happens during $E$ as $p_\ell \notin \mathcal{F}(E)$. It follows that $p_\ell$ does not participate in round $i_{\ell-1}$—a contradiction.

**(b):** Suppose $q$ does cover $p$ in some round $i$ of $E$. Then at the end of round $i$ process $q$ is added to set $G_{i,1}$ and $p$ to set $G_{i,2}$, and first $G_{i,2}$ and then $G_{i,1}$ are pushed on the stack. Thus, $p$ won't get popped from the stack until $q$ has become inactive.

**(c):** This is immediate from the definition of the Local Step Phase: If a set $S$ is pushed on the stack, then either all or none of the processes in $S$ participated in the previous RMR Phase. And processes can only participate in an RMR Phase, if their set is popped from the stack.

**(d):** Recall that in its critical section a process first reads a shared register $r_{crit}$, and then it writes its ID to this register. No process accesses $r_{crit}$ outside its critical section.

Assume by way of contradiction that two processes $p$ and $q$ that to not get rolled forward become inactive in round $i$, and assume (w.l.o.g.) that $p$ enters its critical section before $q$ does. First we argue that $p$ becomes inactive before $q$ enters its critical section. Let $p'$ be the first process that enters its critical section after $p$. When $p'$ takes its first step in the critical section, which is a read of register $r_{crit}$, $p$ must already be inactive, since otherwise $p'$ would find $p$ and $p$ would get rolled forward. Since either $q = p'$ or $q$ enters it critical section after $p'$, it follows that $p$ becomes inactive before $q$ enters its critical section.

Next we observe that $r_{crit}$ cannot be in $q$'s memory segment: if it were, then $p$ would have found $q$ when it accessed $r_{crit}$ in its critical section, and thus $p$ would have been rolled forward. Therefore, $q$ incurs two RMRs during its critical section: one for reading and one for writing register $r_{crit}$. And as we argued earlier, both RMRs are incurred after $p$ became inactive. Since we assumed that both $p$ and $q$ become inactive during round $i$, it follows that $q$ incurs (at least) two RMRs during this round. However, unless a process gets rolled forward, it incurs at most one RMR in each round (in the RMR Phase of the round). Therefore, $q$ gets rolled forward—a contradiction. ∎

## 6.2 Erasing Processes

Consider a colour schedule $M$ that yields an execution $E(M)$. Suppose in some round $j$ a process $b$ finds another process $a$ on register $r$. Then $a$ must have top-written in an earlier round $i$ to register $r$. The idea is to change the value of $M_{i,r}$ from 1 to 0, so that in the execution $E(M')$, resulting from the new colour schedule $M'$, process $b$ does not find process $a$ anymore. However, suppose that in $E(M)$ process $a$ top-wrote also in some other round $i < i' < j$. Then in the new execution, $E(M')$, $a$ is halted in round $i'$, so one of the processes that were covered by $a$ will now become a top-writer. This may change the execution $E(M')$ considerably, and lead to other processes getting rolled forward. In order to avoid this, we flip every matrix entry $M_{i',r}$ from green to red, if process $a$ top-writes in round $i'$ to register $r$. The resulting colour schedule is $N$. This way, in $E(N)$, all processes that were previously covered by $a$ in a round $i'$ will now get halted in round $i'$, and thus don't take any more steps until $a$ becomes inactive. Note that if they were not halted and be covered instead, then they would also not take any more steps until $a$ has become inactive.

For a colour schedule $M$, an integer $i \geq 1$, and a value $\lambda \in \mathbb{N} \cup \{0, \infty\}$, define $E(M, i, \lambda)$ to be the prefix of $E(M)$ that ends

- at the beginning of the RMR Phase of round $i$ if $\lambda = 0$,
- just before the $\lambda$-th step of the Roll-Forward Phase if the Roll-Forward Phase has at least $\lambda$ steps,
- at the end of the Roll-Forward Phase if the Roll-Forward Phase has fewer than $\lambda$ steps, and
- at the end of round $i$ (i.e., after the Local Step Phase) if $\lambda = \infty$.

For every process $p$ and every integer $i \in \{1, \ldots, n^2\}$ define $erase(M, p, i)$ to be the colour schedule $N$, where for $(\ell, r) \in \{1, \ldots, n^2\} \times \mathcal{R}$

$$N_{\ell,r} = \begin{cases} 0 & \text{if } \ell \geq i \text{ and process } p \text{ top-writes on register } r \text{ during round } \ell \text{ of } E(M), \text{ and} \\ M_{\ell,r} & \text{otherwise.} \end{cases}$$

Let $b$ be some process, $r \in \mathcal{R}$ some register, $\lambda \in \mathbb{N} \cup \{0, \infty\}$, and $1 \leq i \leq j \leq n^2$. We define a function $f_{j,\lambda,r,b}$

that maps a colour schedule $M$, a process $a$, and an integer $i$ to a new colour schedule as follows. If

(A1) process $a$ top-writes to register $r$ in round $i$ of $E(M)$,

(A2) $a \notin \mathcal{F}\big(E(M, j, \lambda)\big)$,

(A3) $b$ takes at least one step in round $j$ of $E(M)$, and

(A4) $a$ does not cover $b$ in round $j$ of $E(M, j, \lambda)$,

then $f_{j,\lambda,r,b}(M, a, i) = \mathit{erase}(M, a, i)$; otherwise $f_{j,\lambda,r,b}(M, a, i) = M$.

For the remainder of this section, we fix arbitrarily a process $b$, values $\lambda \in \mathbb{N} \cup \{0, \infty\}$ and $j \in \{1, \ldots, n^2\}$, and a register $r$. This uniquely determines a function $f_{j,\lambda,r,b}$.

Claim 6.3. *Let $a, a' \neq b$ be distinct processes, let $M, M', N$ be colour schedules that differ in at least one of the first $j$ rows, and let $i \in \{1, \ldots, j\}$. If*

$$N = f_{j,\lambda,r,b}(M, a, i) = f_{j,\lambda,r,b}(M', a', i),$$

*then*

*(a) $a = a'$ and*

*there is an index $i' \in \{1, \ldots, j\}$ s.t.*

*(b) $M$ and $M'$ are equal in rows $1, \ldots, i' - 1$,*

*(c) there is exactly one register $r' \in \mathcal{R}$ s.t. $M_{i',r'} \neq M'_{i',r'}$, and*

*(d) if $M_{i',r'} = 1$, then*

*(d1) process $a$ top-writes register $r'$ in round $i'$ of $E(M)$, and*

*(d2) $M'$ and $N$ are equal in rows $i', i' + 1, \ldots, j$,*

Proof. Since $M$, $M'$, and $N$ are distinct, $N = \mathit{erase}(M, a, i) = \mathit{erase}(M', a', i)$. Hence, by (A1), in round $i$ of $E(M)$ resp. $E(M')$ process $a$ resp. $a'$ top-writes register $r$. Then $M_{i,r} = M'_{i,r} = 1$ (or else no process top-writes in round $i$ on register $r$). From the definition of the function *erase*, $M_{i^*,r^*} = N_{i^*,r^*} = M'_{i^*,r^*}$ for all pairs $(i^*, r^*) \in \big(\{1, \ldots, i\} \times \mathcal{R}\big) - \{(i, r)\}$. Since in addition $M_{i,r} = M'_{i,r} = 1$, executions $E(M)$ and $E(M')$ are identical during the first $i$ rounds. As only one process can top-write register $r$ in the $i$-th round of these executions, we have $a = a'$. This proves (a).

Now let $(i', r')$ be the first pair in a lexicographical ordering, where $M_{i',r'} \neq M'_{i',r'}$. By the assumption that $M$ and $M'$ differ in one of the first $j$ rows we have $i' \in \{1, \ldots, j\}$. Moreover, (b) follows immediately from the choice of $i'$. W.l.o.g. $M_{i',r'} = 1$ and $M'_{i',r'} = 0$. By definition of the function *erase*, $M$ and $M'$ dominate $N$, so $N_{i',r'} = 0$. Hence, again from the definition of *erase*, in round $i'$ of $E(M)$ process $a$ top-writes to register $r'$. This proves (d1). But the first $i' - 1$ rounds of $E(M)$ and $E(M')$ are identical, so at the beginning of round $i'$ of $E(M')$ process $a$ is poised to execute a step on register $r'$. It follows from the definition of *erase* that $M'$ and $N$ are equal in row $i'$ and $M$ and $N$ are equal in all entries of that row except the one of column $r'$. Hence, (c) follows.

It remains to show (d2). If $i' = j$, then this follows immediately from the already established fact that $M'$ and $N$ are equal in row $i'$. Hence, from now on assume $i' < j$.

Let $D = E(M, j, \lambda)$. We claim that there is a process $c \neq a$ that participates in the RMR Phase of round $j$ of $D$ and is not covered by $a$ at any time during $D$: If $b$ participates in round $j$, then it gets popped from the stack in that round, so by Claim 6.2 (b) and (A2) $a$ does not cover $b$ in execution $D$ prior to round $j$. By (A4) $a$ does not cover $b$ in round $j$ of execution $D$ either. Hence, if $b$ participates in round $j$, then we can choose $c = b$. Otherwise, $b$ can only take steps in round $j$ after it has been rolled forward. Hence, $b \in \mathcal{F}(D)$. In this case $b$ cannot have been rolled forward because of rule (F3), because that would imply that $b$ had incurred an RMR in round $j$ before it got rolled forward. From rules (F1), (F2), and (F4), there must be at least one other process $c \neq b$ that participates in the RMR Phase of round $j$ and that causes a first process to get rolled forward in that round. But then this process, $c$, gets rolled forward, too, so $c \in \mathcal{F}(D)$. Since $a \notin \mathcal{F}(D)$, we have $c \neq a$, and moreover due to roll-forward rule (F4), $a$ never covered $c$ during $D$. By (d1) process $a$ top-writes register $r'$ in round $i'$ of $E(M)$. Then $c$ does not access register $r'$ in the round-$i'$ RMR Phase of $D$ (if $c$ wrote $r'$ in round $i'$, then $a$ would cover $c$, and if $c$ read $r'$, then $a$ would be rolled forward due to (F1)).

Due to (b) and (c), during the first $i'$ rounds of $E(M')$ process $c$ performs exactly the same steps as in $E(M)$. Now recall that process $a$ top-writes in round $i'$ of $E(M)$. Since in $E(M)$ process $c$ participates in round $j > i'$ at a point when $a$ is still active, it follows that in round $i'$ process $c$ must have top-written, too. (Otherwise, at the end of round $i'$ of $E(M)$ process $c$ would land on the stack in a set below $a$'s set and it would never get popped from the stack before $a$ has become inactive, contradicting that $c$ participates in round $j$ of $E(M)$.) Hence, in round $i'$ of $E(M)$ and thus also in $E(M')$ process $c$ lands in a set on top of the stack. Since $a$ gets halted in round $i'$ of $E(M')$, $a$ does not end up in the same set on the stack as $c$, and in particular it lands in a set below $c$'s. Thus, the only way that $a$ can take a step in a later round of $E(M')$ before $c$ is finished is if $a$ gets rolled forward, but by the assumption (A2) $a \notin \mathcal{F}(E(M', j, \lambda))$. It follows that in $E(M')$ process $a$ does not take any steps and in particular does not top-write in any of the rounds $i', i' + 1, \ldots, j$. Thus, from the definition of function *erase*, it follows that $M'$ equals $N$ in rows $i', i' + 1, \ldots, j$, which completes the proof of (d2). ∎

The next lemma bounds the number of distinct colour schedules that are mapped to the same schedule via the above function. Let $j$, $\lambda$, $r$, and $b$ be fixed arbitrarily as above.

