Tel-Aviv University
The Raymond and Beverly Sackler
Faculty of Exact Sciences
School of Mathematical Sciences

Bounds in the shared memory requirements for Long-Lived & Adaptive objects

Thesis Submitted as part of the requirements
towards M.Sc. degree in Tel-Aviv University
School of Mathematical Sciences
Department of Computer Science

Presented by
Pazi Boxer

This work was carried out under the supervision of
Prof. Yehuda Afek and
Dr. Dan Touitou

January 2001

Acknowledgments

I would like expressing gratitude to my supervisors, Prof. Yehuda Afek and Dr. Dan Touitou, for their guiding, patience and encouragement.

I am in debt to Gideon Stupp for essential discussions and collaboration during the early phases of this research. I would like also to thank Hagit Attiya, Arie Fouren, Eli Gafni, Michael Merritt and Gadi Taubenfeld for many helpful discussions.

I would also like to thank my family for their support.

Abstract

In this paper[1] we prove:

For any constant d there is a large enough n such that there is no long-lived adaptive implementation of collect, renaming or snapshot in the read write model with n processes that uses d or less MWMR registers.

In other words, there is no implementation of a long-lived and adaptive renaming, collect or snapshot object in the atomic read/write model that uses O(1) multi-writer-multi-reader registers and any number of single-writer-multi-reader registers. In 1980 Burns and Lynch [2] proved that at least n multi-writer-multi-reader (MWMR) registers are necessary in any mutual exclusion algorithm that uses only MWMR registers (i.e., atomic registers). It is also relatively easy to see that any adaptive non-trivial algorithm uses at least one multi-writer-multi-reader (MWMR) register even when there are n single-writer-multi-reader (SWMR) registers. Here we extend the techniques of Burns and Lynch and prove that adaptive algorithms that use both SWMR and MWMR registers such as, collect, snapshot and renaming, need in addition to the $\Omega(n)$ SWMR registers a non-constant, $F(n)$ number of MWMR registers.

[1] A preliminary version of this paper was presented in PODC 2000 [1]

Contents

1 Introduction

In recent years, several different long-lived adaptive algorithms whose operation step complexity is a function of the actual number of processes running concurrently with the operation execution, have been presented [3, 4, 5]. Regardless of the type of adaptiveness and which object was implemented, they all require $\Omega(n)$ MWMR registers in their implementation, where n is the total number of processes that *may* participate in the algorithm/implementation. Motivated by the question whether n MWMR registers are a necessity we prove here that the number of MWMR registers used by either long-lived adaptive renaming, snapshot or collect must be a function of n. Specifically, for any constant d there is a large enough n_d such that every long-lived adaptive read/write implementation of collect (snapshot and renaming) with n_d processes must use at least d MWMR registers.

The main proof in this paper is given for a simple object called *weak test and set* [6], or WT&S in short. Below we give long-lived and adaptive reductions from collect, snapshot and adaptive renaming objects to WT&S, thus proving our main theorem. A weak test and set object (fully specified in Subsection 2.2) behaves like a test-and-set object except that if a process tries to acquire the test-and-set bit and at least one more process is running concurrently with it, then it may fail. That is, if several processes invoke a weak-test-and-set operation concurrently, they may *all* fail, leaving the test-and-set bit vacant, un-captured. Proving the main result for a simple and weak object like WT&S enables us to reduce the result to other objects, as we show below for renaming, snapshot and collect.

In proving the impossibility of an adaptive long-lived WT&S object, we follow the Burns and Lynch proof method which is extended here in a non-trivial way. The proof constructs a run in which two processes p and q win the test-and-set bit at the same time. At some point in the constructed run process p captures the weak-test-and-set bit, then a set of processes write on all the MWMR registers written by p erasing all traces of p. This enables another process q to acquire the bit, violating the exclusion property of weak-test-and-set.

The operational semantics (specification) of a collect object is as follows: there are n single-writer-multi-reader registers $C_1, \ldots, C_n$, one for each process. To write in the collect object process P_i writes its value in C_i. To perform a collect operation a concurrent process simply reads the n registers and returns the vector of values read.

Reduction from collect (snapshot) to WT&S: To implement a WT&S object from an adaptive collect (snapshot) object a process simply writes "active" in the collect (snapshot) object and performs an (adaptive) collect (snapshot) operation (Initially all the values in the collect (snapshot) object are "asleep"). If it observes only itself with an "active" value in the collected set (snapshot view), it wins the test and set bit. Otherwise it writes "asleep" and fails in the WT&S operation. A winning process later releases the test and set bit by writing "asleep" in the collect (snapshot) object.

Reduction from adaptive renaming to WT&S: In the *long-lived adaptive renaming* problem, processes repeatedly acquire and release distinct names; A name acquired by a process must be in the range $\{1, \ldots, f(k)\}$, where k is the number of processes that hold names or are actively acquiring a name concurrently with the process that acquires a new name. In particular, for any adaptive renaming the size of a name acquired when a process runs solo (i.e., when no process holds a name or tries to acquire a name) is bounded by some constant $f(1)$. In a long lived and adaptive WT&S implementation from adaptive renaming we use one copy of the adaptive renaming object and an array of $f(1)$ Boolean MWMR registers. A process performs the WT&S operation by first acquiring a name and checking if the name size is smaller or equal to $f(1)$. If the name size is larger the process has failed the WT&S and releases the name it has acquired. Otherwise, it verifies that it is the only process holding a name smaller or equal to $f(1)$. To perform this verification it writes "active" in the MWMR register that corresponds to the name it got, and reads all the $f(1)$ MWMR registers. If all other registers are read "asleep" the process safely wins the WT&S operation. To reset the weak-test-and-set bit a process writes "asleep" in the corresponding MWMR register, and releases the name it had acquired.

Notice that obviously there is no adaptive implementation of a multi-writer-multi-reader register from a system of single-writer-multi-reader registers. This follows because such an implementation uses only single-writer registers. Hence, to read a value written into the implemented multi-writer register a process would have to read all of them or otherwise it would miss the last value written.

Notice also that the long-lived and adaptive implementation of collect in [4] uses $O(n)$ MWMR registers, thus leaving a gap between the linear upper and the non-constant lower bound on the number of MWMR registers necessary for their adaptive implementation of a collect object.

Further notice that there is an adaptive *one-shot* ϵ-agreement algorithm that uses only a constant number of shared (MWMR) registers [7]. The number of shared registers in that algorithm is inversely proportional to ϵ.

The model and basic definitions are given in Section 2. In Section 3 an informal topdown description of the proof is provided. The full proof of the theorem and the main lemma are given in a bottom-up manner in Section 4. Concluding remarks are given in Section 5.

2 The Model

Our model is described in Subsection 2.1. The model is an asynchronous shared memory system with read/write registers and n processes. There are d multi-writer multi-reader registers and the rest are single-writer multi-reader registers. In Subsection 2.2, we define the WT&S object and finally in Subsection 2.3 we provide the required definitions.

2.1 Asynchronous Shared Memory Systems

In this section, we characterize asynchronous shared memory systems [8, 9], which support atomic read and atomic write operations. We start with a description of the notion of a protocol.

Definition 2.1 *An n-process protocol $W = (C, P, R)$ consists of a non-empty set C of runs, an n-tuple $P = (p_1, \ldots, p_n)$ of processes and an m-tuple $R = (r_1, \ldots, r_m)$ of registers.*

An event *corresponds to an atomic step performed by a process. Here we consider only the following types of events:*

- $read_p(r, v)$ *- process p reads the value v from register r,*
- $write_p(r, v)$ *- process p writes the value v into register r.*

In our model, R consists of a set of Multi-Writer and Multi-Reader registers (MWMR) and a set of Single-Writer and Multi-Reader registers (SWMR).

- A *system state* consists of the state of all processes and the value of all registers in the system. A system has one or more *initial* system states in which the system starts its execution.
- A *run* α is a finite or infinite sequence of events, starting from an initial system state. When the sequence is finite, we also say that the run is a *finite run*. When the sequence is empty, the run is called *empty run*.
- A *run segment* x_α of run α is a finite, continuous subsequence of events of α.

2.2 Weak Test & Set

In a similar way to a mutual exclusion object [10, 2, 11, 12] the behavior of a WT&S object is modeled by the following program. Each process is in one of four possible *WT&S states* in this program: thinking, WT&SET, eating and RESET.

TS: object of type WT&S
Process p**:**
repeat forever
    thinking_section$_p$
    tbit := WT&SET$_p$(TS)
    if tbit=0 {
        eating_section$_p$
        RESET$_p$(TS) }
end repeat forever

A WT&S implementation must satisfy the following two properties:

Exclusion: At most one process is eating at any point (system-state) of the execution.

Solo: If a process becomes hungry (goes out of the thinking state), while all other processes are thinking, and runs alone (it is the only process that takes steps) then it must eventually start eating.

The WT&S is a weaker object than the test & set object [13], since it allows two processes to fail, when they are trying to captutre the test & set bit concurrently, while in the test & set object one of them must succeed.

2.3 Preliminaries and definitions

In the following definitions, we assume a protocol W that implements the WT&S object.

- We say that a system state s is an *idle-state*, if all the processes are thinking at s.

  (In our model, each initial system state is also an idle-state.)

- A *solo run segment* of p is a run segment, starting in an idle-state, in which only p takes steps.

- A *p-segment* is a run segment starting in a state s in which process p is thinking, and in which only p takes steps. Notice that a p-segment is not necessarily a p solo run segment.

- We say that p cannot *differentiate* between two run segments x, y starting at s_x and s_y in which p participates, iff whenever p takes steps in x and in y, p is in the same WT&S state and it reads the same values from registers.

- A state s is *transparent* with respect to process p, if there is a p-segment starting at s, that p cannot differentiate from some solo run segment of p starting at an idle-state in some other run and ending with p eating. This specific run segment is called the *longest pseudo-solo run segment* of p, which starts at s.

  Similarly, state s is transparent with respect to a set of processes Q, if s is transparent $\forall p \in Q$.

- All prefixes of the longest pseudo-solo run segment that start at state s, are called *pseudo-solo run segments.*

- We say that process p cannot *distinguish* between states s and s', if p cannot differentiate between two p-segments that start after s and s' and end with the next time that p is thinking.

  Similarly, a set of processes Q cannot *distinguish* between states s and s', if $\forall p \in Q$, p cannot *distinguish* between states s and s'.

- We say that a register r is *covered* by process p at the end of run α, if the next step of p after run α is a write to r.

- We say that a process p is *active* in run α, if the state of p at the end of run α is not thinking.

- Given a run segment x and a state s, $s \cdot x$ denotes the concatenation of x after some run α which ends at s, assuming that such an α exists.

A WT&S implementation is *adaptive* [14, 15], if the number of primitive operations (i.e., read/write operations) is bounded by a function of the contention during the high-level operation (either WT&SET or RESET) execution. Since we are proving an impossibility result, the weaker notion of contention gives us a stronger result. Therefore, we consider the contention to be the *interval* contention [14, 15], which is the total number of processes that are active during the time interval of the high-level operation execution.

Clearly, if an implementation of an object is adaptive, then there is a constant k such that, no solo run segment has more than k steps. I.e., in any solo run of a process (all other processes are thinking) the implemented WT&SET and RESET operations finish within k primitive operations. We say that the algorithm is *"k-solo-adaptive"*.

Moreover, if for some constant k', there are k' processes performing the WT&S implementation together, then each will finish its WT&SET and RESET operations within a constant number of steps.

An adaptive algorithm is also a *wait-free* algorithm, since each process must terminate its execution within a finite number of steps.

W.l.o.g, we assume that each process has only one single-writer multi-reader (SWMR) register. All the SWMR registers of a process can always be replaced by one large SWMR register.

We say that a condition or property holds in a run, if it holds at the end of the run (unless we specifically say otherwise).

3 Informal description of the theorem and its proof

3.1 The Theorem

Theorem 3.1 *For any constants d and k, there is a large enough number $n_{d,k}$ s.t., there is no long-lived k-solo-adaptive WT&S implementation in a system with $n_{d,k}$ or more processes, and with d or less Multi-Writer Multi-Reader registers.*

Corollary 3.2 *For any constant d, there is a large enough n_d s.t., there is no long-lived adaptive WT&S implementation in a system with n_d processes, that uses d or less Multi-Writer Multi-Reader registers.*

Proof of Corollary 3.2: By contradiction, assume that there is an adaptive implementation of WT&S object that uses d MWMR registers and works for any number of processes. Then consider one such implementation, and let k be its k-solo-adaptive parameter. By Theorem 3.1, for $n > n_{d,k}$ no such implementation exists, a contradiction. ■

The following corollary follows from Corollary 3.2 using the reductions from Renaming, Collect and Snapshot objects to WT&S object, presented in the introduction.

Corollary 3.3 *For any constant d, there is a large enough n'_d s.t., there is no long-lived adaptive renaming, collect or snapshot objects implementation in a system with n'_d processes, that uses d or less Multi-Writer Multi-Reader registers.*

Henceforth, the rest of this paper is the proof of Theorem 3.1. First, in Subsections 3.2 and 3.3 we provide an overview of the proof of this theorem. Then in Section 4 the full proof of Theorem 3.1 is given.

3.2 Sketch of Proof of the theorem

The proof is by a way of contradiction. We first assume that there is a WT&S implementation with d MWMR registers for a system with more than $n_{d,k}$ processes. Then, we show that under these conditions there is a run in which two processes, p and q are in the critical section (i.e., eating), at the same time. There are 6 steps in the construction of this run, where by far, the most difficult step is the first step. (See Figure 1, which illustrates this construction. In the full proof, this run is constructed by Lemma 4.6, which is called the Violation Lemma. We link the outline described here with the steps of the run constructed in the proof of the Violation Lemma in Section 4.5.) The six major steps of the proof are:

1. We construct a run prefix α' s.t., the state at the end of α' is transparent with respect to p, and every MWMR register that p writes in its longest pseudo-solo run starting at α', is covered. (See step 1 in Figure 1 and see step 1 in Lemma 4.6, where we denote $\alpha' = \beta_k$.)

2. Let $^{\sim}solo_p$ be the longest pseudo-solo run segment of process p, starting after α'. Hence p is eating in $\alpha' \cdot^{\sim} solo_p$. Let $\{r_1, ..., r_{k'}\}$ be the set of MWMR registers written by p in $^{\sim}solo_p$, where $k' \leq k$. (See step 2 in Figure 1 and see step 4 in Lemma 4.6.)

3. We continue the construction with a run segment *clean* by enabling the covering writes in the following way: $\forall r_i\ i = 1, ..., k'$, we let the process u_i that covers r_i at $\alpha' \cdot^{\sim} solo_p$ write. After all registers in $\{r_1, ..., r_{k'}\}$ are covered, let each process u_i continue its execution until it reaches the thinking state. I.e., in $\alpha' \cdot^{\sim} solo_p \cdot clean$ all the MWMR registers written by p have been overwritten. (See steps 3 and 4 in Figure 1 and see step 2 in Lemma 4.6, where we denote $clean = clean_{c+1}$.)

4. In the full proof in Lemma 4.6, both the set $\{u_i\}\ i = 1, ..., k'$ and process p are carefully selected in such a way that none of the processes u_i reads the single-writer multi-reader register R_{SW_p} that p writes during $^{\sim}solo_p$. Thus, all the MWMR registers that p writes are overwritten and the only trace of p, the content of its SWMR register, is ignored by processes $u_1, ..., u_{k'}$. Therefore, in $\alpha' \cdot^{\sim} solo_p \cdot clean$ all the MWMR registers written by p have been overwritten, and have the same values that they have in some idle state. (See steps 3 and 4 in Figure 1 and see steps 3 and 5 in Lemma 4.6.)

5. We show that there are processes p and q s.t., run $\alpha' \cdot^{\sim} solo_p \cdot clean$ is transparent with respect to q. This is done by carefully selecting processes p and q in such a way that q does not read the SWMR register that p writes in $^{\sim}solo_p$. Therefore, there is a pseudo-solo run segment $^{\sim}solo_q$ starting after $\alpha' \cdot^{\sim} solo_p \cdot clean$ in which q is eating. (See step 5 in Figure 1 and see step 6 in Lemma 4.6.)

6. At the end of $\alpha' \cdot^{\sim} solo_p \cdot clean \cdot^{\sim} solo_q$, both p and q are in the critical section, a contradiction.

In the full proof in Lemma 4.6, we show that the size of $n_{d,k}$ is a function of d and k, and these constants may be different for different implementations.

The sketch of proof for the inductive construction of run α', is described in the next section. This proof is associated to Lemma 4.5 and the full proof is provided in Section 4.

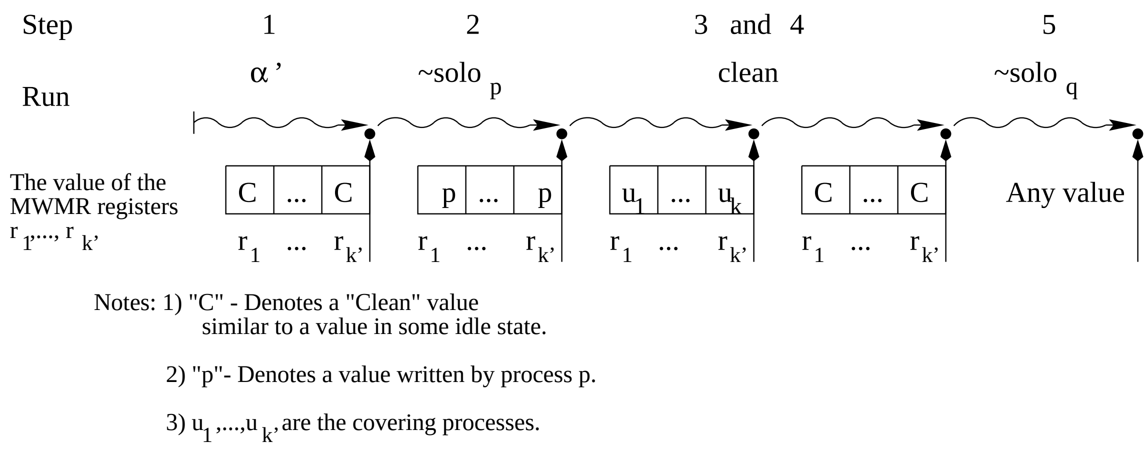

Figure 1: Illustration of the proof of Theorem 3.1.

3.3 Sketch of Proof of the Main Lemma

In the main lemma we construct the required run α', by using several techniques.

For explanatory reasons only and only in this section, we separate these techniques by defining assumptions, starting with very strong assumptions, which are later relaxed until removing all the assumptions. This is done in three phases, where in each phase we define the lemma again in an informal manner (see Lemmas 3.4, 3.5, 3.6 and 3.7). The formal proof of the main lemma (Lemma 4.5) without any assumption is provided in Section 4.4. We try to link between the outline described here and the steps of the proof of Lemma 4.5.

Recall W, d and k from the Model Section above:

Assumption A: There are no write operations to SWMR registers in all legal runs. That is, we assume for the moment that there is a wait-free algorithm that uses only d MWMR and no SWMR registers.

Assumption B: If G is a set of processes and s is a state which is transparent with respect to G, then during their longest pseudo-solo run segments starting at s all the processes in G write in the same MWMR registers at the same order.

Although, under assumption A, the impossibility result presented in this paper follows directly from Burns & Lynch [2], we still provide it here, since the proof serves as the basis for our final proof.

Phase 1:

In the following lemma, we construct α', as presented in step 1 in Section 3.2. In Lemma 3.4 below, run α' is denoted by $s \cdot \beta$ and it satisfies the properties of α'. Property (1): the state at the end of run $s \cdot \beta$ is transparent with respect to some set of processes called G_e, and property (2): there is a cover on all the MWMR registers written by processes in G_e in all their pseudo-solo run segments, starting after $s \cdot \beta$. The size of the set G_e is a parameter and it is determined in the proof of Theorem 3.1.

Lemma 3.4 *Let W be a long-lived k-solo-adaptive WT&S algorithm satisfying assumptions A and B with at most d MWMR registers. Then for any constant e, for any idle state s and for any set of processes G, there exists a constant n_e s.t., if $|G| \geq n_e$, then there is a run segment β and a set of processes $G_e \subseteq G$ s.t., the following holds: (1) $|G_e| \geq e$, (2) all the MWMR registers written in the longest pseudo-solo run segments of processes in G_e, after $s \cdot \beta$, are covered in $s \cdot \beta$, and (3) the state at the end of $s \cdot \beta$ is transparent with respect to G_e.*

Sketch of proof of Lemma 3.4: The proof is by induction on i. (Notice that in the full proof, we state only an inductive lemma, which is Lemma 4.5. Further notice that in the inductive lemma, the state s cannot be an idle-state, since in each induction step there are some active processes that are covering the MWMR registers.) The inductive claim is:

Lemma 3.5 *Let W be a long-lived k-solo-adaptive WT&S algorithm satisfying assumptions A and B with at most d MWMR registers. Then for every i, $0 \leq i \leq k$, for any constant e and for any state s transparent with respect to a set of processes G, there exists a constant $N(e, i)$ s.t., if $|G| \geq N(e, i)$, then there is a run segment β_i and a set of processes $G_i \subseteq G$ s.t., the following holds: (1) $|G_i| \geq e$, (2) either the first i MWMR registers written in the longest pseudo-solo run segments of processes in G_i, after $s \cdot \beta_i$, are covered in $s \cdot \beta_i$, or, the longest pseudo-solo run segments starting at $s \cdot \beta_i$ have less than i writes to MWMR registers and all these writes are covered in $s \cdot \beta_i$ and (3) the state at the end of run $s \cdot \beta_i$ is transparent with respect to G_i.*

Sketch of proof of Lemma 3.5 : The proof is by induction on i. (See Figure 2, which illustrates the construction of run segment β_{j+1} until step 8, for the induction step $i = j+1$. In the full proof, this run is constructed by Lemma 4.5, which is called the Main Lemma. We link the outline described here with the steps of the run constructed in the proof of the Main Lemma in Section 4.4.)

1. For $i = 0$ the claim holds trivially by setting $G_0 = G$, $N(e, 0) = e$ and β_0 to be the empty sequence.

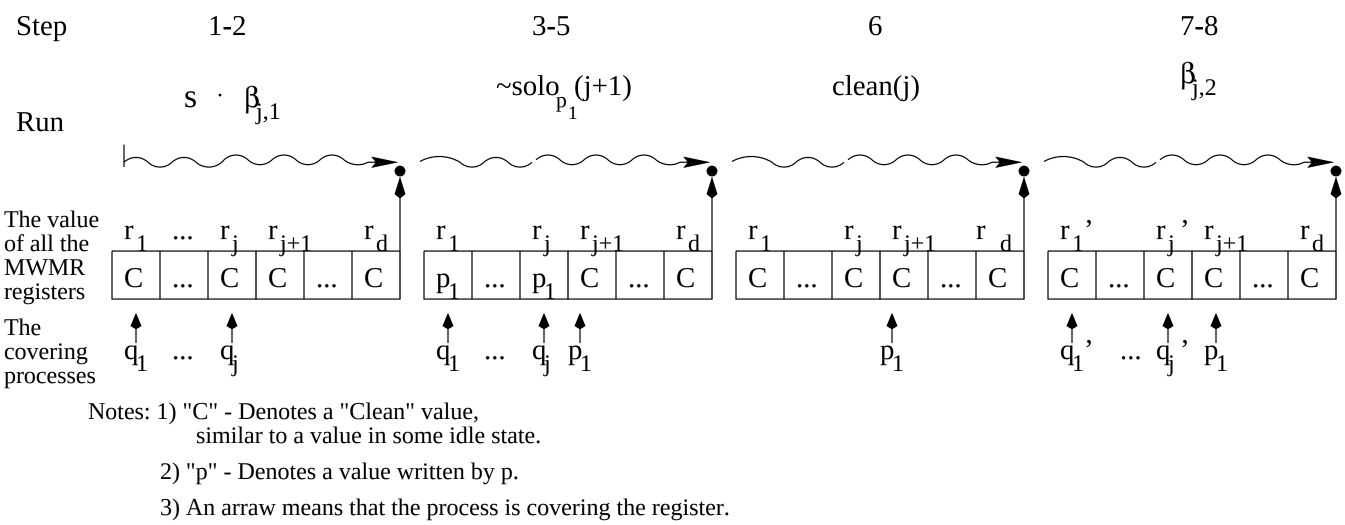

Figure 2: Illustration of the memory state in proof of Lemma 3.5 steps 1- 8 (run β_{j+1}).

2. We assume that the claim holds for $i = j$ and we show that it holds for $i = j + 1$. Assume that G is a large enough set of processes s.t., $|G| \geq N(e, j + 1)$ and s is a state transparent with respect to G. By the induction hypothesis, there exist a run segment β_j that satisfies all the three conditions with respect to a set of processes G_j. We denote $\beta_{j,1} = \beta_j$ and $G_{j,1} = G_j$.