Lemma 6.4. *For every colour schedule $N$, and all integers $i \in \{0, \ldots, j\}$ there are no $k + 2$ pairs $(M_1, a_1), \ldots, (M_{k+1}, a_{k+2})$ such that $M_1, \ldots, M_{k+2}$ are all distinct in their first $j$ rows, and*

$$f_{j,\lambda,r,b}(M_s, a_s, i) = N \quad \textit{for all } 1 \leq s \leq k + 1.$$

Proof. For the purpose of a contradiction, suppose there are pairs $(M_1, a_1), \ldots, (M_{k+2}, a_{k+2})$ as described in the statement of the lemma. W.l.o.g. assume that $N \notin \{M_1, \ldots, M_{k+1}\}$ (i.e., possibly but not necessarily $N = M_{k+2}$). Then we get $a := a_1 = \cdots = a_{k+1}$ from Claim 6.3 (a).

Consider arbitrary $M, M' \in \{M_1, \ldots, M_{k+1}\}$, and let $i'$ be the index in $\{1, \ldots, j\}$ that satisfies Claim 6.3, and in particular $M_{i',r'} = 1$ and $M'_{i',r'} = 0$ for some register $r' \in \mathcal{R}$.

From part (b) of Claim 6.3 it follows that $M$ dominates $M'$ in rows $1, \ldots, i'-1$. By the definition of *erase*, $M$ dominates $N$. Hence, from part (d2) it follows that $M$ dominates $M'$ in rows $i', \ldots, j$. Thus, for any two matrices in $\{M_1, \ldots, M_{k+1}\}$ it holds that one dominates the other in rows $1, \ldots, j$. Since the dominance relation is transitive, it follows that we can relabel the indices of $M_1, \ldots, M_{k+1}$ such that $M_{s+1}$ dominates $M_s$ for $1 \le s \le k$.

Now for every $s \in \{1, \ldots, k\}$ let $i_s \in \{1, \ldots, j\}$ be the index such that $i' = i_s$ satisfies Claim 6.3 for the matrices $M, M' \in \{M_s, M_{s+1}\}$. Since $M_{s+1}$ dominates $M_s$ we get from parts (b), (c) and (d2) of that claim that $M_{s+1}$ and $M_s$ are equal in rows $1, \ldots, i_s - 1$ and differ in row $i_s$, and $M_s$ equals $N$ in rows $i_s, \ldots, j$. Hence, $i_1 < i_2 < \cdots < i_k$. A simple induction on $s = 1, \ldots, k$, shows that process $a$ top-writes in rounds $i_1, i_2, \ldots, i_s$ of execution $E(M_{s+1})$: For $s = 1$ this follows immediately from Claim 6.3 (d1) for $M = M_2$ and $M' = M_1$. Now suppose in execution $E(M_s)$ process $a$ top-writes in rounds $i_1, \ldots, i_{s-1}$. Since $M_s$ equals $M_{s+1}$ in rows $i_1, \ldots, i_s - 1$ and $i_1, \ldots, i_{s-1} < i_s$, process $a$ also top-writes in rounds $i_1, \ldots, i_{s-1}$ of execution $E(M_{s+1})$. From Claim 6.3 (d1) for $M = M_{s+1}$ and $M' = M_s$ it follows that in addition $a$ top-writes in round $i_s$ of $E(M_{s+1})$.

Hence, we have shown that $a$ top-writes in rounds $i_1, \ldots, i_k$ of execution $E(M_{k+1})$. Hence, in its top-write in round $i_k$, $a$ incurs its $k$-th RMR during $E(M_{k+1})$, so $a$ gets rolled forward immediately after. Since $i_k \le j$ we have $a \in \mathcal{F}\big(E(M_{k+1}, j, \lambda)\big)$. But then $M_{k+1} = N$ by the definition of function $f_{i,j,\lambda,r,a,b}$, which contradicts the assumption that $N \notin \{M_1, \ldots, M_{k+1}\}$. ∎

## 6.3 The Erasing Lemma

For an execution $E$ and an integer $i \ge 1$ let $RMR_i(E)$, $RF_i(E)$, and $LS_i(E)$ denote the sub-executions of $E$ that comprise the round-$i$ RMR, Roll-Forward, and Local Step Phase, respectively. executions can be empty, e.g., if $E$ has fewer than $i$ rounds, or if it $E$ ends before the corresponding phase in round $i$ has started. For a register $r$, let $rmr_{i,r}(E)$ denote the set of processes that access register $r$ during $RMR_i(E)$. Further, let $partcpt_i(E)$ denote the set of processes that participate in the round-$i$ RMR Phase of execution $E$, i.e., the processes that take at least one step in $RMR_i(E)$.

Let $state(p, E)$ denote the state of process $p$ at the end of execution $E$. We assume w.l.o.g. that processes record in their state how many steps they have taken, and the sequence of responses from all previous operations. Hence, if $E|p \ne E'|p$, then $state(p, E) \ne state(E', p)$. We also assume w.l.o.g. that processes know and record in their state, whether and when one of their own steps incurred an RMR, and the round numbers in which they took steps. Let $val(r, E)$ denote the value of register $r$ at the end of execution $E$, and let $local(p, E)$ denote the set of registers which are local to $p$ at the end of $E$ (i.e., they are either in $p$'s local memory segment, or $p$ has valid cached copies of them). We write $owner(r)$ to denote the process in whose memory segment a register $r$ is located. Further, let $topw(p, E)$ denote the set of registers which process $p$ top-writes during execution $E$, and $cov(p, E)$ is the set containing $p$ and all processes that get covered by $p$ during $E$.

If $E$ is the empty execution or $E$ ends at the end of a Local Step Phase, then $stack(j, E)$ denotes the set of processes which at the end of $E$ is in the $j$-th level of the stack, where $stack(0, E)$ is the topmost set. For any other execution $E$, $stack(j, E) = stack(j, E')$, where $E'$ is the longest prefix of $E$ that is either empty or ends immediately after a Local Step Phase.

The next lemma is at the core of our analysis and lists the properties of the *erase* function.

LEMMA 6.5. *Let $M$ be some colour schedule and $N = erase(M, a, t_{min})$ for some process $a$ and some integer $t_{min}$. Further, let $i \ge 1$, $\lambda \in \mathbb{N} \cup \{0, \infty\}$, $E' = E(M, i, \lambda)$, and $D' = E(N, i, \lambda)$. If $a \notin \mathcal{F}(E')$, then all of the following are true:*

*(a)* $\forall p \in \overline{cov(a, E')} : \ state(p, E') = state(p, D')$.

*(b)* $\forall r \in \overline{topw(a, E')}$ : *Either* $val(r, E') = val(r, D')$ *or* $owner(r) \in cov(a, E')$.

*(c)* $\forall p \in \overline{cov(a, E')} : \ local(p, E') \cap \overline{topw(a, E')} = local(p, D') \cap \overline{topw(a, E')}$.

*(d)* $\forall p \in \mathcal{P} - \{a\}, j \in \{1, \ldots, i\} : \ topw\big(p, RMR_j(E')\big) = topw\big(p, RMR_j(D')\big)$.

*(e)* $\mathcal{F}(E') = \mathcal{F}(D') \subseteq \overline{cov(a, E')}$, *and* $RF_j(E') = RF_j(D')$ *for all* $j \in \mathbb{N}$.

*(f) Suppose $\lambda = \infty$. If $i < t_1$, then the stacks are identical. Otherwise, there is some index $j^*$ such that*

$a \in stack(j^*, E')$ *and*

$\forall j \in \{0, \ldots, j^* - 1\} : \ stack(j, D') = stack(j, E') - \{a\}$.

In order to prove the lemma, we define $E = E(M)$ and $D = E(N)$. Hence, $E'$ and $D'$ are prefixes of $E$ and $D$, respectively. Let $t_1, \ldots, t_\ell$ be the rounds of $E$ in which process $a$ top-writes, and let $r_j$, $1 \le j \le \ell$, be the register to which $a$ top-writes in round $t_j$. We start with the following claim.

CLAIM 6.6. *If $a \notin \mathcal{F}(E')$, then for $j \in \{1, \ldots, \ell\}$, then all of the following is true:*

*(a)* $cov(a, E') \cap \mathcal{F}(E') = \emptyset$.

*(b) In $E'$ no process in $\mathcal{P} - \{a\}$ accesses register $r_j$ after the round-$t_j$ RMR Phase.*

*(c) In $E'$ no process $p \in cov\big(a, RMR_{t_j}(E)\big) - \{a\}$ executes any steps in the round-$t_j$ Roll-Forward Phase, nor does it get popped from the stack or execute any more steps after round $t_j$.*

*(d) In $E'$ no process $p$ accesses a register $r$, if $owner(r) \ne p$ was covered by $a$ in an earlier step.*

PROOF. **(a):** For the purpose of a contradiction, assume there is a process $p \in cov_j \cap \mathcal{F}(E')$. Then $p \ne a$ because $a \notin \mathcal{F}(E')$. Hence, by the definition of $cov_j$, $p$ gets covered by process $a$ in round $t_j$ of $E'$. But then $p \in \mathcal{F}(E')$ together with roll-forward rule (F4) implies $a \in \mathcal{F}(E')$—a contradiction.

**(b):** Assume the claim is not true. Let $p \ne a$ be the first process to accesses $r_j$ after the round-$t_j$ RMR Phase of $E'$. Since process $a$ top-wrote $r_j$ in that phase, $a$ is still visible on $r_j$ when $p$ accesses it. Hence, $p$ finds $a$ and thus by (F1) $a \in \mathcal{F}(E')$—a contradiction.

**(c):** Let $p \in cov\big(a, RMR_{t_j}(E)\big) - \{a\}$. By part (a), $p, a \in \overline{\mathcal{F}(E')}$. In round $t_j$ process $p$ gets covered by process $a$, so it does not execute any more steps during that round. By Claim 6.2 (b) $p$ does not get popped from the stack after round $t_j$, and since $p$ does not get rolled forward during $E'$, it cannot take any steps in a round where it doesn't get popped from the stack.

**(d):** Suppose $p$ accesses register $r$ and $q := owner(r) \neq p$. Then $p$ finds $q$, so $q \in \mathcal{F}(E')$. By part (a), $q \notin cov(a, E')$. ∎

Proof of Lemma 6.5. We prove the lemma by induction on the length of $E'$, i.e., for increasing $i$ and $\lambda$. First suppose $i < t_1$ or $i = t_1$ and $\lambda = 0$, i.e, $E'$ ends at the beginning of round $t_1$ or earlier: Since $a$ does not top-write in any round $j$, where $t_{min} \leq j < t_1$, it follows from the definition of *erase* that $M$ equals $erase(M, a, t_{min})$ in the first $t_1 - 1$ rows. Hence, the executions $D'$ and $E'$ are identical.

Now assume that $E'$ ends some time during round $i \geq t_1$. We consider three cases, first that $E'$ ends right after the RMR Phase ($\lambda = 1$), second that $E'$ ends at a later point during the Roll-Forward Phase ($2 \leq \lambda < \infty$), and third that $E'$ ends during the Local Step Phase ($\lambda = \infty$).