   By these conditions, at the end of $s \cdot \beta_{j,1}$, the first j MWMR registers written in the longest pseudo-solo run segments of processes in $G_{j,1}$, are covered and $s \cdot \beta_{j,1}$ is transparent with respect to $G_{j,1}$. By assumption B, let $(r_1, ..., r_j)$ be the first j MWMR registers written in the longest pseudo-solo run segments of processes in $G_{j,1}$. (See steps 1-2 in Figure 2 and see step 1 in Lemma 4.5, where we denote $s = \alpha$. Note that the state s has the same meaning as run α, since we are interesting in the state at the end of run α.)

3. Select any process $p_1 \in G_{j,1}$ and a pseudo-solo run segment $^{\sim}solo_{p_1}$ starting at $s \cdot \beta_{j,1}$.

4. If there are less than $j + 1$ write operations to MWMR registers in $^{\sim}solo_{p_1}$ (until p_1 eats), then we are done and $\beta_i = \beta_{j,1}$ for $i = j, j + 1, \ldots k$ (I.e., if p_1 does not have more than j write operations to MWMR registers, then none of the processes in $G_{j,1}$ has a $j + 1st$ write to a MWMR register, since, by Assumption B, they all write in the same MWMR registers in the same order, until eating). (See step 2 in Lemma 4.5.)

5. Otherwise, p_1 has at least $j + 1$ write operations in MWMR registers in $^{\sim}solo_{p_1}$, before eating. Let operation (event) $W_{p_1,j+1}$ be the $j + 1st$ write to a MWMR register by process p_1. Let r_{j+1} be the register that process p_1 writes in its $W_{p_1,j+1}$, and let

$^{\sim}solo_{p_1}(j+1)$ be the longest prefix of $^{\sim}solo_{p_1}$ that does not include $W_{p_1,j+1}$. Any pseudo-solo run segment that starts after $s \cdot \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1)$ can read p_1's write operations to $r_1, ..., r_j$ and therefore, the state is not transparent. (Notice that because of assumption B, p_1 is enabled to write in the same MWMR register r_{j+1} as all processes in $G_{j,1}$ in their pseudo-solo run segments, starting at $s \cdot \beta_{j,1}$.) (See steps 3-5 in Figure 2 and see step 5 in Lemma 4.5, where we denote $p_1 = p$.)

6. To erase all the traces of p_1, we next enable each of the processes $q_1, ..., q_j$ which are covering $r_1, ..., r_j$ respectively to write in $r_1, \ldots, r_j$ and to run until they reach a thinking state. We denote this run segment as $clean(j)$. Such a $clean(j)$ extension to $s \cdot \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1)$ exists, since W is a wait-free WT&S implementation. Therefore, each process $q_1, ..., q_j$ reaches its thinking state after a finite number of steps. Let $\alpha_{j+1,1} = \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1) \cdot clean(j)$. Then, by the end of run $s \cdot \alpha_{j+1,1}$ the state is transparent with respect to $G_{j,1} - \{p_1\}$. Here we use assumption A, which permits write operations only in MWMR registers, otherwise, later pseudo solo executions could observe p_1 by reading its SWMR register. By the end of run $s \cdot \alpha_{j+1,1}$, only one process, p_1, is enabled to write in r_{j+1}, while the registers $r_1, ..., r_j$ are not necessarily covered any more. (See step 6 in Figure 2 and see step 3 in Lemma 4.5, where we denote $clean(j) = clean_{c+1}$.)

7. Since the state at $s \cdot \alpha_{j+1,1}$ is transparent with respect to $G_{j,1} - \{p_1\}$, we can activate the induction hypothesis again. Hence a cover of a sequence of j MWMR registers is constructed by first activating (by the induction) a run segment $\beta_{j,2}$ s.t., the state at the end of $s \cdot \alpha_{j+1,1} \cdot \beta_{j,2}$ is transparent to a set of processes $G_{j,2} \subseteq G_{j,1} - \{p_1\}$. Let $r'_1, ..., r'_j$ be the new j MWMR registers covered at the end of $s \cdot \alpha_{j,1} \cdot \beta_{j,2}$ and let $q'_1, ..., q'_j$ be the processes covering them. (See steps 7-8 in Figure 2 and see step 11 in Lemma 4.5.)

REPEAT-COVER technique:

8. Let $r'_1, r'_2, ..., r'_j, r'_{j+1}$ be the first $j+1$ registers written in the pseudo-solo segments of the processes in $G_{j,2}$. Assume that $r'_{j+1} = r_{j+1}$. In such a case, we are done, since $r'_1, r'_2, ..., r'_j$ are covered by processes $q'_1, ..., q'_j$ and r_{j+1} is covered by process p_1. (See steps 7-8 in Figure 2 and see step 12 in Lemma 4.5.)

9. If however $r'_{j+1} \neq r_{j+1}$, (because the pseudo solo runs in $\beta_{j,2}$ do not necessarily write to the same registers as the pseudo solo runs in $\beta_{j,1}$) then we construct the required cover on r'_{j+1}, by repeating steps 3- 7 above, but starting at $s \cdot \alpha_{j+1,1} \cdot \beta_{j,2}$. I.e., we keep process p_1 covering r_{j+1} and extend the run with a segment in which a process

$p_2 \in G_{j,2}$ is covering r'_{j+1}. After repeating steps 3- 7 once, we have a cover on both r_{j+1} and r'_{j+1}, by two different processes, p_1 and p_2 in the run $s \cdot \alpha_{j+1,1} \cdot \alpha_{j+1,2} \cdot \beta_{j,3}$. We repeat and construct a new cover on another $j + 1st$ register, until either one of the following: we finish in step 4, or in step 8, or we repeat until all MWMR registers are covered, i.e., up to at most d times.

By the termination condition of this loop, there is now a cover on all $j + 1$ MWMR registers written by the current collection of pseudo solo runs. Since d is a constant, the run we have constructed is of constant length and there are enough processes to construct it with the repeating covers, since G is a large enough set. (See steps 12 and 13 in Lemma 4.5.)

10. In conclusion for $i = j + 1$, $\beta_i = \beta_{j+1,d} = s \cdot \alpha_{j+1,1} \cdots \alpha_{j+1,d} \cdot \beta_{j,d+1}$ the concatenation of all repeating run segments in steps 3- 7 and $G_{j+1} = G_{j,d+1}$ and $|G_{j+1}| \geq e$, (which is the set G after removing from it the processes participating in $\beta_{j+1,d}$).

In the full proof in Lemma 4.5 in step 15, we show that $N(e, i)$ is a function of d, i, k, e and the number of active processes in the run.

■

Phase 2:

Assumption B is relaxed in this phase, i.e., during WT&S operations processes are now allowed to write to different MWMR registers at different orders. The proof in phase one fails when assumption B is relaxed in the following way. The cover we have constructed in phase one is not necessarily on the same MWMR registers that processes like p_1 or p_2 are going to write to (i.e., each is allowed to write to a different set of registers).

The technique to overcome this difficulty relies on two factors, (1) the number of MWMR registers d is a constant, (2) the algorithm is a k-solo-adaptive algorithm i.e., the maximum length of all longest pseudo-solo run segments is k. Therefore, at a given transparent state s with respect to a large enough set of processes G, we consider the different sequences of write operations to MWMR by the different longest pseudo-solo run segments of processes in G. The number of different sequences of write operations to MWMR registers is bounded by a function of d and k. Each such possible sequence defines an equivalent class in G. By taking the largest equivalent class of processes we obtain a large enough subset of G such that each process in the subset performs the same sequence of writes to MWMR registers in its longest pseudo-solo run segment, starting at s. Such a subset is said to satisfy the *MULTI-unique* property. (For a definition of the Multi-unique property, the reader can refer to Definition 4.1 in Section 4.2.)

Notice that at two different states s and s' that are transparent with respect to a set of processes G, any process in G may have two different sequences of write operations to MWMR registers in its longest pseudo-solo runs from s and from s'. Therefore, we cannot compute in advance the required sub-set of processes. Instead, we compute the subset that corresponds to each state as described below.

We rewrite Lemma 3.5 without assumption B as follows:

Lemma 3.6 *Let W be a long-lived k-solo-adaptive WT&S algorithm satisfying assumption A with at most d MWMR registers. Then for every i, $0 \leq i \leq k$, for any constant e and for any state s transparent with respect to a set of processes G, there exists a constant $N(e,i)$ s.t., if $|G| \geq N(e,i)$, then there is a run segment β_i and a set of processes $G_i \subseteq G$ s.t., the following holds: (1) $|G_i| \geq e$, (2) either the first i MWMR registers written in the longest pseudo-solo run segments of processes in G_i, after $s \cdot \beta_i$, are covered in $s \cdot \beta_i$, or, the longest pseudo-solo run segments starting at $s \cdot \beta_i$ have less than i writes to MWMR registers and all these writes are covered in $s \cdot \beta_i$ and (3) the state at the end of run $s \cdot \beta_i$ is transparent with respect to G_i.*

Sketch of proof of the inductive step in Lemma 3.6:

All steps of the proof, but steps 3, 4 and 5 are the same as in phase one. Steps 4 and 5 are the same as in phase one, but here instead of referring to assumption B, we argue by using the MULTI-unique property, as explained below. Unless stated otherwise we use the same notations and symbols as in phase one.

MULTI-unique: In step 3 of phase one, before selecting process p_1, we should first reduce the set $G_{j,1}$ to a sub-set $\hat{G}_{j,1}$. The set $\hat{G}_{j,1}$ is selected s.t., every process in this set is writing in the same MWMR registers and at the same order, when executing its longest pseudo-solo run segment, starting after $s \cdot \beta_{j,1}$.

This construction is associated with Claim 4.2 in Section 4.3 and the details are as follows: In each iteration of steps 3- 7 in each inductive step the corresponding set $\hat{G}_{j,1}$ (which was $G_{j,1}$ in previous phase) is constructed in several rounds. In each round we select a sub-set of the subset selected in the previous round. Let Q_0 stand for $G_{j,1}$ and let the set of processes after round h be $Q_h \subseteq Q_0$. In round $h+1$, we observe the $h+1$'st write operation in all the longest pseudo-solo run segments of processes in Q_h (the previous round), starting after $s \cdot \beta_{j,1}$ and we construct the new set Q_{h+1}, as follows:

1. If more than half of the processes do not have the longest pseudo-solo run segment with $h+1$ write operations in MWMR registers before eating, then we define Q_{h+1} to be these processes (i.e., $|Q_{h+1}| \geq \frac{1}{2}|Q_h|$) and Q_{h+1} is $\hat{G}_{j,1}$, the set of processes that holds the Multi-unique property at $s \cdot \beta_{j,1}$.

2. Else, we denote the rest of the processes by Q'_{h+1} and $|Q'_{h+1}| \geq \frac{1}{2}|Q_h|$. Thus, processes in Q'_{h+1} have an $h+1$'st write operation in MWMR registers. Therefore, there is at least one MWMR register that a set of size $\frac{|Q'_{h+1}|}{d}$ is writing in it. The recursive function is $|Q_{h+1}| \geq \frac{|Q'_{h+1}|}{d} \geq \frac{|Q_h|}{2d}$.

Since there are at most k rounds, then $|Q_k| \geq \frac{|Q_0|}{[2d]^k}$. Finally, we denote $\hat{G}_{j,1} = Q_k$ thus, $|\hat{G}_{j,1}| \geq \frac{|G_{j,1}|}{[2d]^k}$, (since $Q_0 = G_{j,1}$).

■

Phase 3:

Here assumption A is relaxed i.e., during the WT&S operations processes may write in their SWMR registers. The run constructed in the proofs of Lemma 3.5 and Lemma 3.6, may not be valid anymore, as processes are allowed to write in their SWMR registers. Consider the case in the run in which a SWMR register of an active (i.e., not thinking) process p, is read by another process q. Thus, process q can point out at p (e.g., by leaving in the MWMR registers a "look at p's SWMR register" instruction). Therefore, this run cannot reach a transparent state as long as p is active. The reasoning presented in step 6 may collapse in one of the following two ways: (See step 6 in Lemma 4.5.)

(1) The participating processes in any *clean* run segment may read the SWMR registers of other active processes, like process p. By this, leaving the system in a non-transparent state with respect to none of the processes. For more details see case 1 in the sketch of proof, below. (See also step 7 in Lemma 4.5.)

(2) After some *clean* run segment, a process q starts its q-segment execution and it may read a SWMR register of another active process p. Thus, q does not perform a pseudo-solo run segment any more, instead it may stop its execution without covering the MWMR registers. In addition q may write in different MWMR registers, because the Multi-unique property is defined only for pseudo-solo run segments. For more details see case 2 in the sketch of proof, below. (See also step 8 in Lemma 4.5.)

These problematic situations are avoided here by preventing from processes whose SWMR registers are later read from taking part in the constructed run. Therefore, in any given state in the run (of phase one), if process q reads the SWMR register of process p and p is active, then we construct another run in which p is replaced by another process p'. This replacement is such that process q still reads the same SWMR registers, i.e., the register of p and not of p'. Let β be the run in which p participates and let β' be a new run in which another process p' is participating instead of p. Process p''s behavior in β' is somehow "equivalent" to p's behavior. That is p and p' are writing and covering the same MWMR registers in β and β', respectively. We show that a process like p' always exists because: (1) There is a large enough set of processes to select p' from s.t., p' did not participate in the run before. This is because only a function of d and k processes are participating in the run constructed in phase one. (2) Process q can perform only a constant number of read operations, since the number of concurrently active processes in the run is a function of d and k and since the algorithm is adaptive. In the full proof, we define the "replacement" operation from run β to run β' in Definition 4.3, which is called *Multiple-Transitions*.

During the inductive proof below, we maintain a large enough set of "equivalent" runs, which allow us to replace the run at any point in which we fail to reach a transparent state. This set is reduced as the construction of the run progresses. In the full proof, we show how to construct this set of runs from a given run and a set of processes. In our proof, if a run β satisfies several conditions with respect to a set of processes G, then we can construct a set of "equivalent" runs. Constructing a new run β' from β is by replacing the problematic processes (like p above) with other processes from G. Thus, as much as G is a larger set, then there are more runs to construct. During the proof, the set G is getting smaller and there are less "equivalent" runs. In this outline, we assume that there is a set of "equivalent" runs and we show how it can be used to solve the two problems explained above. The method of constructing "equivalent" runs from a given run, is described in the full proof in Lemma 4.4.

Definition 3.1 *Two runs β and β' are equivalent with respect to a set of processes G if (1) the state at the end of both runs β and β' is transparent with respect to G, (2) the sets of MWMR registers covered in β and β' are the same, and (3) if process p participates in both β and β' then p cannot differentiate between the two runs.*

While constructing a run in the proof, whenever a process p which had been selected to participate in the run is discovered by covering processes, we need to replace it with some other process p' that cannot be discovered. This is achieved by considering an equivalent run in which p' takes steps instead of p. We therefore restate inductive Lemma 3.5 and Lemma 3.6 as follows: (Notice that instead of state s, we state the lemma with a run α, since we may need to change the active processes at s and the equivalence property was defined only for runs.)

Lemma 3.7 *Let W be a long-lived k-solo-adaptive WT&S algorithm with at most d MWMR registers. Then for every i, $0 \leq i \leq k$, for any constant e and for any run α transparent with respect to a set of processes G, there exists a constant $N(e,i)$ s.t., if $|G| \geq N(e,i)$ and there is a large set of runs* equivalent *to α with respect to G, then there is a run segment β_i and a set of processes $G_i \subseteq G$ s.t., the following holds: (1) $|G_i| \geq e$, (2) either the first i MWMR registers written in the longest pseudo-solo run segments of processes in G_i, after $\alpha \cdot \beta_i$, are covered in $\alpha \cdot \beta_i$, or, the longest pseudo-solo run segments starting at $\alpha \cdot \beta_i$ have less than i writes to MWMR registers and all these writes are covered in $\alpha \cdot \beta_i$ and (3) the state at the end of run $\alpha \cdot \beta_i$ is transparent with respect to G_i. And in addition, there is a large set of runs* equivalent *to $\alpha \cdot \beta_i$ with respect to G_i.*

Sketch of proof of Lemma 3.7: A set of runs equivalent to β_{j+1} is constructed s.t., each run is equivalent with respect to a sub-set of the set of processes, G_{j+1}. Unless stated otherwise we use the same notations and symbols as in phases one and two. We do not provide here an evaluation of the size of $N(e, j+1)$. We only go through the modifications necessary to the steps of phase one.

Equivalent set of runs:

As explained above, in step 6, since assumption A is removed the following two cases should be addressed:

1. After run $\alpha \cdot \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1) \cdot clean(j)$, if any of the processes which participates in $clean(j)$ (i.e., $q_1, ..., q_j$), reads from the SWMR register $R_{SW_{p_1}}$, which p_1 wrote, then the state is not transparent with respect to $G_{j,1}$. However, since W is an adaptive algorithm, the number of steps in $clean(j)$ is a constant, given that k and d are constants. Therefore, the total number of SWMR registers that may be read by any $q_1, ..., q_j$ during any $clean(j)$ run segment is bounded by some constant, say l. Hence, because of the one-to-one correspondence between processes and SWMR registers, at most l active processes may be discovered during $clean(j)$ run segment. Removing these l processes from $G_{j,1}$ leaves us with a large enough set $G_{invis} \subseteq G_{j,1}$ (*invis* stands for invisible).

   Unfortunately process p_1 is not the only process that may be found active by $q_1, ..., q_j$ during $clean(j)$. In fact, since we repeat the inductive construction up to d times in step 9 of phase one, $clean(j)$ may be a part of the d'th repetition. In such a case, in addition to p_1 all the other $d-1$ processes that have been chosen to participate in the $j+1$'st cover are alive and may be discovered by $q_1, ..., q_j$ during $clean(j)$, as well.

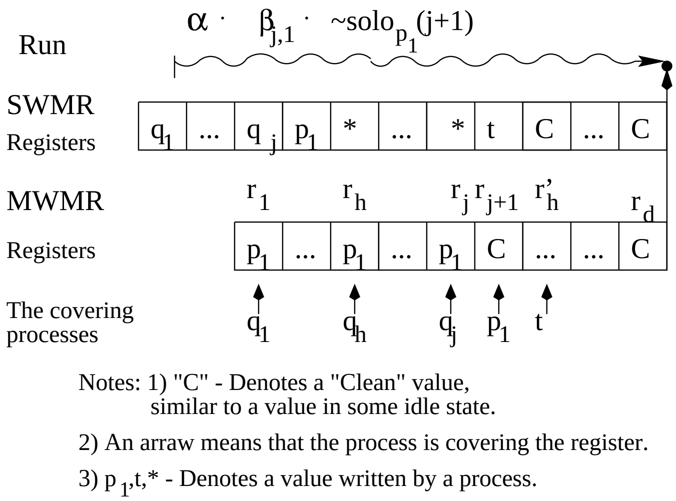

Figure 3: Illustration of the memory snapshot, after run $\alpha \cdot \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1)$ during step 6 in proof of Lemma 3.6.

Consider the following example of the above situation: (For an illustration of the corresponding memory snapshot see Figure 3).

Suppose that processes $q_1, ..., q_j$ read R_{SW_t} the SWMR register of an active process t. If in some induction step $h < j+1$, t was selected to cover the MWMR register r'_h, where $r'_h \neq r_h$, then obviously the induction step was repeated at least one more time in order to cover r_h by q_h. Therefore, both t and q_h are active at the end of $\alpha \cdot \beta_{j,1}$.

To overcome this problem, we need to construct a run $\alpha \cdot \beta^*_{j,1}$ equivalent to $\alpha \cdot \beta_{j,1}$ with respect to some set of processes, in which a process like t is "replaced" by another process t' s.t., none of processes $q_1, ..., q_j$ reads the SWMR register of t'.

Intuitively, constructing a new run equivalent to run $\alpha \cdot \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1)$, is done by repeating the construction of the current run from the beginning i.e., $i = 0, ..., j$. In the new construction, we remove active processes that are later "discovered" by $q_1, ..., q_j$. In their place we select other processes and their pseudo-solo run segment prefixes, from G_{invis} i.e., processes that their SWMR registers are not read by $q_1, ..., q_j$. Notice that we are replacing both p_1 and t in this construction s.t., p_1 is replaced by some process p'_1 and t is replaced some where in $\beta^*_{j,1}$ with t'. In the new run $\alpha \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$, that we have just generated, none of the processes $q_1, ..., q_j$ reads any SWMR register that belongs to active (not thinking) processes. Notice that $q_1, ..., q_j$ are allowed to read each other SWMR register, but still the state at the end of the run is transparent.

In conclusion, during run $\alpha \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$, processes $q_1, ..., q_j$ do not read any SWMR register of active processes by the end of this run. Moreover, processes

$q_1, ..., q_j$ do not read the SWMR registers of the processes in G_{invis}. Therefore, by the end of run $\alpha \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$, there is no "trace" of the active processes in any of the MWMR registers, and the traces in the SWMR registers are ignored. The set $G'_{invis} = G_{invis} - \{\textit{the replaced processes (like } t')\}$ is the set of processes to select from in the extension of the run or when replacing processes in future steps of the construction.

2. The state after some $clean(j)$ run segment in the run constructed in phase one, may not be transparent with respect to processes that start their execution after $clean(j)$. Even though, processes $q_1, ..., q_j$ (which participate in $clean(j)$), do not read any active process in the run during $clean(j)$, the state may still not be transparent to processes afterwards. Let a process p_2 be a process in G'_{invis}, where G'_{invis} is as defined in case 1 above. After $clean(j)$ run segment (in run $\alpha \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$), p_2 starts its $p2$-segment. In this case, p_2 may read SWMR registers belong to some active processes in the run. Thus, p_2 is not running a pseudo-solo run segment after $clean(j)$.

   Before constructing a run equivalent to the run of phase one, in which the "problematic" processes are not participating, we show that there are such equivalent runs. As shown in case 1 above, we construct an equivalent run by selecting processes from the currently available set of processes i.e., G'_{invis}. But, what if each process we select from G'_{invis} to participate in an equivalent run, is "discovered"? (i.e., its SWMR register is read). Therefore, we find a sub-set of G'_{invis} which includes only processes that do not read other's SWMR registers in the sub-set, when starting their execution after $\alpha \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$. We construct a sub-set $G^*_{j,1}$ of the set of processes G'_{invis}, as follows: First, the processes after any transparent state perform at most k operations, when running a pseudo-solo run segment. Second, as previously shown, the length of run segment $\beta^*_{j,1}$ (as run $\beta_{j,1}$) is bounded by a function of k and d operations and it also bounds the number of processes that are writing in SWMR registers in $\beta^*_{j,1}$. We remove processes from G'_{invis} that other processes read their SWMR register s.t., there is a set of processes named $G^*_{j,1}$ of at least $\frac{|G'_{invis}|}{f(k)}$, where $f(k) = (2k+1) * (k+1)$ and $f(k)$ is constructed and calculated in Claim 4.1 in Section 4.3.

   In this induction step (step $j+1$), we can generate an equivalent run to $\alpha \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$, by selecting processes from $G^*_{j,1}$. Note that since $p'_1 \in G_{invis}$, we may need to replace it again with another process, say $p''_1 \in G^*_{j,1}$. Further notice that if a problematic process participates in α, then we have to replace run α with another run α^* in which this process does not participate. It is possible, since one of the lemma's conditions says that there is a set of runs equivalent to α with respect to

G and since $G^*_{j,1} \subseteq G_{j,1} \subseteq G$.

In step 7 of the first phase, we use the set of processes $G^*_{j,1} \subseteq G_{j,1}$ instead of the set $G_{j,1} - \{p_1\}$, and the run $\alpha^* \cdot \beta^*_{j,1} \cdot^{\sim} solo_{p'_1}(j+1) \cdot clean(j)$ instead of run $s \cdot \beta_{j,1} \cdot^{\sim} solo_{p_1}(j+1) \cdot clean(j)$ as written in phase one.

■

4 Theorem Proof

In this section we proof Theorem 3.1. The Theorem was introduced in Section 3.1 and here we state it once again.

Theorem 3.1 *For any constants d and k, there is a large enough number $n_{d,k}$ s.t., there is no long-lived k-solo-adaptive WT&S implementation in a system with $n_{d,k}$ or more processes, and with d or less Multi-Writer Multi-Reader registers.*

4.1 Proof of Theorem 3.1

The proof is by a way of contradiction. We first assume that there is a WT&S implementation with d MWMR registers for a system with more than $n_{d,k}$ processes. Then, by Lemma 4.6, under these conditions there is a run in which two processes, p and q ($p \neq q$) are in the critical section (i.e., eating), at the same time. A contradiction.

■

Lemma 4.6 is called the violation lemma and in its proof we use Lemma 4.5, which is called the main lemma. Both lemmas and their proofs are below, in Sections 4.4 and 4.5. First in Section 4.2 we provide several definitions, and then in Section 4.3 we provide the required Lemmas and Claims, which are used in the proofs of Lemmas 4.5 and 4.6.