**Case 1: $\lambda = 1$.** Hence, $E'$ and $D'$ end immediately after the RMR Phase of round $i$. Assume that the induction hypothesis holds for the prefixes $E''$ and $D''$ of $E$ and $D$, resp., which end at the end of round $i-1$. Then $E' = E'' \circ RMR_i(E')$ and $D' = D'' \circ RMR_i(D')$, Let $G_i(E)$ and $G_i(D)$ be the sets of active processes that are popped from the stack at the beginning of round $i$ in $E$ and in $D$ (and thus in $E'$ and $D'$), respectively. From the induction hypothesis (f) for $E''$ and $D''$ and Claim 6.6 (c) we know that

$$G_i(E) = G_i(D) \wedge cov(a, E'') \cap G_i(E) = \emptyset \text{ if } i = t_1, \text{ and}$$
$$G_i(E) - \{a\} = G_i(D) \wedge cov(a, E'') \cap G_i(E) \subseteq \{a\} \text{ if } i > t_1. \tag{6.1}$$

Consider a register $r \in topw(a, E'')$. By Claim 6.6 (b) no process except possibly $a$ will access register $r$ during $RMR_i(E)$, so

$$\forall r \in topw(a, E'') : \ rmr_{i,r}(E) \subseteq \{a\}. \tag{6.2}$$

If $r$ is red in round $i$ of $E$, then by definition of *erase*, $r$ is also red in round $i$ of $D$, so $rmr_{i,r}(D) = \emptyset$. Now suppose that $r$ is green in round $i$ of $E$. Then (6.2) implies that no process in $G_i(E) - \{a\}$ is poised to access $r$ at the end of $E''$ (or else it would access $r$ during the round-$i$ RMR Phase of $E$). By (6.1), $G_i(E) - \{a\} \subseteq \overline{cov(a, E'')}$, and by part (a) of the induction hypothesis applied to $E''$ and $D''$ every process in $\overline{cov(a, E'')}$ is in the same state at the end of $E''$ as at the end of $D''$. It follows that also at the end of $D''$ no process in $G_i(E) - \{a\}$ is poised to access register $r$. Since $G_i(E) - \{a\} = G_i(D)$ (by (6.1)), we get

$$\forall r \in topw(a, E'') : \ rmr_{i,r}(D) = \emptyset. \tag{6.3}$$

Now consider a register $r \in topw(a, RMR_i(E))$. By construction a process executes an operation on $r$ during the round-$i$ RMR Phase of $E'$ if and only if it is either $a$ or it gets covered by $a$ during that phase. Moreover, by the definition of *erase*, $N_{i,r} = 0$ so in $RMR_i(D)$ no process accesses $r$. Thus,

$$\forall r \in topw(a, RMR_i(E)) :$$
$$rmr_{i,r}(E) = cov(a, RMR_i(a, E)) \subseteq cov(a, E') \text{ and}$$
$$rmr_{i,r}(D) = \emptyset. \tag{6.4}$$

Now consider a register $r \in \overline{topw(a, RMR_i(E))}$. If $r$ is red in round $i$ of $E$, then $r$ is red in round $i$ of $D$, too, and so $RMR_i(E) = RMR_i(D) = \emptyset$. Now suppose $r$ is green in round $i$ of $E$. Since $r \notin topw(a, RMR_i(E))$, by definition of *erase* register $r$ is also green in round $i$ of $D$. Then $rmr_{i,r}(E)$ and $rmr_{i,r}(D)$ are the sets of processes in $G_i(E)$ and $G_i(D)$, respectively, which are poised to access $r$ at the end of execution $E''$ and $D''$, respectively. From (6.1) none of the processes in $G_i(D) = G_i(E) - \{a\}$ is in $cov(a, E'')$. Thus, from part (a) of the induction hypothesis all of the processes in $G_i(D) = G_i(E) - \{a\}$ are in the same state at the end of $E''$ as at the end of $D''$, so if a process other than $a$ is poised to access $r$ at the end of $E''$ it is also poised to access $r$ at the end of $D''$. Thus, $rmr_{i,r}(E) - \{a\} = rmr_{i,r}(D)$. Moreover, since $r \notin topw(a, RMR_i(E))$, no process gets covered by $a$ on $r$ during $RMR_i(E)$. Since none of the processors participating in round $i$ of $E$ is in $cov(a, E'')$, we have

$$\forall r \in \overline{topw(a, RMR_i(E))} :$$
$$rmr_{i,r}(E) - \{a\} = rmr_{i,r}(D) = rmr_{i,r}(E) \cap \overline{cov(a, E')}. \tag{6.5}$$

Note that $partcpt_i(E')$ is the union of all $rmr_{i,r}(E')$ for all $r \in \mathcal{R}$. Hence, (6.4) and (6.5) imply that

$$partcpt_i(E) \cap \overline{cov(a, E')} = partcpt_i(D). \tag{6.6}$$

In the following we show for every register $r \in \mathcal{R}$ that

$$\forall p \in \mathcal{P} - \{a\} :$$
$$p \text{ top-writes to } r \text{ during } RMR_i(E) \text{ if and only if}$$
$$p \text{ top-writes to } r \text{ in } RMR_i(D). \tag{6.7}$$

Moreover, if $r \in \overline{topw(a, E')}$, then

$$val(r, E') = val(r, D'), \tag{6.8}$$

and

$$\forall p \in rmr_{i,r}(D) :$$
$$state(p, E') = state(p, D') \wedge local(p, E') = local(p, D'). \tag{6.9}$$

Then together with the induction hypothesis, (6.7) implies (d), and (6.8) implies (b). Moreover, if $p \in \overline{cov(a, E')}$ then (6.4) and (6.5) imply that either $p$ participates neither during $rmr_i(E')$ nor during $RMR_i(D')$, or, in both executions it accesses the same register $r$. In the latter case, by (6.4) that register $r$ is not in $topw(a, E')$, so (6.9) applies. In either case, parts (a) and (c) of follow.

We prove (6.7)-(6.9) for some fixed register $r \in \mathcal{R}$: If $M_{i,r} = N_{i,r} = 0$, no process executes any steps on $r$ during $RMR_i(E)$ or $RMR_i(D)$, so (6.7)-(6.9) are trivially true. Since $M$ dominates $N$, $N_{i,r} = 1$ and $M_{i,r} = 0$ is not possible. If $M_{i,r} = 1$ and $N_{i,r} = 0$, then by construction of $N$, process $a$ is the (only) top-writer during $RMR_i(E)$ on register $r$, but no process accesses $r$ during $RMR_i(D)$, so (6.7) is true. Moreover, then $r \in topw(a, E')$, so there is nothing to show for (6.8) and (6.9).

Now assume $M_{i,r} = N_{i,r} = 1$. If $a \notin rmr_{i,r}(E)$, then by (6.5) $R := rmr_{i,r}(E) = rmr_{i,r}(D)$ and no process in $R$ is in $cov(a, E'')$. Hence, due to part (a) of the induction hypothesis all processes invoke exactly the same operation on $r$ in $RMR_i(E')$ as in $RMR_i(D')$. The order of all operations on $r$ is uniquely determined by the set of processes executing an operation on that register, so this is the same for $RMR_i(E')$ and $RMR_i(D')$. By part (b) of Claim 6.6 $r \notin topw(a, E'')$, and by part (d) $owner(r) \notin cov(a, E'')$. Thus, by part (b) of the induction hypothesis, $r$ has the same value at the end of $E''$ as at the end of $D''$. Hence, the sequence of operations (including responses) executed on $r$ in $RMR_i(E')$ is exactly the same as in $RMR_i(D')$, and so (6.7)-(6.9) follow.

Now assume $a \in rmr_{i,r}(E)$. By the assumption that $N_{i,r} = 1$ and the construction of $N$, process $a$ is not a top-writer in round $i$. Hence, by (6.5) $rmr_{i,r}(E) = rmr_{i,r}(D) \cup \{a\}$ and no process in $rmr_{i,r}(E) - \{a\}$ is in $cov(a, E')$. If no process writes $r$ during $RMR_i(E)$, then $a$ also reads and does not change the value of register $r$. In this case, (6.7)-(6.9) follow immediately. If some process writes $r$ during $RMR_i(E)$, then the process $p$ that top-writes $r$ is not $a$, as otherwise $N_{i,r} = 0$ by the definition of *erase*. Moreover, either $a$ reads during $RMR_i(E)$, or, if it writes, its write-operation gets immediately overwritten. All other processes in $rmr_{i,r}(E) - \{a\}$ execute exactly the same operations in the same order during $RMR_i(E)$ as during $RMR_i(D)$. Hence, (6.7)-(6.9) follow.

This completes the proof of (a)-(d).

We now prove (e). In particular, we show that if a process $p$ gets rolled forward in $RMR_i(E)$, then and only then it gets rolled forward in $RMR_i(D)$. This establishes $\mathcal{F}(E') - \mathcal{F}(E'') = \mathcal{F}(D') - \mathcal{F}(D'')$ and thus from part (e) of the induction hypothesis we get $\mathcal{F}(E') = \mathcal{F}(D')$. By Claim 6.6 (a), $cov(a, E') \cap \mathcal{F}(E') = \emptyset$, so (e) follows.

We informally say an event (F$i$), $1 \le i \le 4$, occurs, if there is a process $p$ that gets rolled forward because of rule (F$i$).

Suppose (F1) occurs during $RMR_i(E)$ or $RMR_i(D)$ because some process $p$ finds a process $q \neq p$ on a register $r$. Then $a \notin \{q, p\}$: In $RMR_i(D)$ process $a$ takes no steps, and in $RMR_i(E)$ process $a$ cannot be found or find another process, because $a \notin \mathcal{F}(E')$. Therefore, $r \notin topw(a, E'')$, because by (6.2) only process $a$ accesses such a register in $RMR_i(E')$, and by (6.2) no process accesses such a register in $RMR_i(D')$. Hence, by the induction hypothesis, the same process is visible on $r$ at the end of $E''$ as at the end of $D''$. Thus, if a process gets found in both, $RMR_i(E)$ and in $RMR_i(D)$, then it must be the same process $q$ in both executions. If $r \in topw(a, RMR_i(E))$, then by (6.4) $rmr_{i,r}(E) \subseteq cov(a, E')$, so a process in that set finds $q$ which contradicts Claim 6.6 (a). If $r \in \overline{topw(a, RMR_i(E))}$ then we get from (6.5) that $rmr_{i,r}(E) = rmr_{i,r}(D)$ as $a \notin rmr_{i,r}(E)$ (because otherwise $a$ would find $q$). Hence, exactly the same set of processes find $q$ in $RMR_i(E)$ as in $RMR_i(D)$.

Now we consider (F2). Suppose some process $p$ spoils some process $q$ on a register $r$ either during $RMR_i(E)$ or during $RMR_i(D)$. Then by Claim 6.6 (a), $p, q \notin cov(a, E')$, and in particular $p, q \neq a$. From (6.2) and (6.3) it follows that no process can spoil another process on a register $r \in topw(a, E'')$ during $RMR_i(E')$ or during $RMR_i(D')$. If $r \in topw(a, RMR_i(E))$, then process $a$ top-writes to $r$ in $RMR_i(E)$, so no process spoils another process on $r$ during $RMR_i(E)$. Moreover, in this case register $r$ is red in round $i$ of $D$, so no process spoils another process on $r$ during $RMR_I(D)$. Hence, $r \notin topw(a, E')$. Since $p, q \notin cov(a, E')$, then from induction hypothesis (c), $q$ has a valid cached copy of $r$ at the beginning of round $i$ of both executions, $D'$ and $E'$, and as established earlier, each of $p$ and $q$ executes in $RMR_i(D')$ exactly the same step as in $RMR_i(E')$. It follows that $p$ spoils $q$ in both executions.