4.2 Definitions

Definition 4.1 [Multi-unique]. *We say that a set of processes Q satisfies the* Multi-unique *property at a state s, if s is transparent with respect to Q and for each MWMR register r, if r is the j'th MWMR register that is written in the longest pseudo-solo run segment of a process $p \in Q$, that starts at s, (for some j, $1 \leq j \leq k$), then r is the j'th MWMR register that is written in all longest pseudo-solo run segments of all processes in Q, when starting at s.*

Definition 4.2 [Transition]. *Let p and q be processes and α_p, α_q runs s.t., p is active at the end of α_p and q is active at the end of α_q. Suppose that $\alpha_p = y \cdot x_p \cdot z$, where x_p is a p-segment, and p does not participate neither in y nor in z. Suppose also that $\alpha_q = y' \cdot x_q \cdot z'$, where x_q is a q-segment, and q does not participate neither in y' nor in z'. If $p \neq q$, $y = y'$ and $z = z'$, then we say that there is a* transition *from α_p to α_q* using process q *and denote it by $\alpha_p[^{p}\!\longrightarrow^{q}]\alpha_q$.*

Definition 4.3 [Multiple Transitions]. *We say that there are f transitions from a run α to a run β using processes $q_1, ..., q_f$, for some $f > 0$, if there are f runs $\alpha_1, ..., \alpha_f$, f processes $p_1, ..., p_f$ and another f processes $q_1, ..., q_f$ s.t., $\forall i, j$ $p_i \neq q_j$, $\alpha[^{p_1} \longrightarrow ^{q_1}]\alpha_1$, $\alpha_1[^{p_2} \longrightarrow ^{q_2}]\alpha_2$, ..., $\alpha_{f-1}[^{p_f} \longrightarrow ^{q_f}]\alpha_f$ and $\alpha_f = \beta$. We denote it simply by $\alpha \longrightarrow \beta$.*

Definition 4.4 [Unreadable]. *We say that a state s is* unreadable *with respect to a set of processes Q, if s is transparent with respect to Q and for each process $p \in Q$ all its pseudo-solo run segments that start at s, do not read any SWMR register belongs to other processes in Q.*

In the proof of our main lemma (Lemma 4.5), we construct a pattern of runs instead of one run. This pattern enables us to construct from the original run many alternative runs, which contain the same write operations in MWMR registers, but different write operations in SWMR registers. This method is required in our proof of the main Lemma, since the run constructed may have a write operation in a SWMR register that other processes read. In our proof, we construct a pattern of runs by first constructing a run, which satisfies the *alternate-run property*, as defined in Definition 4.5 below. Then in Lemma 4.4 we show how to construct new runs from a given run that satisfies the *alternate-run property*.

Definition 4.5 ***[Alternate-Run].*** *A run α satisfies the* alternate-run *property with respect to a set of processes Q, if we can divide α to run segments s.t., $\alpha = alone_1 \cdot clean_1 \cdots alone_c \cdot clean_c$, for some constant number $c \geq 0$ and $\forall i$ $(1 \leq i \leq c)$ the following properties are satisfied:*

$P1$: *The run segment $alone_i$ is a p_i-segment of some process p_i. At the beginning of $alone_i$, process p_i starts its WT&S execution, possibly write in some MWMR registers, and by the end of $alone_i$ p_i either covers a MWMR register or eats. In addition, all MWMR registers $r_1, ..., r_j$ written by p_i, for some $j \geq 0$, are covered at $alone_1 \cdot clean_1 \cdots alone_{i-1} \cdot clean_{i-1}$ by processes $q_1, ..., q_j$.*

$P2$: *In run segment $clean_i$, only processes $q_1, ..., q_j$ participate. Each process q_l first writes in r_l, $1 \leq l \leq j$, and when all the registers are written, $q_1, ..., q_j$ continue their execution until they reach the* thinking *state. (When $j = 0$, $clean_i$ is the empty run segment.)*

$P3$: *$p_i \notin Q$.*

$P4$: *In the beginning of $alone_i$, all processes in $Q \cup \{p_i\}$ satisfy the Multi-unique property.*

$P5$: *Process p_i does not participate in $alone_l$, $\forall l \neq i$.*

P6: *The state at the beginning of $alone_i$, is unreadable with respect to $Q \cup \{p_i\}$.*

P7: *During $clean_i$, the only SWMR registers processes $q_1, ..., q_j$ read are either their own or the SWMR registers belongs to processes which do not participate in α. In addition, processes $q_1, ..., q_j$ do not read any SWMR registers belongs to processes in Q.*

P8: *At the end of run α both Q satisfies the Multi-unique property and the state is unreadable with respect to Q.*

4.3 Basic Lemmas and proofs

In this subsection, we provide the Lemmas and Claims necessary for the proofs of the main lemma, (Lemma 4.5) and the violation lemma, (Lemma 4.6).

The following two Claims 4.1 and 4.2 say that given a transparent state with respect to a large enough set of processes G, then there is an additional property that is satisfied with respect to a large enough sub-set of G. The additional properties are *unreadable* and *Multi-unique*.

Claim 4.1 *Let W be a long-lived k-solo-adaptive WT&S algorithm with at most d MWMR registers. Then for every constant e, a state s and a set of processes G, if $|G| \geq e * (2k+1) * (k+1)$ and s is transparent with respect to G, then there is a set of processes G_u, $G_u \subseteq G$ s.t., s is unreadable with respect to G_u and $|G_u| \geq e$.*

Proof of Claim 4.1.

Let G_u be a set of processes. By the *unreadable* definition, a state s is unreadable with respect to G_u if:

1. s is transparent with respect to G_u and
2. each process $p \in G_u$ that starts any of its pseudo-solo run segments at s, does not read any SWMR register of some other process in G_u.

In this proof, we will do the following:

- Construct a set of processes $G_u \subseteq G$ s.t., condition 1 and 2 are satisfied.
- We will show that if $|G| \geq e * (2k+1) * (k+1)$, then $|G_u| \geq e$.

We first select a sub-set of processes G' from G, as follows: Since W is k-solo-adaptive, then each process can read at most k registers in its longest pseudo-solo run segment. Therefore, in all the longest pseudo-solo run segments of processes in G starting after a transparent state s, there are at most $k * |G|$ read operations. In the worst case, all the read operations are only from SWMR registers. In this case, there are at most $\frac{k*|G|}{k+1}$ SWMR registers that are read $k+1$ times or more in all the longest pseudo-solo run segments of processes in G, starting at state s. Thus, at least $|G| - \frac{k*|G|}{k+1} = \frac{|G|}{k+1}$ SWMR registers are read at most k times in all the longest pseudo-solo run segments of processes in G, starting at state s. We mark these SWMR registers by R'_{SW}, $|R'_{SW}| \geq \frac{|G|}{k+1}$. Let G' be the set of processes whose SWMR are in R'_{SW}. Clearly, $|G'| \geq \frac{|G|}{k+1}$.

We tag the processes in G', as follows:

1. At the beginning, all processes in G' are untagged.

2. Select any untagged process $q \in G'$ and mark it with $SELECT$. Let r_q be q's corresponding SWMR register, $r_q \in R'_{SW}$.

3. We tag with $REMOVE$ all the processes in G' that read r_q in their longest pseudo-solo run segment starting at s. To guarantee that later on, q itself will not be tagged with $REMOVE$, then we tag some more processes in G' with $REMOVE$, which q reads from their corresponding SWMR registers, in the longest pseudo-solo run segment of q starting at s. By our construction of G', there are at most k processes in G' that read r_q in their pseudo-solo run segment starting at s. In the longest pseudo-solo run segment, q may read at most k registers, since W is a k-solo-adaptive algorithm. Therefore, we tagged at most $2k$ processes with $REMOVE$.

4. Repeat from step 2 until all processes are tagged.

The set G_u consist of all the processes marked by $SELECT$ and $|G_u| \geq \frac{|G'|}{2k+1} \geq \frac{|G|}{(2k+1)*(k+1)}$. Therefore, if $|G| \geq e * (2k+1) * (k+1)$, then $|G_u| \geq e$.

The first condition which says that state s is transparent with respect to G_u is satisfied immediately, since s is transparent with respect to G and $G_u \subseteq G$.

The second condition is proved for the longest pseudo-solo run segments of processes in G_u. But all the pseudo-solo run segments of a process at a given state are the prefixes of the longest pseudo-solo run segment. Therefore, if a process does not read the SWMR registers of other processes in G_u in its longest pseudo-solo run segment, it cannot read these registers in any prefix of this run segment.

■

Claim 4.2 *Let W be a long-lived k-solo-adaptive WT&S algorithm with at most d MWMR registers. Then for every constant e, a state s and a set of processes G, if $|G| \geq e * [2d]^k$ and s is transparent with respect to G, then there is a set of processes G_M, $G_M \subseteq G$ s.t., G_M satisfies the Multi-unique property at s and $|G_M| \geq e$.*

Proof:

The proof is by induction on the number of write operations in MWMR registers during the longest pseudo-solo run segments starting at s.

Basis: $i = 0$. When all the write operations are only in SWMR registers and there is no write operation in MWMR registers, then the Multi-unique property is satisfied at s for $G_0 = G$.

Hypothesis. Suppose that the lemma is true for some $0 \leq i < k$ and $G_i \subseteq G$.

Induction step. We prove the lemma for $i + 1$.

In induction step $i + 1$, we observe the $i + 1$'st write operation in MWMR registers of all the longest pseudo-solo run segments of processes in G_i, starting at s. Then, we construct the new set G_{i+1}, as follows:

1. If more than half of the processes do not have a pseudo-solo run segment with $i + 1$ write operations in MWMR registers before eating, then we define G_{i+1} to be these processes i.e., $|G_{i+1}| \geq \frac{1}{2}|G_i|$. Thus, $G_M = G_{i+1}$ is the set of processes that holds the Multi-unique property at s.

2. Else, we denote the rest of the processes by G'_{i+1} and $|G'_{i+1}| \geq \frac{1}{2}|G_i|$. Hence, processes in G'_{i+1} have an $i + 1$'st write operation in some MWMR registers. Therefore, there is at least one MWMR register that a set of size $\frac{|G'_{i+1}|}{d}$ is writing in it. We denote this set by G_{i+1}. The recursive function is $|G_{i+1}| \geq \frac{|G'_{i+1}|}{d} \geq \frac{|G_i|}{2d}$.

Since there are at most k rounds (steps), then $|G_k| \geq \frac{|G_0|}{[2d]^k}$. Finally, we denote $G_M = G_k$ and since $G_0 = G$, then $|G_M| \geq \frac{|G|}{[2d]^k}$. Another way to present this equation is by setting $|G_M| = e$, then $|G| \geq e * [2d]^k$ guarantees that $|G_M| \geq e$.

■

If there is a run α satisfying the alternate-run property with respect to a set of processes G, then each process in G can start its pseudo-solo run segment instead of any *alone* run segment in α. This is immediately from property $P6$ of the alternate-run. But, we have to show that the new run is a valid run and that we can repeat and replace processes again and again. To show that there is one transition, we state Lemma 4.3, which says that given a run α that satisfies the alternate-run property with respect to a set of processes G, then there is one-transition to a new run β, which also satisfies the alternate-run property with respect to a sub-set of processes. Since run β satisfies the alternate-run property, it allows us to repeat the transition-process again. Therefore, Lemma 4.3 is used to prove the next lemma (Lemma 4.4), which says that there are Multi-transitions.

Lemma 4.3 [**One Transition**] *Let W be a long-lived k-solo-adaptive WT&S algorithm with at most d MWMR registers. Then for any run α satisfying the alternate-run property with respect to a set of processes G, if there is an active process p in α and $|G| > 1$, then for any process q in G, there is a run β s.t., $\alpha[^{p}\longrightarrow^{q}]\beta$. Additionally, there is a set of processes G_β, $G_\beta = G - \{q\}$ s.t., run β satisfies the alternate-run property with respect to G_β.*

Proof:

We show that we can transit any active process p in α with another process in G. Let $alone_i$ be the run segment that p participates in α and s_i be the state at the beginning of $alone_i$ in α.

1. Since state s_i satisfies the Multi-unique property (condition $P4$ of the alternate-run property) with respect to $G \cup \{p\}$, then there is a process q and q-segment s.t., q writes in the same MWMR registers and covers the same MWMR register as p, (since $q \in G \cup \{p\}$).

2. Let β be a run s.t., it contains all events as α, but $alone_i$. Instead of $alone_i$ events, we add the q-segment events at s_i and denote it by $alone'_i$. By step 1, all the MWMR registers written by q are overwritten in run segment $clean_i$ in run β, as in run α.

3. By condition $P6$ of the alternate property satisfied by α, state s_i is transparent with respect to G and $q \in G$. Therefore, the q-segment ($alone'_i$) which starts at s_i is also a pseudo-solo run segment of q.

4. By condition $P7$ of the alternate run property, none of the participating processes in any *clean* run segment of run α read the SWMR registers which belong to processes in G and of $q \in G$. Processes in any *clean* run segment in run β make the same steps as in run α, since they read the same values from the SWMR and MWMR registers.

5. The state at the beginning of each *alone* run segment $alone_j$ that follow $alone_i$ (for $j \geq i+1$), is unreadable with respect to G and $q \in G$. This is directly by condition $P6$ of the alternate-run property, which is satisfied by α. Thus, the participating processes in $alone_j$ cannot read the SWMR register of q and perform the same steps in both runs α and β.

6. At the end of run β, each process in G cannot read the SWMR register of q, since the state at the end of run α is unreadable with respect to G (by property P8 of the alternate-run) and since $q \in G$. Therefore, processes in $G - \{q\}$ cannot distinguish between runs α and β.

Let $G_\beta = G - \{q\}$. As explained above, run β has the same events as α, but the events in $alone_i$ and with q participating instead of p. Run β satisfies the alternate-run property with respect to G_β, since all properties are satisfied for each j ($j > 0$), as follows:

For $j < i$: The prefix of run α is identical to the prefix of run β, by our construction.

For $j = i$:

$P1$: By our construction in steps 1 - 3, run segment $alone'_i$ is a pseudo-solo run segment in which q writes in the same j MWMR registers as p and covers the $j+1$ MWMR register. Therefore, all the write operations in MWMR registers of q are covered in the following $clean_i$ run segment.

$P2$: Run segment $clean_i$ is the same in runs α and β, as explained in step 4 above.

$P3$: The participating process in $alone'_i$ is q and $q \notin G_\beta$, since $G_\beta = G - \{q\}$.

$P4$: In the beginning of $alone'_i$, the Multi-unique property holds with respect to $G_\beta \cup \{q\}$, since run α and β are identical up to this point and in run α the Multi-unique property holds with respect to $G \cup \{p\}$ and $G = G_\beta \cup \{q\}$.

$P5$: Process q does not participate in $alone_l$, $\forall l \neq i$, since $q \in G$ and none of the processes in G participates in α.

$P6$: The state in the beginning of $alone'_i$ is still unreadable with respect to $G \cup \{p\}$, since the prefix of the run till $alone'_i$ is the same as in α. Therefore, the state in the beginning of $alone'_i$ is also unreadable with respect to $G_\beta \cup \{q\}$, since $G_\beta \cup \{q\} = G$.

$P7$: By step 4 above, processes in *clean* run segments in run β perform the same steps as in run α and they do not read the SWMR register of q, since $q \in G$.

$P8$: See below.

For $j > i$: Suppose that a process p_j participates in $alone_j$. All properties are satisfied as follows:

$P1$: The run segment $alone_j$ is the same in run α and β, as explained above in step 5.

$P2$: Run segment $clean_j$ is the same in runs α and β, as explained in step 4 above.

$P3$: $p_j \notin G_\beta$, since in run α $p_j \notin G$ and $G_\beta = G - \{q\}$.

$P4$: In the beginning of $alone_j$, the Multi-unique property holds with respect to $G_\beta \cup \{p_j\}$, since: (1) none of the processes in $G_\beta \cup \{p_j\}$ can distinguish between α and β at the begining of $alone_j$, as explained in step 5 above, (2) in run α the Multi-unique property holds with respect to $G \cup \{p_j\}$. Since $G = G_\beta \cup \{q\}$, then it holds with respect to $G_\beta \cup \{p_j\}$.

$P5$: In run α process p_j does not participate in $alone_l$, $\forall l \neq j$. Also, we do not select process p_j to participate in $alone_i$ in run β, since $p_j \notin G$ and we transit only to processes from G.

$P6$: The state in the beginning of $alone_j$ is unreadable with respect to $G_\beta \cup \{q\}$, since (1) none of the processes in $G_\beta \cup \{p_j\}$ can distinguish between α and β, as explained in step 5 above, (2) in run α the state in the beginning of $alone_j$ is unreadable with respect to $G \cup \{p_j\}$. Since $G = G_\beta \cup \{q\}$, then it holds with respect to $G_\beta \cup \{p_j\}$.

$P7$: By step 4 above, processes in *clean* run segments in run β perform the same steps as in run α and they do not read the SWMR register of q, since $q \in G$.

$P8$: See below.

Finally, property $P8$ is satisfied as follows: the state at the end of run β is transparent and unreadable with respect to G_β and processes in G_β maintain the Multi-unique property by the end of β, since processes in G_β cannot distinguish between α and β, as explained in step 6 above. Therefore, run β satisfies the alternate-run property with respect to G_β. ■

Lemma 4.4 [**Multiple Transitions**] *Let W be a long-lived k-solo-adaptive WT&S algorithm with at most d MWMR registers. Then for any run α satisfying the alternate-run property with respect to a set of processes G and for any $x \geq 0$, if there are x active processes in α and $|G| > x$, then for any set G_x of x processes, where $G_x \subseteq G$, there is a run β s.t., there are x transitions from α to run β using processes in G_x. Additionally, there is a set of processes G_β, $G_\beta = G - G_x$ s.t., run β satisfies the alternate-run property with respect to G_β.*

Proof:

The proof is by induction on the number of transitions. Let $t \geq 0$ be the number of active processes in α. Note that if $t = 0$, then there are no transitions and run $\beta = \alpha$ satisfies all the lemma terms.

Basis: $x = 1$. By Lemma 4.3, for any active process p in α and any process $q \in G$, there is a run β s.t., $(\alpha[^{p}\!\longrightarrow^{q}]\beta)$ and $G_\beta = G - \{q\}$.

Hypothesis. Suppose the lemma is true for some $1 \leq x < t$.

Induction step. We prove the lemma for $x + 1$. By the inductive hypothesis, there is a new run α_x s.t., there are x transitions from α to α_x and α_x satisfies the alternate-run property with respect to a set of processes $G - G_x$. Since run α_x satisfies the alternate-run property, then all the required conditions of Lemma 4.3 are fulfilled. Hence, for any active process p in α_x (and in α) and for any process $q \in G - G_x$, there is a run α_{x+1} s.t., there is a transition from α_x to α_{x+1}, $\alpha_x[^{p}\!\longrightarrow^{q}]\alpha_{x+1}$. Let $G_{x+1} = G_x \cup \{q\}$. By Lemma 4.3, the new run α_{x+1} satisfies the alternate-run property with respect to $G_\beta = G - G_x - \{q\} = G - G_{x+1}$.

Notice that the participating processes in any *clean* run segment are inactive by the end of run α and therefore they remain after every transition, since we transit only active processes at the end of α. Further notice that for any two new processes in α_{x+1}, say q_l and q_j, where $j > l \geq 1$, then q_j cannot read the SWMR register written by q_l. It is simply because the state at the beginning of the pseudo-solo run of q_j is unreadable with respect to G and $q_l \in G$.

■

4.4 The main Lemma

Lemma 4.5 **[Main Lemma]** *Let W be a long-lived k-solo-adaptive WT&S algorithm with at most d MWMR registers. Then for every i, $0 \leq i \leq k$, for every constant e and for any run α satisfying the alternate-run property with respect to a set of processes G, there exists a constant $N(e,i)$ s.t., if $|G| \geq N(e,i)$, then there is a run β_i and a set of processes $G_i \subseteq G$ s.t., $\beta_i = \alpha^* \cdot y_i$, where $\alpha \longrightarrow \alpha^*$, y_i is a run segment and the following hold:*

1. *$|G_i| \geq e$,*
2. *either the first i MWMR registers written in the longest pseudo-solo run segments of processes in G_i, after β_i, are covered in β_i, or, the longest pseudo solo runs starting at β_i have less than i writes in MWMR registers before eating and all these writes are covered in β_i.*

3. run β_i satisfies the alternate-run property with respect to G_i.

High level description of the Proof of Lemma 4.5: See Section 3.3.

Proof of Lemma 4.5. Proof is by induction on i.

Basis. $i = 0$. Zero cover means no cover on MWMR registers and the run $\beta_0 = \alpha$ satisfies term 2 of the lemma. Let $G_0 = G$ and $N(e, 0) = e$, thus $|G_0| = e$ and condition 1 is fulfilled. Condition 3 of the lemma is satisfied because α satisfies the alternate-run property with respect to G.

Hypothesis. Suppose the lemma is true for $i = j$, $0 \leq j \leq k - 1$.

Induction step $i = j + 1$**.**

(Since there are several iterations to construct the set of processes G_{j+1} from G_j, then we use $Q^{(x)}_{j+1,y}$ notation for intermediate steps, where x and y are explained in the proof below.)

1. By the induction hypothesis ($i = j$), for a given number $e \geq 0$ (that is calculated by the end of this proof), there is a run β_j that satisfies all three conditions with respect to a set of processes G_j. We denote $G_{j,1} = G_j$ and $\beta_{j,1} = \beta_j$.

2. If after $\beta_{j,1}$, all the pseudo-solo run segments have at most j write operations in MWMR registers before eating, then by condition 2 of the lemma, run $\beta_{j,1}$ satisfies all the terms with respect to $G_{j,1}$ and we are done.

3. Suppose that run $\beta_{j,1} = alone_1 \cdot clean_1 \ldots alone_c \cdot clean_c$ for some constant $c \geq 0$. We construct an extension $alone_{c+1} \cdot clean_{c+1}$ to run $\beta_{j,1}$ by first constructing run segment $clean_{c+1}$, as follows: By condition 2 of the lemma, there are j processes $q_1, ..., q_j$ that cover the first j MWMR registers $r_1, ..., r_j$, written in all the pseudo-solo run segments of processes in $G_{j,1}$, when starting after $\beta_{j,1}$. We construct a run segment $clean_{c+1}$ similarly to the definition of *clean* run segment in the alternate-run property (condition $P2$). First we let each process q_i to write in r_i and then we let all the processes $q_1, ..., q_j$ to continue until thinking.

4. We select a sub-set of processes $Q^{(1)}_{j+1,1}$ from $G_{j,1}$ s.t., none of the participating processes $q_1, ..., q_j$ in $clean_{c+1}$ reads SWMR registers written by processes in $Q^{(1)}_{j+1,1}$. Since there are at most $|clean_{c+1}|$ read operations in $clean_{c+1}$, then there are at most $|clean_{c+1}|$ SWMR registers that are read. By the one-to-one correspondence between SWMR registers and the processes that are writing in them, we can remove these processes

from $G_{j,1}$ and the remained set of processes is denoted by $Q^{(1)}_{j+1,1}$. Thus $|Q^{(1)}_{j+1,1}| = |G_{j,1}| - |clean_{c+1}|$.

To show that the sub-set $Q^{(1)}_{j+1,1}$ exists, we prove that $clean_{c+1}$ has a constant length. Since W is an adaptive algorithm, then the number of operations that processes are allowed to do in $clean_{c+1}$, depends on the number of active processes in the run. By the induction hypothesis, run $\beta_{j,1}$ has at most a constant c of active processes, (because in each *alone* run segment in $\beta_{j,1}$, a new process is starting its execution). Thus, $|clean_{c+1}| = f(c)$, where $f(c)$ is a function of a constant c.

5. We construct a run segment $alone_{c+1}$ as follows: Select any process $p \in Q^{(1)}_{j+1,1}$ and a pseudo-solo run segment $^{\sim}solo_p$ s.t., $^{\sim}solo_p$ starts at the end of run $\beta_{j,1}$, and in $^{\sim}solo_p$ p writes in $r_1, ..., r_j$ and covers the $j+1$ MWMR register r_{j+1}. Let $alone_{c+1} =^{\sim} solo_p$.

   $^{\sim}solo_p$ exists, because of the following:

   (a) By the inductive hypothesis term 3, $\beta_{j,1}$ satisfies the alternate-run property with respect to $G_{j,1}$. Condition $P8$ of the alternate-run property says that the state by end of a run is transparent. Therefore, the state at the end of run $\beta_{j,1}$ is transparent with respect to $G_{j,1}$. By this, every process in $G_{j,1}$ has pseudo-solo run segments after run $\beta_{j,1}$ and specifically $p \in Q^{(1)}_{j+1,1}$, $Q^{(1)}_{j+1,1} \subseteq G_{j,1}$.