Next consider (F3). From Claim 6.2 (c), if one process gets rolled forward in $RMR_i(E)$ or in $RMR_i(D)$ because it has incurred $k$ RMRs, then all processes in $partcpt_i(E)$ and in $partcpt_i(D)$, respectively, execute their $k$-th RMR in $RMR_i(E)$ and $RMR_i(D)$, respectively. From part (a) of the induction hypothesis, all processes $p \in \overline{cov(a, E')}$ have incurred equally many RMRs during $E''$ as during $D''$. According to (6.6), $partcpt_i(E) \cap \overline{cov(a, E')} = partcpt_i(D)$. Thus, if some process in $partcpt_i(E)$ gets rolled forward during $RMR_i(E)$ due to (F3), then that same process gets also rolled forward during $RMR_i(D)$. Now suppose that a process $p \in partcpt_i(D)$ executes its $k$-th RMR on some register $r$ during $RMR_i(D)$ but does not do so during $partcpt_i(E)$. Thus, $p \in rmr_{i,r}(D)$. Then $p \in cov(a, E')$. From (6.1), $p \notin cov(a, E'')$ since $p \in partcpt_i(D) \subseteq G_i(D)$. Hence, $p \in cov(a, RMR_i(E))$. But by part (a) of the induction hypothesis, at the beginning of round $i$ of $E$ process $p$ is poised to access register $r$. Hence, $p \in topw(a, RMR_i(E))$, so by (6.4) $rmr_{i,r}(D) = \emptyset$—a contradiction.

Finally, we consider (F4). This event occurs only if a process $p$ gets rolled forward during $RMR_i(E')$ or $RMR_i(D')$ and that triggers some other process $q$ to get rolled forward (i.e., it causes $q \in \mathcal{F}(E')$). There may be a chain of such (F4) events, i.e., a sequence of processes $p_1, \ldots, p_\ell$ may get rolled forward and $p_s$ triggers event (F4) for $p_{s+1}$. But the first process, $p_1$ gets rolled forward because of rules (F1)-(F3). Then from the induction hypothesis and as already established, $p_1 \in \mathcal{F}(E')$, $p_1 \in \mathcal{F}(D')$, and $p_1 \notin cov(a, E')$; this is the base case for an induction on $s$, where we show $p_s \in \mathcal{F}(E') \cap \mathcal{F}(D')$. For the inductive step, note that $p_{s+1}$, $1 \le s < \ell$, covers $p_s$ on a register $r$ in some round $j \in \{1, \ldots, i\}$ during $E'$ or $D'$. Since $p_{s+1} \notin \mathcal{F}(E')$, we have $p_{s+1} \neq a$. Thus, by part (d) of the induction hypothesis, $p_{s+1}$ top-writes in round $j$ of $E'$ and in round $j$ of $D'$ to the same register. Since $p_s \in \mathcal{F}(E')$ we have $p_s \notin cov(a, E')$ (by claim 6.6 (a)). Hence, from part (a) of the induction hypothesis it follows that $p_s$ executes the same steps in round $j$ of $E'$ as in round $j$ of $D'$. Thus, $p_{s+1}$ covers $p_s$ in round $j$ of both executions. Since $p_s \in \mathcal{F}(E') \cap \mathcal{F}(D')$ rule (F4) implies that $p_{s+1} \in \mathcal{F}(E') \cap \mathcal{F}(D')$. This completes the proof of (e).

Since $E'$ does not end at the end of a Local Step Phase, part (f) follows trivially for $E'$.

**Case 2: $2 \le \lambda < \infty$.** We argue that

$$RF_i(E') = RF_i(D') \text{ and } \mathcal{F}(E') = \mathcal{F}(D') \subseteq \overline{cov(a, E')}. \tag{6.10}$$

Then (a)-(c) and (e) follow immediately from the induction hypothesis. Since there is nothing to show for (d) and (f), this completes the proof.

Let $E'' = E(M, i, \lambda - 1)$ and $D'' = D(N, i, \lambda - 1)$, and suppose that the induction hypothesis is true for $E''$ and $D''$. (For $\lambda = 1$ this follows from Case 1 above.) If $E' = E''$ then $\lambda > |RF_i(E)$, so the round-$i$ Roll-Forward Phase of $E$ is finished after $E''$, i.e., $\mathcal{F}(E'') = \mathcal{I}(E'')$. From part (a) and (e) of the induction hypothesis, $\mathcal{F}(D'') = \mathcal{I}(D'')$, so the

round-$i$ Roll-Forward Phase of $D$ is also finished after $D''$, so $D' = D''$. Hence, $E' = E''$ and $D' = D''$, so (6.10) follows immediately from the induction hypothesis.

Now assume $E' = E'' \circ op$ for some operation $op$ on a register $r$ by process $p$. Since $a \notin \mathcal{F}(E'')$, $p \neq a$. By Claim 6.6 (c) no process in $cov(a, E'') - \{a\}$ takes any steps during $RF_i(E)$, so $p \notin cov(a, E'') = cov(a, E')$. From part (b) of Claim 6.6, $r \notin topw(a, E'') = topw(a, E')$, and from part (d), $owner(r) \notin cov(a, E') = cov(a, E'')$. Hence, due to part (b) of the induction hypothesis $val(r, E'') = val(r, D'')$. Since by part (e) of the induction hypothesis $\mathcal{F}(E'') = \mathcal{F}(D'') \subseteq \overline{cov(a, E'')}$ and since by part (a) every process in $\mathcal{F}(E'')$ is in exactly the same state at the end of $E''$ as at the end of $D''$, $p$ is the next process in $\mathcal{F}(D'')$ to take a step in $D'$, and its operation invocation will be exactly the same as in $E'$. Since $val(r, E'') = val(r, D'')$ so the operation response will also be the same in both executions. Hence, $D' = D'' \circ op$ and thus $D' = E'$.

It remains to show $\mathcal{F}(E') = \mathcal{F}(D') \subseteq \overline{cov(a, E')}$. During the Roll-Forward Phase, only rules (F1) and (F4) can cause new processes to be added to the roll-forward set. Suppose (F1) occurs when process $p \in \mathcal{F}(E'')$ executes $op$, i.e., it finds a process $q \neq p$, $q \notin \mathcal{F}(E'') = \mathcal{F}(D'')$, on some register $r$. If $r$ is in $q$'s local memory segment, then $p$ finds $q$ also when it executes $op$ in $D'$. Otherwise, $q$ is visible on $r$ at the end of $E''$. Then $p \neq owner(r) \notin cov(a, E'')$ by Claim 6.6 (d). But then, since $r \notin topw(a, E')$, from part (b) of the induction hypothesis the same process $q$ is visible on $r$ at the end of $D''$. Hence, the same process $q$ gets rolled forward due to operation $op$.

Now suppose that (F4) occurs because $p$ gets rolled forward and $q$ covered $p$. Then with exactly the same arguments as those made for the RMR Phase, either $p$ and $q$ get rolled forward in both, $E'$ and $D'$, or in none of them. This shows that $\mathcal{F}(E') = \mathcal{F}(D')$. Finally, $\mathcal{F}(E') \subseteq \overline{cov(a, E')}$ follows immediately from Claim 6.6 (a).

**Case 3: $\boldsymbol{\lambda = \infty}$.** In this case, $E'$ and $D'$ end after the round-$i$ Local Step Phases. Let $E''$ and $D''$ be the prefixes of $E'$ and $D'$, respectively, that end after the round-$i$ Roll-Forward Phases. Then $E' = E'' \circ LS_i(E)$ and $D' = D'' \circ LS_i(D)$.

First note that the Local Step Phase does not affect the validity of cached copies a processes has: A process $p$ has to incur an RMR on register $r$ to get a new valid cached copy of that register. Moreover, only a write-operation that incurs an RMR by some process $q$ can invalidate $p$'s cached copy of $r$. Hence, throughout the Local Step Phase, the valid cached copies of processes don't change, so (c) follows immediately from the induction hypothesis.

Now assume that (a)-(f) are true for $E''$ and $D''$. Let $E_p$ and $D_p$ be the sequence of operations executed by process $p$ during $LS_i(E)$ and during $LS_i(D)$, respectively. We first show that properties (a) and (b) are preserved throughout executions $E_p$ and $D_p$.

First consider the case $p \notin cov(a, E'') = cov(a, E')$. Recall that when the Local Step Phase starts, all processes that were previously rolled-forward are inactive. Hence, we have $p \notin \mathcal{F}(E'') = \mathcal{F}(D'')$. Note that in this case $p$ has no valid cache-copies of a register in $topw(a, E')$ at the end of $E''$ or the end of $D''$: By Claim 6.6 (b) process $p$ has not accessed such a register in $E''$ after process $a$ top-wrote to that register, and thus from induction hypothesis (a) the same is true for $D''$. Moreover, whenever $p$ accesses a register $r$ then from Claim 6.6 (d) and the assumption that $p \notin cov(a, E')$ we have $owner(r) \notin cov(a, E'')$. Thus, due to part (b) of the induction hypothesis, process $p$ finds exactly the same information on registers it accesses during $LS_i(E)$ as during $LS_i(D)$. By part (c) of the induction hypothesis, $p$ has exactly the same cached copies at the end of $E''$ as at the end of $D''$. Moreover, during $LS_i(E)$ none of the registers in $p$'s local memory segment or in $p$'s cache can be changed by a process $p' \neq p$, as this would require $p'$ to incur an RMR. Hence $p$ performs exactly the same sequence of operations in $LS_i(E)$ as in $LS_i(E)$, i.e., $E_p = D_p$.

Now consider the case $p \in cov(a, E'')$. Then (a) is trivially true for $p$. Moreover, during $E_p$ or $D_p$, process $p$ can only write to a register in its local memory segment, as any other write would incur an RMR. Hence, (b) follows.

It follows that executions $E_p$ and $D_p$ preserve properties (a) and (b). We have already argued that the sets of valid cached copies cannot not change during the Local Step Phases, so (c) is preserved. Clearly, no process top-writes, so there is nothing to show for (d). Any of the roll-forward events (F1)-(F3) require that a process incurs an RMR, so they and thus also (F4) don't happen during the Local Step Phase. Thus, (e) is maintained.

It remains to show (f): Recall that $E'$ and $D'$ finish at the end of the round-$i$ Local Step Phase of $E$ and $D$, respectively. Let $E''$ and $D''$ be the prefixes of $E'$ and $D'$, respectively, that end at the beginning of the round-$i$ RMR Phase. We use the same notation as in Case 1 ($\lambda = 1$) of the proof. Recall that $G_i(E)$ and $G_i(D)$ are the sets that were popped from the stack at the beginning of round $i$ of executions $E$ and $D$, respectively. Nothing changes for the sets that were below $G_i(E)$ and $G_i(D)$, on the stack, so we only have to show that the sets that are pushed on the stack satisfy (f). Let $M_i(E')$ and $M_i(D')$ be the sets of processes in $G_i(E)$ and $G_i(D)$, respectively, that are still active at the end of $E'$ resp. $D'$, when the Local Step Phase has ended. From (6.1) and since $\mathcal{F}(E') = \mathcal{F}(D')$, we have either

$$M(E') = M(D') \text{ or } M(E') = M(D') \cup \{a\}. \tag{6.11}$$

According to the rules of the Local Step Phase, $M(E')$ and $M(E')$ are partitioned into three subsets, $G_{i,s}(E)$ resp. $G_{i,s}(D)$ for $s \in \{1, 2, 3\}$, and these three sets are pushed on the stack.