   (b) Again because $\beta_{j,1}$ satisfies the alternate-run property with respect to $G_{j,1}$, then by condition $P8$, $G_{j,1}$ satisfies the Multi-unique at the end of $\beta_{j,1}$. Thus, after run $\beta_{j,1}$ all the MWMR registers written in all the longest pseudo-solo run segments of processes in $G_{j,1}$, are the same. Since $p \in Q^{(1)}_{j+1,1}$ and $Q^{(1)}_{j+1,1} \subseteq G_{j,1}$, then there is a prefix of p's longest pseudo-solo run segment, in which p writes in the same MWMR registers that are covered i.e., $r_1, ..., r_j$.

6. The run $\beta_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ should satisfy the alternate-run property and specifically condition $P8$, which says that the state at the end of this run is transparent and unreadable. But, there are two reasons why the state at the end of $\beta_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ may not be transparent with respect to $Q^{(1)}_{j+1,1}$:

   (1) If processes in $clean_{c+1}$ read x SWMR registers written by active processes in $\beta_{j,1}$ for some x $(x \geq 1)$.

   (2) If processes in $Q^{(1)}_{j+1,1}$ execute their pseudo-solo run segments after $\beta_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ and they read a set of x SWMR registers R_{SW_x} written in the run before, by a set of x active processes G_x, for some x $(x \geq 1)$.

In the following two steps (7 and 8), we construct a run $\alpha_{j+1,1}$ s.t., $\beta_{j,1} \cdot alone_{c+1} \cdot clean_{c+1} \longrightarrow \alpha_{j+1,1}$ and we select a sub-set of processes from $Q^{(1)}_{j+1,1}$ s.t., the state at the end of $\alpha_{j+1,1}$ is transparent and unreadable with respect to the new sub-set of processes.

7. As explained in step 6, the state at the end of run $\beta_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ may not be transparent with respect to $Q^{(1)}_{j+1,1}$, if processes in $clean_{c+1}$ read x SWMR registers written by active processes in $\beta_{j,1}$ for some x ($x \geq 1$). Since run $\beta_{j,1}$ satisfies the alternate-run property with respect to $G_{j,1}$ and $Q^{(1)}_{j+1,1} \subseteq G_{j,1}$, then by Lemma 4.4 there is a run $\beta^*_{j,1}$ s.t., all the active processes in $\beta^*_{j,1}$ are only from $Q^{(1)}_{j+1,1}$. The set of processes $Q^{(1)}_{j+1,1}$ was selected in step 4 s.t., none of the participating processes $q_1, ..., q_j$ in $clean_{c+1}$ reads SWMR registers written by processes in $Q^{(1)}_{j+1,1}$. Lemma 4.4 says that $\beta_{j,1} \longrightarrow \beta^*_{j,1}$, $\beta^*_{j,1}$ satisfies the alternate-run property with respect to a set of processes $Q^{(2)}_{j+1,1}$ and $Q^{(2)}_{j+1,1} \subset Q^{(1)}_{j+1,1}$. In Lemma 4.4, the size of $Q^{(2)}_{j+1,1}$ is calculated: $|Q^{(2)}_{j+1,1}| = |Q^{(1)}_{j+1,1}| - x$. Since there are at most c active processes in $\beta_{j,1}$, then $x \leq c$ and $|Q^{(2)}_{j+1,1}| \geq |Q^{(1)}_{j+1,1}| - c$.

   Notice that processes in $clean_{c+1}$ cannot read the SWMR register that p writes during $alone_{c+1}$, since $p \in Q^{(1)}_{j+1,1}$.

8. As explained in step 6, the state after run $\beta^*_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ may not be transparent with respect to $Q^{(1)}_{j+1,1}$ nor $Q^{(2)}_{j+1,1} \subseteq Q^{(1)}_{j+1,1}$, when processes in $Q^{(1)}_{j+1,1}$ or $Q^{(2)}_{j+1,1}$ execute their pseudo-solo run segment after this run. These processes may read a set of x SWMR registers R_{SW_x} written in the run before, by a set of x active processes G_x, for some x ($x \geq 1$).

   Only for a moment, suppose that we remove the processes in G_x from the run, by multiple transitions to a new run using x processes from $Q^{(2)}_{j+1,1}$. It seems like this operation should solve the problem. But, the state at the end of the new run has been changed and there may be other x' processes that their SWMR registers are read. We can repeat the same operation again till either this loop terminates, or it goes to an infinite loop. To avoid an infinite loop, we carefully select a set of processes from $Q^{(2)}_{j+1,1}$ s.t., it will allow us to do the multiple transitions only once and to have a new run that is unreadable.

   The solution is divided into three steps: In step 8a, run $temp_{j+1,1}$ is constructed only to find the set of processes $Q^{(3)}_{j+1,1} \subseteq Q^{(2)}_{j+1,1}$ s.t., $temp_{j+1,1}$ is transparent and unreadable with respect to $Q^{(3)}_{j+1,1}$. In step 8b, we construct a run $\alpha_{j+1,1}$ by multiple-transitions from run $\beta^*_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ s.t., all the active processes are from $Q^{(3)}_{j+1,1}$. Finally, in step 8c, we prove that there is a large enough set of processes that cannot distinguish between the runs $\alpha_{j+1,1}$ and $temp_{j+1,1}$. Thus, the state at the end of run $\alpha_{j+1,1}$ is unreadable.

(a) Only for a while, we remove all operations done by processes in G_x by removing x corresponding *alone* run segments from run $\beta^*_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ and constructing a run $temp_{j+1,1}$. (For example, if the original run is presented by: $original = alone_1 \cdot clean_1 \cdot alone_2 \cdot clean_2 \ldots alone_{c+1} \cdot clean_{c+1}$, then by removing $alone_2$ the new run is: $temp = alone_1 \cdot clean_1 \cdot clean_2 \ldots alone_{c+1} \cdot clean_{c+1}$. Suppose that process p_2 participates in $alone_2$ and p_2 is active by the end of the *original* run. By the alternate run definition, process p_2 participates only in $alone_2$ during *original* run. Other processes in *original* run, do not read the SWMR written by p_2 and cannot differentiate a run in which p_2 participates or not.)

Run $temp_{j+1,1}$ is a valid run, because of the following three reasons:

(1) When removing a process from the run, we removed all its operations in the run. This follows immediately our construction, in which we removed only active processes and their *alone* run segments. By the alternate run definition, an active process can participate only in one *alone* run segment, and must not participate in any *clean* run segment. By this, removing the *alone* run segment from the original run, means removing all steps of the process in the run. Thus, we removed all traces of processes in G_x.

(2) None of the remained processes from run $\beta^*_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ read any of the registers in R_{SW_x}, since the following: (I) All participating processes in *clean* run segments do not read the SWMR registers from R_{SW_x} in run $\beta^*_{j,1}$, since $\beta^*_{j,1}$ satisfies the alternate-run property and by condition $P7$ processes in *clean* run segments cannot read the SWMR registers belong to other active processes in the run, (unless they participate in the same *clean* run segment). (II) The participating processes in the remained *alone* run segments in run $\beta^*_{j,1}$ start at transparent states (by condition $P6$). Therefore, they do not read the registers of active processes in the run before, specifically R_{SW_x}. (III) Also, in $alone_{c+1} \cdot clean_{c+1}$, all participating processes cannot read the SWMR registers written by the removed processes. It is because, process p starts its execution in $alone_{c+1}$ in a transparent state (see step 5), and processes in $clean_{c+1}$ cannot read any write operation of active processes in the run, as explained in step 7.

(3) The MWMR registers written by the *removed* processes are immediately over-

written in the following *clean* run segments. Therefore, processes that participate after that, can not read the write operations in MWMR registers performed by the *removed* processes.

Therefore, processes in run $temp_{j+1,1}$ continue to perform the same steps, they did in run $\beta^*_{j,1} \cdot alone_{c+1} \cdot clean_{c+1}$ and the run is valid.

Run $temp_{j+1,1}$ enables us to find a set of processes s.t., the state at the end of $temp_{j+1,1}$ is unreadable with respect to these processes, as follows: After run $temp_{j+1,1}$, the state is transparent with respect to $Q^{(2)}_{j+1,1}$, since we removed all the write operations in SWMR registers that are read later. By Claim 4.1, given a transparent state with respect to a set of processes, then there is a set of processes $Q^{(3)}_{j+1,1}$, where $Q^{(3)}_{j+1,1} \subseteq Q^{(2)}_{j+1,1}$ s.t., the state after run $temp_{j+1,1}$ is unreadable with respect to $Q^{(3)}_{j+1,1}$. In Claim 4.1, the size of $Q^{(3)}_{j+1,1}$ is calculated: $|Q^{(3)}_{j+1,1}| \geq \frac{|Q^{(2)}_{j+1,1}|}{(2k+1)*(k+1)}$.

(b) Constructing run $\alpha_{j+1,1}$, as follows: In step 7, we constructed a run $\beta^*_{j,1}$, which satisfies the alternate-run property with respect to $Q^{(2)}_{j+1,1}$. We use Lemma 4.4 once again on $\beta^*_{j,1}$ and replace processes in G_x with other processes from $Q^{(3)}_{j+1,1}$. By Lemma 4.4, there is a run $\beta^{**}_{j,1}$ and a set of processes $Q^{(4)}_{j+1,1}$, $Q^{(4)}_{j+1,1} \subseteq Q^{(3)}_{j+1,1}$ s.t., $\beta^*_{j,1} \longrightarrow \beta^{**}_{j,1}$, $\beta^{**}_{j,1}$ satisfies the alternate-run property with respect to $Q^{(4)}_{j+1,1}$ and $|Q^{(4)}_{j+1,1}| = |Q^{(3)}_{j+1,1}| - x$. Since there are at most c active processes in $\beta^*_{j,1}$ (as $\beta_{j,1}$), then $x \leq c$ and $|Q^{(4)}_{j+1,1}| \geq |Q^{(3)}_{j+1,1}| - c$.

If process $p \in G_x$ and p participates in $alone_{c+1}$, then we select another process $p' \in Q^{(4)}_{j+1,1}$ and its pseudo-solo run segment $^{\sim}solo_{p'}$ instead of $alone_{c+1}$ s.t., p' writes in the same MWMR registers and p' covers the same MWMR register as p. We denote $Q^{(5)}_{j+1,1} = Q^{(4)}_{j+1,1} - \{p'\}$ and $alone'_{c+1} =^{\sim} solo_{p'}$. We denote $\alpha_{j+1,1} = \beta^{**}_{j,1} \cdot alone'_{c+1} \cdot clean_{c+1}$.

(c) Processes in $Q^{(5)}_{j+1,1}$ cannot distinguish between runs $\alpha_{j+1,1}$ and $temp_{j+1,1}$, when executing their pseudo-solo run segments after these runs, as follows: Suppose that a process $q \in Q^{(5)}_{j+1,1}$ starts its longest pseudo-solo run segment after run $temp_{j+1,1}$. As showed in 8a, q cannot read any of the SWMR registers written by processes in $Q^{(3)}_{j+1,1}$. After run $\alpha_{j+1,1}$, the value of the registers is as after run $temp_{j+1,1}$, but with some SWMR registers written by processes in $Q^{(3)}_{j+1,1}$. Since q does not read any SWMR register written by processes in $Q^{(3)}_{j+1,1}$, then q cannot distinguish between the two runs. Hence, after run $\alpha_{j+1,1}$ the state is unreadable (and transparent) with respect to $Q^{(5)}_{j+1,1} \subseteq Q^{(3)}_{j+1,1}$.

9. At the end of run $\alpha_{j+1,1}$, there is a set of processes that satisfies the Multi-unique property, as follows: In Claim 4.2, given a transparent state with respect to a set of processes, we calculate a sub-set of processes s.t., the sub-set of processes satisfies the Multi-unique property. Since the state at the end of $\alpha_{j+1,1}$ is transparent with respect to $Q^{(5)}_{j+1,1}$, then there is a set $Q^{(6)}_{j+1,1}$ s.t., $Q^{(6)}_{j+1,1} \subseteq Q^{(5)}_{j+1,1}$, $|Q^{(6)}_{j+1,1}| \geq \frac{|Q^{(5)}_{j+1,1}|}{[2d]^k}$ and $Q^{(6)}_{j+1,1}$ satisfies the Multi-unique at the end of $\alpha_{j+1,1}$.