Sets $G_{i,1}(E)$ and $G_{i,1}(D)$ contain all processes in $M(E')$ and $M(D')$ that top-wrote in $RMR_i(E)$ and in $RMR_i(D)$, respectively. Since $a$ does not top-write in $RMR_i(D)$ (but perhaps in $RMR_i(E)$), we immediately get from (6.7) that

$$G_{i,1}(E) - \{a\} = G_{i,1}(D). \tag{6.12}$$

Since $G_{i,1}(E)$ respectively $G_{i,1}(D)$ land on top of the stack, the top element of the stack satisfies property (f). Moreover, if $a \in G_{i,1}(E)$, then $j^* = 0$ and we are done.

Thus, suppose $a \notin G_{i,1}(E)$. Then $a$ does not top-write in round $i$ of $E'$, so from $i \geq t_1$ we conclude $i > t_1$. Hence, from (6.1) $a \notin G_i(D)$ and thus $a \notin M(D')$. Therefore (6.11) simplifies to

$$M(E') = M(D') \cup \{a\}. \tag{6.13}$$

We show that in this case $G_{i,2}(E) - \{a\} = G_{i,2}(D)$. Since the three sets that are pushed on the stack in $D'$ and $E'$ partition $M(E')$ and $M(D')$, resp., we then get from (6.13) that $G_{i,3}(E) - \{a\} = G_{i,3}(D)$. Since first $G_{i,3}(E)$ resp.

$G_{i,3}(D)$ and then $G_{i,2}(E)$ resp. $G_{i,2}(D)$ are pushed on the stack, the claim (f) follows.

By the assumption $a \notin G_{i,1}(E)$, $a$ is not a top-writer in $RMR_i(E)$. Then $cov(a, RMR_i(E)) = \emptyset$, and so by Claim 6.6 (c) no process in $cov(a, E') - \{a\}$ participates in the round-$i$ RMR Phase of $E'$. Thus, from (6.5) we have

$$partcpt_i(E) - \{a\} = partcpt_i(D). \tag{6.14}$$

Now note that by definition of the Local Step Phase, $G_{i,2}(E)$ respectively $G_{i,2}(D)$ contain the set of processes that are covered in round $i$, i.e., those processes that participate but don't top-write. Formally, $partcpt_i(E) = G_{i,1}(E) \cup G_{i,2}(E)$, and $partcpt_i(D) = G_{i,1}(D) \cup G_{i,2}(D)$. Now, the claim, $G_{i,2}(E) - \{a\} = G_{i,2}(D)$, follows immediately from (6.12) and (6.14). ∎

## 6.4 Proof of Lemma 6.1

Fix $b$, $j$, and $\zeta$ and for every colour schedule $M$ define $\mathcal{E}(M) = \mathcal{E}_{b,j,\zeta}(M)$ and $\mathcal{E}_{finds}(M) = \mathcal{E}^{find}_{b,j,\zeta}(M)$. Choose a colour schedule $M$ at random and let $E = E(M)$ be the execution defined by $M$. Suppose event $\mathcal{E}(M)$ occurs. Then choose $\lambda = 0$ if $\zeta = 0$, and otherwise choose $\lambda$ such that $E' = E(M, j, \lambda)$ is the prefix of $E$ that ends just before $b$'s $\zeta$-th RMR in the round-$j$ Roll-Forward Phase of $E$.

Now suppose that $\mathcal{E}_{finds}(M)$ occurs. Then $b$ finds a process $a'$ on a register $r$ when it executes in $E$ the operation that follows $E'$. Hence, $a'$ is visible on $r$ at the end of $E'$. Moreover, $a' \notin \mathcal{F}(E')$, so $a$ can be visible on $r$ at the end of $E$ only if it previously top-wrote to $r$. Now let $i$ be the earliest round whose RMR Phase completed during $E'$ such that some process $a \notin \mathcal{F}(E') \cup \{b\}$ top-wrote to $r$ in round $i$. (It is possible that $a \neq a'$.)

Now let $N = f_{j,\lambda,r,b}(M, a, i)$. From the discussion above it follows that properties (A1)-(A3) are satisfied. Moreover, (A4) is true: Suppose $b$ is covered by $a$ in the RMR Phase of round $j$ of $E' = E(M, j, \lambda)$. Then $E'$ must end during the Roll-Forward Phase of round $j$, so $b$ is rolled-forward before it finds $a'$. But then $b \in \mathcal{F}(E')$ and so $a \in \mathcal{F}(E')$ due to (F4)—a contradiction.

Since (A1)-(A4) are true, we have $N = erase(M, a, i)$. Let $D = D(N)$ and $D' = E_{b,j,\lambda}(N)$.

In $D'$ no process top-writes to register $r$ in round $i$ or later: Suppose process $p$ does in some round $j'$, $i \leq j' \leq j$, and $p$ is the first process to do so. Then by construction of $erase(M, a, i)$, $p \neq a$. But then by Lemma 6.5 (d), process $p$ also top-writes to register $r$ in round $j'$ of $E$. But when this happens, $a$ is visible, so $p$ finds $a$, contradicting $a \notin \mathcal{F}(E')$. Now recall that by our choice of $i$, no process in $\overline{\mathcal{F}(E')}$ top-writes register $r$ in $E'$ before round $i$. From the definition of the mapping *erase*, the first $i-1$ rounds of $D$ and $E$ are identical, and from Lemma 6.5 (d), $\mathcal{F}(D') = \mathcal{F}(E')$. Hence, in $D$ no process in $\overline{\mathcal{F}(E')}$ top-writes register $r$ before round $i$, either. It follows that at the end of $D'$ either no process is visible on $r$, or if some process $q$ is visible on $r$, then $q \in \mathcal{F}(D')$. I.e.,

$$\text{At the end of } D' \text{ no process in } \overline{\mathcal{F}(E')} \text{ is visible on } r. \tag{6.15}$$

Now note that during $E'$ process $b$ never accesses a register $r'$ that $a$ top-writes during $E'$. If $b$ accessed $r'$ in a round other than the one when $a$ top-writes $r'$ for the first time, then either $a$ would find $b$ or $b$ would find $a$, and so $a \in \mathcal{F}(E')$. IF $b$ accessed $r'$ in the round in which $a$ top-writes $r'$ for the first time, then $a$ would cover $b$ which cannot happen according to Claim 6.6 (c).

Hence, $b \notin cov(a, E')$, so we can apply Lemma 6.5 (a), and we obtain

$$state(b, E') = state(b, D'). \tag{6.16}$$

Now Lemma 6.5 (e) implies that if $\zeta > 0$, then the step that follows $D'$ in the execution $E(N)$ will be $b$'s $\zeta$-th RMR of the round-$j$ Roll-Forward Phase. But from (6.16) we see that in this step $b$ does not find any other process. If $\zeta = 0$, then we get from Lemma 6.5 (f) that $b$ will be popped from the stack at the beginning of round $i$ of $D$. Since $N_{i,r} = 1$ ($a$ does not top-write $r$ in the round-$j$ RMR Phase of $E$) $b$ will access $r$. Again from (6.16) we conclude that $b$ does not find any other process during the round-$i$ RMR Phase of $D$.

Therefore, we have established the following:

$$\begin{aligned}&\text{If } \mathcal{E}(M) \wedge \mathcal{E}^{find}(M), \text{ then}\\ &\quad \text{there exist a colour schedule } N,\ a \in \mathcal{P}, \text{ and } i \leq j\text{:}\\ &\qquad N = f_{j,\lambda,r,b}(M, a, i) \text{ and } \mathcal{E}(N) \wedge \overline{\mathcal{E}^{find}(N)}.\end{aligned} \tag{6.17}$$

Let $G_j(D)$ and $G_j(E)$ be the sets of processes popped from the stack in the round-$j$ RMR Phases of $D$ and $E$, respectively. There is at least one process $c \in G_j(E) - \{a\}$: If $\lambda = 0$, then $c = b$, and otherwise $c$ is the first process that triggers a process to be rolled forward during the round-$j$ RMR Phase of $E$. From 6.5 (f) it follows that $G_j(D) = G_j(E) - \{a\}$, so $c \in G_j(D)$. Let $T_j(N)$ and $T_j(M)$ be the set of indices $\ell$ such that $c$ participates during round $\ell < j$ of $D$ and $E$, respectively. Let $D''$ and $E''$ be the prefixes of $E$ and $D$, respectively, that end immediately after round $j-1$. From Claim 6.6 (a) applied to $E''$ we have $c \notin cov(a, E'')$. Thus, from Lemma 6.5 (a) we get $state(a, E'') = state(a, D'')$. Hence, $T_j(N) = T_j(M)$. Finally, we argue that $i \in T_j(N) \cup \{j\}$: If not then $i \notin T_j(M)$ and $i < j$, so process $a$ top-writes in round $i$ while $c$ does not participate in round $i$. But then at the end of round $i$ process $c$ is in a set on the stack below $a$'s set, so $c$ will not get popped from the stack again until $a$ has been rolled forward (contradicting $c \in G_j(E)$ and $a \notin \mathcal{F}(E')$). Hence, (6.17) is equivalent to:

$$\begin{aligned}&\text{If } \mathcal{E}(M) \wedge \mathcal{E}^{find}(M), \text{ then}\\ &\text{there exist a colour schedule } N,\ a \in \mathcal{P}, \text{ and } i \in T_j(N) \cup \{j\}\text{:}\\ &\qquad N = f_{j,\lambda,r,b}(M, a, i) \text{ and } \mathcal{E}(N) \wedge \overline{\mathcal{E}^{find}(N)}.\end{aligned} \tag{6.18}$$

Now fix some colour schedule $N$. Let $\mathcal{M}(N)$ be the set of triples $(M, a, i)$ such that $M \neq N$ is a colour schedule, $a \in \mathcal{P}$, $i \in T_j(N) \cup \{j\}$ and $f_{j,\lambda,r,b}(M, a, i) = N$. Note that $|T_j(N) \cup \{j\}| \leq k$, because by roll-forward rule (F3) no process can participate in more than $k$ rounds. Hence, from Lemma 6.4 we get

$$|\mathcal{M}(N)| \leq k \cdot (k+2). \tag{6.19}$$

Moreover, for any $(M, a, i) \in \mathcal{M}(N) - \{N\}$, the probability that $N$ is chosen by the adversary is at most $\varepsilon/(1-\varepsilon)$ times the probability that $M$ is chosen by the adversary, because $M$ dominates $N = erase(M, a, i)$, and there is at least one array entry that is red in $N$ but green in $M$. I.e.,

$$\forall M \in \mathcal{M}(N):\ \text{Prob}(N) \leq \text{Prob}(M) \cdot \frac{\varepsilon}{1-\varepsilon}. \tag{6.20}$$

Now let $\mathcal{N}$ be the set of colour schedules such that $\mathcal{E}(N) \wedge \overline{\mathcal{E}^{find}}$. Then from (6.18) we conclude for random colour

schedules $M'$ and $N'$:

$$\begin{aligned}
&\mathrm{Prob}\Big(\mathcal{E}(M') \wedge \mathcal{E}^{find}(M')\Big) \\
&\quad \le \sum_{N\in\mathcal{N}} \sum_{(M,a,i)\in\mathcal{M}(N)} \mathrm{Prob}(M) \\
&\quad \overset{(6.20)}{\le} \sum_{N\in\mathcal{N}} \sum_{(M,a,i)\in\mathcal{M}(N)} \mathrm{Prob}(N)\cdot\frac{\varepsilon}{1-\varepsilon} \\
&\quad \overset{(6.19)}{\le} \sum_{N\in\mathcal{N}} k(k+2)\cdot\mathrm{Prob}(N)\cdot\frac{\varepsilon}{1-\varepsilon} \\
&\quad = k(k+2)\frac{\varepsilon}{1-\varepsilon}\cdot\sum_{N\in\mathcal{N}}\mathrm{Prob}(N) \\
&\quad = k(k+2)\frac{\varepsilon}{1-\varepsilon}\cdot\mathrm{Prob}\big(N'\in\mathcal{N}\big).
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathrm{Prob}\Big(\mathcal{E}^{find}(M') \;\Big|\; \mathcal{E}(M')\Big) \\
&\qquad = \frac{\mathrm{Prob}\big(\mathcal{E}(M')\wedge\mathcal{E}^{find}(M')\big)}{\mathrm{Prob}(\mathcal{E}(M'))} \\
&\qquad \le \frac{k(k+2)\frac{\varepsilon}{1-\varepsilon}\cdot\mathrm{Prob}\big(\mathcal{E}(N')\wedge\mathcal{E}^{find}(N')\big)}{\mathrm{Prob}(\mathcal{E}(M'))} \\
&\qquad = k(k+2)\frac{\varepsilon}{1-\varepsilon}\cdot\mathrm{Prob}\Big(\mathcal{E}^{find}(N') \;\Big|\; \mathrm{Prob}\big(\mathcal{E}(N')\big)\Big).
\end{aligned}$$

# 7. PROOF OF LEMMA 4.1

The proof is based on a potential function analysis. With every process $p \in \mathcal{P}''$ that is *active* after round $i \in \{0,\ldots,n^2\}$,[4] we associate a potential

$$\Phi_{p,i} = \log_k N_{p,i} - X_{p,i} - Y_{p,i},$$

where $N_{p,i}$ is the number of *active* processes that are in the same set as $p$ on the stack after round $i$; $X_{p,i}$ is the total number of RMRs incurred by $p$ until the end of round $i$; and $Y_{p,i}$ is the number of valid cached copies of registers that $p$ has after round $i$. If process $p \in \mathcal{P}''$ is not active after round $i$, then its potential is $\Phi_{p,i} = 0$. We define the (total) potential after round $i$ as

$$\Phi_i = \sum_{p\in\mathcal{P}''} \Phi_{p,i}.$$

Note that the potential initially (after round 0) is $\Phi_0 = N\log_k N$, where $N := |\mathcal{P}''|$. Since $\mathrm{Exp}[N] = \varepsilon|\mathcal{P}'|$, it follows from Jensen's inequality and the convexity of function $x\log_k x$, that

$$\mathrm{Exp}[\Phi_0] \ge \varepsilon|\mathcal{P}'|\cdot\log_k(\varepsilon|\mathcal{P}'|). \tag{7.1}$$

In Section 7.1, we prove that the expected potential decrease in a round is bounded by the expected number of RMRs incurred in that round scaled by some constant factor. Then, in Section 7.2, we use this result to derive Lemma 4.1.

## 7.1 Analysis of a Single Round

Let $X_i$, $0 \le i \le r$, denote the total number of RMRs incurred until the end of round $i$. The following lemma is the main result of this section.

[4]By "round 0" we mean the period before the first round starts.

LEMMA 7.1. *There is a constant $c > 0$ such that for all rounds $1 \le i \le r$,*

$$\mathrm{Exp}[\Phi_{i-1} - \Phi_i] \le c\cdot\mathrm{Exp}[X_i - X_{i-1}].$$

Below we introduce some notation and give an outline of the steps of the proof. In the proof we only look at a single round, round $i$, and all the notation we describe is with respect to that round. Thus in the notation we do not make explicit the dependence on $i$.

We denote by $G_{part}$ the set of processes that participate in the RMR Phase of round $i$, and by $G_{halt} = G_i - G_{part}$ the set of halted processes. We let $X_{rmr} = |G_{part}|$ be the number of RMRs induced in the RMR Phase, and we let $X_{rf}$ be the number of RMRs induced in the Roll-Forward Phase. Also, we denote by $Y_{inv}$ the number of cached copies that were valid at the beginning of round $i$ and got invalidated in the RMR Phase.

The proof first bounds the decrease in the number of active process during round $i$, in terms of $X_{rmr}$, $X_{rf}$, and $Y_{inv}$. Then it bounds the potential difference induced by this decrease, and also by the further partitioning of $G_i$ into smaller sets.

Recall that in the RMR phase, each process $p \in G_{part}$ executes a single step; in this step, $p$ may *find* some other process, or it may *spoil* some process. Let $F_{rmr\text{-}find}$ be the number of processes that find other processes in the RMR Phase, and let $F_{spoil}$ be the number of processes that spoil other processes. We show that $\mathrm{Exp}[F_{rmr\text{-}find}]$ in bounded, roughly, by $\varepsilon k^2\cdot\mathrm{Exp}[X_{rmr}]$ (Claim 7.2), and that $\mathrm{Exp}[F_{spoil}]$ is bounded by $\varepsilon\cdot\mathrm{Exp}[Y_{inv}]$ (Claim 7.3).

In the Roll-Forward Phase following the RMR Phase, any processes that found or spoilt some processes in the RMR Phase get rolled forward, together with processes that were found or spoilt. As a result, more processes may be found and thus get rolled forward. We denote by $F_{rf\text{-}found}$ the number of these new processes added to the roll-forward set because they are found during the Roll-Forward Phase. We show that $\mathrm{Exp}[F_{rf\text{-}found}]$ is bounded by, roughly, $\varepsilon k^2\cdot\mathrm{Exp}[X_{rf}]$ (Claim 7.4). Processes can also get rolled forward because they have incurred their $k$-th RMR, or because of covering rule (F4). Let $F$ be the number of all the processes that get rolled-forward during the round. We show that $F$ is bounded by $2k\cdot(F_{spoil}+F_{rmr\text{-}find}+F_{rf\text{-}found})+I\cdot X_{rmr}$, where $I = 1$ if processes in $G_{part}$ incur their $k$-th RMR in the RMR phase, and $I = 0$ otherwise (Claim 7.5).

Recall that the subset of processes in $G_i$ that are still active after the end of the round is partitioned into three sets before they are pushed back onto the stack: top-writers, participating processes that did not top-write, and halted processes We show that the expected potential difference induced by partitioning $G_i$ into sets $G_1^*$, $G_2^*$, and $G_{halt}$, where $\{G_1^*, G_2^*\}$ is an arbitrary partition of $G_{part}$, is bound by $O(1)\cdot\mathrm{Exp}[X_{rmr}]$ (Claim 7.6)—this does not take into account that some processes become inactive during the round. Then, we show that the expected additional decrease in the potential that results from the removal of $F+1$ processes in total from these sets is bounded, roughly, by $k\cdot\mathrm{Exp}[F] + O(1)\cdot\mathrm{Exp}[X_{rmr}]$ (Claim 7.7).

In the final part of the proof, we combine the above results to derive Lemma 7.1.

### *Bounding the Processes Rolled Forward*

The first result is an upper bound on the expected number of processes that find a process in the RMR Phase.

CLAIM 7.2. $\text{Exp}[F_{rmr\text{-}find}] \leq \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2) \cdot \text{Exp}[X_{rmr}]$.

PROOF. For each process $p$, let $\mathcal{E}_p$ be the event that $p$ participates in the RMR Phase, and $\mathcal{E}'_p$ the event that it finds a process in that phase. Then, by Lemma 6.1, $\text{Prob}(\mathcal{E}'_p \mid \mathcal{E}_p) \leq \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2)$, and thus,

$$\begin{aligned}
\text{Exp}[F_{rmr\text{-}find}] &= \sum_p \text{Prob}(\mathcal{E}'_p) \\
&= \sum_p \text{Prob}(\mathcal{E}'_p \mid \mathcal{E}_p) \cdot \text{Prob}(\mathcal{E}_p) \\
&\leq \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2) \cdot \sum_p \text{Prob}(\mathcal{E}_p) \\
&= \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2) \cdot \text{Exp}[|G_{part}|].
\end{aligned}$$

■

Next we bound the expected number of processes that spoil a process. Recall that $Y_{inv}$ is the number of cached copies that are valid at the beginning of the round but are invalidated during the RMR Phase.

CLAIM 7.3. $\text{Exp}[F_{spoil}] \leq \varepsilon \cdot \text{Exp}[Y_{inv}]$.

PROOF. Fix the configuration $C$ at the beginning of the round, before the adversary decides the colours of the registers. Let $G_{potw}$ denote the set of "potential" top-writers for the round, i.e., the subset of processes $p \in G_i$ that will be top-writers if they participate in the round. We denote by $r_p$ the register that $p \in G_{potw}$ is poised to write. and by $V_p$ the set of processes $q \in G_{potw} - \{p\}$ that have a valid cached copy of $r_p$ in $C$. Since we have fixed the configuration $C$ at the beginning of the round, the set $G_{potw}$ is also fixed, and so are the registers $r_p$ and the sets $V_p$, for all $p \in G_{potw}$.

For each $p \in G_{potw}$, let $\mathcal{E}_p$ be the event that $p$ is a top-writer in this round, which is the same as the event that $r_p$ is green in this round. Thus, the events $\mathcal{E}_p$, for $p \in G_{potw}$, are mutually independent (since $r_p \neq r_q$ for distinct $p, q \in G_{potw}$), and $\text{Prob}(\mathcal{E}_p) = \varepsilon$. Let $\mathcal{E}'_p$ be the event that $p$ spoils some process in this round. Clearly, $\mathcal{E}'_p = \mathcal{E}_p \wedge \bigvee_{q \in V_p} \mathcal{E}_q$, and thus

$$\begin{aligned}
\text{Prob}(\mathcal{E}'_p) &= \text{Prob}\left(\mathcal{E}_p \wedge \bigvee_{q \in V_p} \mathcal{E}_q\right) \\
&\leq \text{Prob}(\mathcal{E}_p) \cdot \sum_{q \in V_p} \text{Prob}(\mathcal{E}_q) \\
&= \text{Prob}(\mathcal{E}_p) \cdot |V_p| \cdot \varepsilon.
\end{aligned}$$

Now,

$$\begin{aligned}
\text{Exp}[F_{spoil}] &= \sum_{p \in G_{potw}} \text{Prob}(\mathcal{E}'_p) \\
&\leq \sum_{p \in G_{potw}} \text{Prob}(\mathcal{E}_p) \cdot |V_p| \cdot \varepsilon \\
&= \text{Exp}[Y_{inv}] \cdot \varepsilon.
\end{aligned}$$

■

We now bound the expected number of processes that are found by other processes in the Roll-Forward Phase.

CLAIM 7.4. $\text{Exp}[F_{rf\text{-}found}] \leq \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2) \cdot \text{Exp}[X_{rf}]$.

PROOF PROOF OF CLAIM 7.4. For $1 \leq \lambda \leq X_{rf}$, let $\mathcal{E}_\lambda$ be the event that some process is found during the step in which the $\lambda$-th RMR of the Roll-Forward Phase is incurred. From Lemma 6.1 it follows that

$$\text{Prob}(\mathcal{E}_\lambda \mid \lambda \leq X_{rf}) \leq \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2).$$

Then, for $Z_\lambda = \mathbf{1}_{\mathcal{E}_\lambda}$ the indicator random variable of event $\mathcal{E}_\lambda$, we have $\text{Exp}[Z_\lambda \mid \lambda \leq X_{rf}] \leq \frac{\varepsilon}{1-\varepsilon} \cdot k(k+2)$. And since $F_{rf\text{-}found} = \sum_{\lambda=1}^{X_{rmr}} Z_\lambda$, the claim follows from Wald's theorem. ■

Recall that $F$ is the total number of processes that get rolled forward in round $i$. In the next claim we bound $F$ in terms of the quantities we bounded in the previous claims. By $\ell$ we denote the number of rounds in which the processes in set $G_{part}$ have participated, including round $i$ (by Claim 6.2(c), all these processes have participated in exactly the same sequence of rounds). Note that $\ell$ is also equal to the number of RMRs that each process in $G_{part}$ has incurred until the end of the RMR Phase in round $i$.

CLAIM 7.5.

$$F \leq 2k \cdot (F_{rmr\text{-}find} + F_{spoil} + F_{rf\text{-}found}) + \mathbf{1}_{\{\ell=k\}} \cdot X_{rmr}.$$

PROOF. By Rule (F1), at most $2F_{rmr\text{-}find}$ processes get rolled forward because during the RMR Phase they either find a process or are found by a process; and $F_{rf\text{-}found}$ processes get rolled forward because they are found by a process during the Roll-Forward Phase. By (F2), at most $2F_{rmr\text{-}find}$ processes get rolled forward because either they spoil some process or they are spoilt by a process (not all processes that are spoilt get rolled forward). By (F3), if $\ell = k$ then a number of $|G_{part}| = X_{rmr}$ processes get rolled forward because they incur their $k$-the RMR in the RMR Phase. It remains to account for the processes that get rolled forward due to (F4). From Claim 6.2(a) it follows that for every process that gets rolled forward because of (F1) or (F2), at most $(k-1)$ other processes get rolled forward because of (F4). And from Claim 6.2(b) it follows that if a process gets rolled-forward because of (F3), then this does not cause any additional processes to get rolled forward because of (F4). Combining the above yields the claim. ■

From the above result and Claim 6.2(d), it follows that at most $F + 1$ processes become inactive in round $i$.

### *Bounding the Potential Difference*

First we bound the decrease $\Delta\Phi_{split}$ in the potential that results if we partition set $G_i$ into sets $G_1^*$, $G_2^*$ and $G_3^*$, where $\{G_1^*, G_2^*\}$ is an arbitrary partition of $G_{part}$ and $G_3^* = G_{halt}$. In the description of the adversary, the partitioning takes place at the end of the round. However, it is equivalent if at the beginning of the round (after the adversary has decided the colours of the register) we partition the processes in $G_i$ into top-writers, participating processes that do not top-write, and halted processes, and then at the end of the round we remove from these sets any inactive processes.

From the definition of potential,

$$\Delta\Phi_{split} = f(|G_i|) - \sum_{j=1}^{3} f(|G_j^*|),$$

where $f$ is the function

$$f(x) = \begin{cases} x \log_k x & \text{if } x \geq 1; \\ 0 & \text{if } x \leq 1. \end{cases}$$

CLAIM 7.6. $\text{Exp}[\Delta\Phi_{split}] \leq c' \cdot \text{Exp}[X_{rmr}]$, *for some constant* $c'$.

PROOF. Fix $G_i$. Then,

$$\text{Exp}[\Delta\Phi_{split}] = f(|G_i|) - \text{Exp}\left[\sum_{j=1}^{3} f(|G_j^*|)\right].$$

We have

$$\begin{aligned} &\text{Exp}\left[\sum_{j=1}^{3} f(|G_j^*|)\right] \\ &\quad = \sum_{j=1}^{3} \text{Exp}[f(|G_j^*|)] \geq \sum_{j=1}^{3} f(\text{Exp}[|G_j^*|]), \\ &\quad \geq 2f\left(\frac{\text{Exp}[|G_1^*|] + \text{Exp}[|G_2^*|]}{2}\right) + f(\text{Exp}[|G_3^*|]), \end{aligned}$$

where the second relation follows from Jensen's inequality since $f$ is a convex function, and the last relation follows again from the convexity of $f$.

Let $m = |G_i|$. Then, $\text{Exp}[|G_1^*|] + \text{Exp}[|G_2^*|] = \text{Exp}[|G_{part}|] = \varepsilon m$ and $\text{Exp}[|G_3^*|] = m - \varepsilon m$. If we substitute these values above, and we assume that $\varepsilon m/2 \geq 1$, we get

$$\begin{aligned} &\text{Exp}\left[\sum_{j=1}^{3} f(|G_j^*|)\right] \\ &\quad \geq 2f\left(\frac{\varepsilon m}{2}\right) + f(m - \varepsilon m) \\ &\quad = 2\frac{\varepsilon m}{2} \log_k \left(\frac{\varepsilon m}{2}\right) + (m - \varepsilon m) \log_k (m - \varepsilon m) \\ &\quad = \varepsilon m \left(\log_k m + \log_k(\varepsilon/2)\right) \\ &\qquad\qquad + (m - \varepsilon m)\left(\log_k m + \log_k(1 - \varepsilon)\right) \\ &\quad = \varepsilon m(\log_k m - O(1)) \\ &\qquad\qquad + (m - \varepsilon m)\left(\log_k m - O(\varepsilon/\log k)\right) \\ &\quad = m \log_k m - O(1) \cdot \varepsilon m \\ &\quad = f(|G_i|) - O(1) \cdot \text{Exp}[|G_{part}|]. \end{aligned}$$

Rearranging, and observing that $|G_{part}| = X_{rmr}$ yields

$$\text{Exp}[\Delta\Phi_{split}] = O(1) \cdot \text{Exp}[X_{rmr}].$$

Now if $\varepsilon m/2 < 1$, we have $f(\varepsilon m/2) = 0$, and similarly as before,

$$\begin{aligned} \text{Exp}\left[\sum_{j=1}^{3} f(|G_j^*|)\right] &\geq f(m - \varepsilon m) \\ &= (m - \varepsilon m)\left(\log_k m - O(\varepsilon/\log k)\right) \\ &= m \log_k m - O(1) \cdot \varepsilon m \\ &= f(|G_i|) - O(1) \cdot \text{Exp}[X_{rmr}]. \end{aligned}$$

■

Suppose that at the beginning of round $i$ there are $L - 2$ sets of processes on the stack, thus after set $G_i$ is popped and split into sets $G_1^*, G_2^*, G_3^*$, we have $L$ sets. Let $N_1, \ldots, N_L$ be the number of active processes in these sets at the beginning of the round, and let $N_1', \ldots, N_L'$ be the corresponding numbers at the end of the round. Clearly, $N_j' \leq N_j$, for $1 \leq j \leq L$, and $F \leq \sum_{j=1}^{L}(N_j - N_j') \leq F + 1$, where the upper bound follows from Claim 6.2(d). We now bound the decrease $\Delta\Phi_{shrink}$ in the potential due to the decrease in the sizes of these sets,

$$\Delta\Phi_{shrink} = \sum_{j=1}^{L} \left(f(N_j) - f(N_j')\right).$$

CLAIM 7.7. $\text{Exp}[\Delta\Phi_{shrink}] \leq (k + c') \cdot \text{Exp}[F] + c' \cdot \text{Exp}[X_{rmr}]$, *for some constant* $c'$.

PROOF. We break $\Delta\Phi_{shrink}$ into two terms, $\Delta\Phi_{shrink} = \Delta\Phi_1 + \Delta\Phi_2$, where $\Delta\Phi_1$ accounts for the removal of $F$ processes in total from arbitrary sets, and $\Delta\Phi_2$ accounts for the removal from $G_1^*$ or $G_2^*$, of the (at most) one process $p$ that becomes inactive in round $i$ without getting rolled forward. Clearly, $p$ must be a process in $G_1^* \cup G_2^* = G_{part}$, because before $p$ becomes inactive it must perform an RMR in the RMR Phase of round $i$.

First we bound $\Delta\Phi_1$. We write $N_1'', \ldots, N_L''$ to denote the set sizes after the removal of the $F$ processes. Then

$$\Delta\Phi_1 = \sum_{j=1}^{L} \left(f(N_j) - f(N_j'')\right).$$

From the convexity of $f$ and the fact that $N_j' \leq N_j$ for all $1 \leq j \leq L$, it follows by induction that

$$\sum_{j=1}^{L} \left(f(N_j) - f(N_j'')\right) \leq f\left(\sum_{j=1}^{L} N_j\right) - f\left(\sum_{j=1}^{L} N_j''\right).$$

The right-hand size is equal to

$$f\big(\sum_{j=1}^{L} N_j\big) - f\big(\sum_{j=1}^{L} N_j - F\big),$$

and by of the convexity of $f$, this is at most $f(n) - f(n-F) = (k + O(1)) \cdot F$. Thus,

$$\text{Exp}[\Delta\Phi_1] \leq (k + O(1)) \cdot \text{Exp}[F]. \tag{7.2}$$

Next we bound $\Delta\Phi_2$. Fix $G_i$ and let $m = |G_i|$. If $G_{part} = \emptyset$ then $\Delta\Phi_2 = 0$, and if $G_{part} \neq \emptyset$ then

$$\begin{aligned} \Delta\Phi_2 &\leq f(|G_{part}|) - f(|G_{part}| - 1) \leq f(m) - f(m-1) \\ &\leq 3 \log_k m. \end{aligned}$$

Also, $\text{Prob}(G_{part} \neq \emptyset) \leq \min\{1, \text{Exp}[G_{part}]\} = \min\{1, \varepsilon m\}$. Then,

$$\begin{aligned} \text{Exp}[\Delta\Phi_2] &= \text{Exp}[\Delta\Phi_2 \mid G_{part} \neq \emptyset] \cdot \text{Prob}(G_{part} \neq \emptyset) \\ &\leq (3 \log_k m) \cdot \min\{1, \varepsilon m\}. \end{aligned}$$

If $\varepsilon m \leq 1$ then $\log_k m = O(1)$, and if $\varepsilon m \geq 1$ then $\log_k m = O(\varepsilon N)$. Therefore,

$$\text{Exp}[\Delta\Phi_2] = O(1) \cdot \varepsilon m = O(1) \cdot \text{Exp}[X_{rmr}]. \tag{7.3}$$

The claim now follows from (7.2) and (7.3). ■

### *Putting the Pieces Together*

We now have all the pieces we need to derive Lemma 7.1. The potential difference $\Phi_{i-1} - \Phi_i$ is comprised of three parts: (1) The decrease $\Delta\Phi_{split} + \Delta\Phi_{shrink}$ in the sum, over all active processes $p$ at the end of the round, of $\log_k N_{p,i}$; (2) The number of RMRs incurred in round $i$, which is $X_{rmr} + X_{rf}$, minus the number of RMRs incurred in all rounds by processes that become inactive in round $i$, which is least $\mathbf{1}_{\{\ell=k\}} \cdot k \cdot |G_{part}| = \mathbf{1}_{\{\ell=k\}} \cdot kX_{rmr}$; and (3) The increase in the number of valid cached copies, which is upper-bounded by the number $X_{rmr}$ of RMR operations in the RMR phase minus the number $Y_{inv}$ of cached copies that are invalidated in the RMR phase. Thus,

$$\begin{aligned}\Phi_{i-1} - \Phi_i \leq\ & (\Delta\Phi_{split} + \Delta\Phi_{shrink}) \\ & + (X_{rmr} + X_{rf} - \mathbf{1}_{\{\ell=k\}} \cdot kX_{rmr}) \\ & + (X_{rmr} - Y_{inv}).\end{aligned}$$

Taking the expectation and applying Claims 7.6 and 7.7 yields

$$\begin{aligned}\mathrm{Exp}[\Phi_{i-1} - \Phi_i] \leq \mathrm{Exp}\big[&(\beta_1 - \mathbf{1}_{\ell=k} \cdot k) \cdot X_{rmr} + X_{rf} - Y_{inv} \\ &+ (k + \beta_2) \cdot F\big],\end{aligned}$$

for constants $\beta_1, \beta_2 > 0$. By applying Claim 7.5 to bound $F$, and then Claims 7.2, 7.3, and 7.4, we get that for some constant $\beta_3 > 0$,

$$\begin{aligned}&\mathrm{Exp}[\Phi_{i-1} - \Phi_i] \\ &\quad\leq \mathrm{Exp}\big[(\beta_1 - \mathbf{1}_{\ell=k} \cdot k) \cdot X_{rmr} + X_{rf} - Y_{inv} \\ &\qquad + 2k(k + \beta_2) \cdot (\beta_3 \varepsilon k^2 \cdot X_{rmr} + \varepsilon \cdot Y_{inv} + \beta_3 \varepsilon k^2 \cdot X_{rf}) \\ &\qquad + (k + \beta_2) \cdot \mathbf{1}_{\ell=k} \cdot X_{rmr}\big] \\ &\quad\leq c \cdot \mathrm{Exp}[X_{rmr} + X_{rf}],\end{aligned}$$

since $\varepsilon = O(1/k^4)$. This completes the proof of Lemma 7.1.

## 7.2 The Bound on the Total Number of RMRs

We now bound the expectation of the total number of RMR incurred in all rounds.

Summing over all rounds $1 \leq i \leq n^2$ in both sides of the inequality in Lemma 7.1, we obtain

$$\begin{aligned}\mathrm{Exp}[\Phi_0] - \mathrm{Exp}[\Phi_{n^2}] &\leq c \cdot (\mathrm{Exp}[X_{n^2}] - \mathrm{Exp}[X_0]) \\ &= c \cdot \mathrm{Exp}[X_{n^2}],\end{aligned}$$

since $X_0 = 0$, and thus

$$\mathrm{Exp}[X_{n^2}] \geq c^{-1}(\mathrm{Exp}[\Phi_0] - \mathrm{Exp}[\Phi_{n^2}]). \tag{7.4}$$

Next we show a (crude) upper bound on $\mathrm{Exp}[\Phi_{n^2}]$ in terms of $\mathrm{Exp}[X_{n^2}]$.

LEMMA 7.8. $\mathrm{Exp}[\Phi_{n^2}] \leq \dfrac{n \log_k n}{\varepsilon n^2} \cdot \mathrm{Exp}[X_{n^2}]$.

PROOF. Let $i^*$ be the first round after which the stack is empty, or round $n^2$ if no such round exists. For every $1 \leq i \leq i^*$, define the 0/1 random variable $Z_i$ with $Z_i = 1$, if $X_i > X_{i-1} \vee G_i = \emptyset$ (i.e., at least one RMR is incurred in round $i$ or the set of processes popped from the stack at the beginning of round $i$ contains no active processes); and $Z_i = 0$, otherwise.

We start by showing that

$$\sum_{i=1}^{i^*} Z_i \leq X_{n^2}. \tag{7.5}$$

We will argue that with each round $i$ for which $Z_i = 1$, we can associate a *distinct* RMR incurred during round $i$ or some previous round; this directly implies the above inequality. Fix some round $i$ such that $Z_i = 1$. If $X_i > X_{i-1}$ then at least one RMR in incurred in the *RMR Phase* of round $i$; we can associate any of these RMRs with round $i$. Suppose now that $X_i = X_{i-1}$. Then, since we assume that $Z_i = 1$, it holds that $G_i = \emptyset$, i.e., no active process is popped from the stack in round $i$. Since no empty sets are ever pushed onto the stack, and only processes that are active are pushed onto the stack, it follows that some inactive process $p$ is popped in round $i$, and $p$ was active when it was last pushed onto the stack, in some round $i' < i$. Thus, it must be true that $p$ got rolled forward during the Roll-Forward Phase of some round $j$, $i' < j < i$. Further, when $p$ was pushed onto the stack at the end of round $i'$, it was poised to incur an RMR, because either $p$ *participated* in the RMR Phase of round $i'$ and it was then stopped at the end of the Local Step Phase, or $p$ was *halted* in round $i'$. Thus, when $p$ got rolled forward in round $j$ it incurred that RMR. This is the RMR that we associate with round $i$, in this case.

We now observe that $\mathrm{Exp}[Z_i \mid i \leq i^*] \geq \varepsilon$: If $G_i \neq \emptyset$, then some active process $p$ is popped from the stuck at the beginning of round $i$. Process $p$ will incur an RMR in the RMR Phase of round $i$ if the register that $p$ is poised to access is green in this round, which happens with probability $\varepsilon$. In the complementary case, i.e., if $G_i \neq \emptyset$, then, by definition, $Z_i = 1 \geq \varepsilon$. So, in both cases it holds that $\mathrm{Exp}[Z_i \mid i \leq i^*] \geq \varepsilon$. Then, by Wald's Theorem,

$$\mathrm{Exp}\left[\sum_{i=1}^{i^*} Z_i\right] \geq \varepsilon \cdot \mathrm{Exp}[i^*].$$

Now,

$$\mathrm{Exp}[i^*] \geq n^2 \cdot \mathrm{Prob}\big(i^* = n^2\big) \geq n^2 \cdot \mathrm{Prob}\big(\Phi_n^2 > 0\big),$$

since $\Phi_n^2 > 0$ implies $i^* = n^2$. Also, by Markov's inequality,

$$\begin{aligned}\mathrm{Prob}(\Phi_{n^2} = 0 \mid \Phi_0) &= \mathrm{Prob}(\Phi_0 - \Phi_{n^2} = \Phi_0 \mid \Phi_0) \\ &\leq \frac{\Phi_0 - \mathrm{Exp}[\Phi_{n^2} \mid \Phi_0]}{\Phi_0} \\ &\leq 1 - \mathrm{Exp}[\Phi_{n^2} \mid \Phi_0]/(n \log_k n),\end{aligned}$$

since $\Phi_0 \leq n \log_k n$. Taking the expectation yields

$$\mathrm{Prob}(\Phi_{n^2} = 0) \leq 1 - \mathrm{Exp}[\Phi_{n^2}]/(n \log_k n),$$

and thus, $\mathrm{Prob}(\Phi_{n^2} > 0) \geq \mathrm{Exp}[\Phi_{n^2}]/(n \log_k n)$. Therefore,

$$\mathrm{Exp}\left[\sum_{i=1}^{i^*} Z_i\right] \geq \varepsilon n^2 \cdot \mathrm{Exp}[\Phi_{n^2}]/(n \log_k n).$$

The claim now follows from the above inequality and (7.5). ∎

Combining (7.4) with Lemma 7.8, we obtain

$$\mathrm{Exp}[X_{n^2}] \geq \frac{\mathrm{Exp}[\Phi_0]}{c + (n \log_k n)/(\varepsilon n^2)} \geq \frac{\mathrm{Exp}[\Phi_0]}{2c}.$$

Finally, by applying the lower bound for $\mathrm{Exp}[\Phi_0]$ from (7.1), and using the assumption that $|\mathcal{P}'| \geq n/2$, we obtain that $\mathrm{Exp}[X_{n^2}] = \Omega(\varepsilon n \log_k n)$.

This completes the proof of Lemma 4.1.

## 8. REFERENCES


[1] James H. Anderson and Yong-Jik Kim. Fast and scalable mutual exclusion. In *Proceedings of the 13th International Symposium on Distributed Computing (DISC)*, pages 180–194, 1999.

[2] James H. Anderson and Yong-Jik Kim. Adaptive mutual exclusion with local spinning. In *Proceedings of the 14th International Symposium on Distributed Computing (DISC)*, pages 29–43, 2000.

[3] James H. Anderson and Yong-Jik Kim. An improved lower bound for the time complexity of mutual exclusion. *Distributed Computing*, 15:221–253, 2002.

[4] Thomas E. Anderson. The performance of spin lock alternatives for shared-memory multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):6–16, 1990.

[5] James Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16:165–175, 2003.

[6] Hagit Attiya, Danny Hendler, and Philipp Woelfel. Tight RMR lower bounds for mutual exclusion and other problems. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 217–226, 2008.

[7] Michael Bender and Seth Gilbert. Mutual exclusion with $O(\log^2 \log n)$ amortized work. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 728–737, 2011.

[8] Robert Cypher. The communication requirements of mutual exclusion. In *7th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 147–156, 1995.

[9] Robert Danek and Wojciech M. Golab. Closing the complexity gap between FCFS mutual exclusion and mutual exclusion. *Distributed Computing*, 23(2):87–111, 2010.

[10] Edsger W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 8:569, 1965.

[11] Rui Fan and Nancy A. Lynch. An $\Omega(\log n)$ lower bound on the cost of mutual exclusion. In *Proceedings of the 25th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 275–284, 2006.

[12] Faith Ellen Fich, Danny Hendler, and Nir Shavit. Linear lower bounds on real-world implementations of concurrent objects. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 165–173, 2005.

[13] Wojciech Golab, Danny Hendler, Vassos Hadzilacos, and Philipp Woelfel. Rmr-efficient implementations of comparison primitives using read and write operations. *Distributed Computing*, 2012. To appear.

[14] Wojciech Golab, Lisa Higham, and Philipp Woelfel. Linearizable implementations do not suffice for randomized distributed computation. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 373–382, 2011.

[15] Danny Hendler and Philipp Woelfel. Adaptive randomized mutual exclusion in sub-logarithmic expected time. In *Proceedings of the 29th SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 141–150, 2010.

[16] Danny Hendler and Philipp Woelfel. Randomized mutual exclusion with sub-logarithmic RMR-complexity. *Distributed Computing*, 24(1):3–19, 2011.

[17] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufman, 2008.

[18] Prasad Jayanti. Adaptive and efficient abortable mutual exclusion. In *Proceedings of the 22nd SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 295–304, 2003.

[19] Prasad Jayanti, Srdjan Petrovic, and Neha Narula. Read/write based fast-path transformation for FCFS mutual exclusion. In *31st Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, pages 209–218, 2005.

[20] Yong-Jik Kim and James H. Anderson. A time complexity bound for adaptive mutual exclusion. In *Proceedings of the 15th International Symposium on Distributed Computing (DISC)*, pages 1–15, 2001.

[21] Yong-Jik Kim and James H. Anderson. Nonatomic mutual exclusion with local spinning. *Distributed Computing*, 19(1):19–61, 2006.

[22] John M. Mellor-Crummey and Michael L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Trans. Comput. Syst.*, 9(1):21–65, 1991.

[23] John M. Mellor-Crummey and Michael L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Transactions on Computer Systems*, 9(1):21–65, 1991.

[24] Jae-Heon Yang and James H. Anderson. A fast, scalable mutual exclusion algorithm. *Distributed Computing*, 9(1):51–60, 1995.

[25] Andrew Chi-Chih Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.