10. Run $\alpha_{j+1,1}$ satisfies the alternate-run property with respect to $Q^{(6)}_{j+1,1}$, as follows:

    Since run $\alpha_{j+1,1} = \beta^{**}_{j,1} \cdot alone'_{c+1} \cdot clean_{c+1}$, then we first prove that the prefix $\beta^{**}_{j,1}$ satisfies the alternate-run property and then the suffix $alone'_{c+1} \cdot clean_{c+1}$. Since in step 8b we construct run $\beta^{**}_{j,1}$ s.t., it satisfies the alternate-run property with respect to $Q^{(4)}_{j+1,1}$, then it is true for $Q^{(6)}_{j+1,1} \subseteq Q^{(4)}_{j+1,1}$.

    The extension $alone'_{c+1} \cdot clean_{c+1}$ maintains all the properties of the alternate-run with respect to $Q^{(6)}_{j+1,1}$, as follows:

    P1: In step 5, we constructed run segment $alone_{c+1}$ to be a pseudo-solo run segment of process p s.t., p writes in $r_1, ..., r_j$ and covers r_{j+1}. In step 8b, run segment $alone'_{c+1}$ was constructed as $alone_{c+1}$. Therefore, $alone'_{c+1}$ is a pseudo-solo run segment of process p' s.t., p' writes in $r_1, ..., r_j$ and covers r_{j+1}.

    P2: In step 3, we constructed run segment $clean_{c+1}$ by the alternate-run definition. Then, in the following steps, $clean_{c+1}$ remained the same, since we transit only active processes.

    P3: Process p' that participates in $alone'_{c+1}$ was selected in step 8b above s.t., $p' \notin Q^{(5)}_{j+1,1}$. Since $Q^{(6)}_{j+1,1} \subseteq Q^{(5)}_{j+1,1}$, then $p' \notin Q^{(6)}_{j+1,1}$.

    P4: As proved in step 8b, $\beta^{**}_{j,1}$ satisfies the alternate-run property with respect to $Q^{(4)}_{j+1,1}$. Hence, $Q^{(4)}_{j+1,1}$ satisfies the Multi-unique property at the end of $\beta^{**}_{j,1}$. In step 8b, we selected p' s.t., $p' \in Q^{(4)}_{j+1,1}$ and since we constructed $Q^{(6)}_{j+1,1} \subseteq Q^{(4)}_{j+1,1}$, then $p' \cup Q^{(6)}_{j+1,1}$ also satisfies the Multi-unique property at the end of $\beta^{**}_{j,1}$, which is the same state as beginning of $alone'_{c+1}$.

    P5: Process p' does not participate in $alone_l$, $\forall l \neq c+1$, as follows: run $\beta^{**}_{j,1}$ satisfies the alternate-run property with respect to $Q^{(4)}_{j+1,1}$ and $p' \in Q^{(4)}_{j+1,1}$. Condition $P3$ of the alternate-run property says that processes in $Q^{(4)}_{j+1,1}$ do not participate in $\beta^{**}_{j,1}$. Since $p' \in Q^{(4)}_{j+1,1}$, it does not participate in $\beta^{**}_{j,1}$ and it starts its execution only in $alone_{c+1}$.

    P6: As proved in step 8b, $\beta^{**}_{j,1}$ satisfies the alternate-run property with respect to $Q^{(4)}_{j+1,1}$. Hence, by condition $P8$ of the alternate-run property, the state at the end of

$\beta^{**}_{j,1}$ is unreadable with respect to $Q^{(4)}_{j+1,1}$. In step 8b, we selected p' s.t., $p' \in Q^{(4)}_{j+1,1}$. Since $Q^{(6)}_{j+1,1} \subseteq Q^{(4)}_{j+1,1}$, then the state at the end of $\beta^{**}_{j,1}$, which is the state that p' starts its execution in $alone'_{c+1}$, is unreadable with respect to $p' \cup Q^{(6)}_{j+1,1}$.

P7: In step 7, we constructed the run s.t., none of the participating processes in $clean_{c+1}$ reads the SWMR registers written in the run by active processes. In addition, in step 4, we removed from the set of processes $G_{j,1}$ all the processes that their SWMR registers are read by processes participating in $clean_{c+1}$ and denoted this set by $Q^{(1)}_{j+1,1}$. Since $Q^{(6)}_{j+1,1} \subseteq Q^{(1)}_{j+1,1}$, then processes participating in $clean_{c+1}$ do not read any SWMR register belongs to processes in $Q^{(6)}_{j+1,1}$, too.

P8: By the end of $\alpha_{j+1,1}$, the state is transparent and unreadable with respect to $Q^{(6)}_{j+1,1}$ as proved in step 8 above. The set of processes $Q^{(6)}_{j+1,1}$ satisfies the Multi-unique property at $\alpha_{j+1,1}$, as proved in step 9.

11. In run $\alpha_{j+1,1}$, only $j+1$'st MWMR register is covered, since we activated the covering processes to write in the first j MWMR registers and these covering processes are thinking in $\alpha_{j+1,1}$. We use the induction hypothesis ($i = j$) again to construct a new cover on the first j MWMR registers. We can reuse the induction hypothesis with $\alpha_{j+1,1}$ and $Q^{(6)}_{j+1,1}$, since run $\alpha_{j+1,1}$ satisfies the alternate run property with respect to $Q^{(6)}_{j+1,1}$, (as proved in step 10). Then, for a given number $e \geq 0$ (other than the number e in step 1), there is a run $\beta_{j+1,1}$ and a set of processes $Q^{(7)}_{j+1,1} \subseteq Q^{(6)}_{j+1,1}$ s.t., the first j MWMR registers written in all pseudo-solo run segments of processes in $Q^{(7)}_{j+1,1}$, starting at $\beta_{j+1,1}$, are covered.

12. By the induction hypothesis term 3, run $\beta_{j+1,1}$ satisfies the alternate-run property with respect to $Q^{(7)}_{j+1,1}$. Thus, by condition $P8$ of the alternate-run property, $Q^{(7)}_{j+1,1}$ satisfies the Multi-unique property at the end of $\beta_{j+1,1}$. If the $j+1$ MWMR register that is written in all pseudo-solo run segments starting at $\beta_{j+1,1}$ is already covered, we are done and all the lemma conditions hold for $Q^{(7)}_{j+1,1}$ and $\beta_{j+1,1}$, by the induction hypothesis.

    Else, we repeat constructing a cover on the $j+1$'st MWMR register as previously described in steps (2- 12). We denote $\beta_{j+1,v}$ as a run constructed in induction step $j+1$, in the v'th repeating time of steps (2- 12). We denote $Q^{(x)}_{j+1,v}$ as the set of processes constructed in induction step $j+1$, in the v'th repeating time of steps (2- 12), where $1 \leq x \leq 7$.

13. There is a cover on $j+1$ MWMR registers (term 2 of this lemma), since each time of repeating on steps (2- 12) and constructing a cover on the $j+1$'st MWMR register,

additional one cover on a new $j+1$'st MWMR register is achieved. After $v=d$ times, all the MWMR registers are covered, unless we finished before, either in step 12 or in step 2. In order to simplify our proof, we do not allow using a cover that was constructed in induction step $j+1$ by lower steps. Thus, it is a simple calculation of repeating up to d times on step $j+1$. By the termination condition of this loop, there is a cover on $j+1$ MWMR registers. Let $G_{j+1} = Q^{(7)}_{j+1,v}$.

14. Run $\beta_{j+1,v}$ satisfies the alternate-run property (term 3 of this lemma), by induction hypothesis as showed in step 12.

15. To complete our proof, we prove term 1 of this lemma by calculating the size of G s.t., $|G_{j+1}| \geq e$, as follows: In step 1, we used the induction hypothesis on j. Let $N(e_1, j)$ be the function that calculates how many processes are needed to construct j length cover s.t., e_1 processes are left i.e., $N(|G_{j,1}|, j)$, where $|G_{j,1}| = e_1$. On each step, we shrink the set of processes, because we select a sub-set of processes. Let g be the function of all steps that represent the construction of $Q^{(6)}_{j+1,1}$ from $G_{j,1}$ which are the steps of constructing a cover on a $j+1$'st MWMR register. Thus, the required number of processes s.t., $|Q^{(6)}_{j+1,1}| = e_2$ is: $g(N(|G_{j,1}|, j))$. In step 11, we use the induction hypothesis again and $|Q^{(7)}_{j+1,1}|$ processes remain. Thus, the required number of processes s.t., $|Q^{(7)}_{j+1,1}| = e_3$ is: $N(g(N(|G_{j,1}|, j)), j)$. In step 13, we repeat the construction of steps (3- 12) up to d times. To represent this loop, we use a helpful function $f(e,j,d)$ and the recursive function is as follows: For $j \geq 0$ and $x \geq 0$:

    - $N(e, j+1) = f(e, j, d)$
    - $f(e, j, x+1) = N(g(f(e, j, x)), j)$
    - $f(e, j, 0) = N(e, j)$
    - $N(e, 0) = e$

    We denote the solution of this recursive function by a function F_j s.t., $|G| = N(e,j) = F_j(d, k, |\alpha|, e)$. As will be shown below, $F_j(d, k, |\alpha|, e)$ depends on d, k and $|\alpha|$ because g is a function of those constants. Note that the exact solution of the recursive function is too complicated and unimportant for our result, therefore we use the notation of $F_j(d, k, |\alpha|, e)$ as a solution of the recursive function.

    Before we calculate function g, we first determine the length of $\beta_{j+1,v}$. Let h_j be the maximum length of $\beta_{j,v}$, $\beta^*_{j,v}$, $\beta^{**}_{j,v}$ and $clean_{c+1}$, for some $j, c \geq 1$ and v $(1 \leq v \leq d)$. Note that the length of each of these runs is a function of $|\beta_{j,v}|$. (Since $\beta_{j,v} \longrightarrow \beta^*_{j,v}$ and $\beta^*_{j,v} \longrightarrow \beta^{**}_{j,v}$, then it means that some active processes in run $\beta_{j,v}$ were replaced.

Since all these runs satisfy the alternate run property as showed above, then the active processes are taking steps in their pseudo-solo run segments starting at a transparent state. Therefore, each active processes in these runs is bounded to k steps.) By our construction, when $v = 1$ i.e., the first time constructing a cover on a $j+1$ MWMR register, $\beta_{j+1,1} = [\beta_{j,1} \cdot^{\sim} solo_p \cdot clean_{c+1,1}] \cdot \beta_{j,2}$. Notice that processes in the run prefix in [ ] may have replaced. Then, $|\beta_{j+1,1}| \leq h_j + k + h_j + h_j$. By step 13, we may repeat this construction up to d times, where each repeating is denoted by $\{repeat\}$ in the equivalence below: $\beta_{j+1,d} = [\{\beta_{j,1}\}\{\cdot^{\sim} solo_{p_1} \cdot clean_{c+1,1} \cdot \beta_{j,2}\} \cdots \{^{\sim} solo_{p_d} \cdot clean_{c+1,d} \cdot \beta_{j,d+1}\}]$. We denote the length of $\beta_{j+1,d}$ by h_{j+1} and $h_{j+1} \leq h_j + (k + h_j + h_j) + ... + (k + h_j + h_j) = h_j + d * (k + 2h_j) \leq d * (k + 3h_j)$. Thus, the recursive function is $h_{j+1} \leq d * (k + 3h_j)$ and $h_0 = |\alpha|$, since $\beta_0 = \alpha$. To simplify our recursive function calculation, we increase the size by replacing addition with multiplying s.t., $h_{j+1} \leq d*k*3h_j$. Solving this function, results with $h_j \leq |\alpha| * [3d*k]^j$.

The function g is as follows: In step 4, $|Q^{(1)}_{j+1}| = |G_{j,1}| - |clean_{c+1}|$, then $|Q^{(1)}_{j+1}| \geq |G_{j,1}| - h_j$. In step 7, $|Q^{(2)}_{j+1}| \geq |Q^{(1)}_{j+1}| - c$. In step 8a, $|Q^{(3)}_{j+1,1}| \geq \frac{|Q^{(2)}_{j+1,1}|}{(2k+1)*(k+1)}$, then $|Q^{(3)}_{j+1,1}| \geq \frac{|G_{j,1}| - h_j - c}{(2k+1)*(k+1)}$. In step 8b, $|Q^{(4)}_{j+1,1}| \geq |Q^{(3)}_{j+1,1}| - c$ and $Q^{(5)}_{j+1,1} = Q^{(4)}_{j+1,1} - \{p'\}$. Thus, $|Q^{(5)}_{j+1,1}| \geq [\frac{|G_{j,1}| - h_j - c}{(2k+1)*(k+1)}] - c - 1 = \frac{|G_{j,1}| - h_j - c}{(2k+1)*(k+1)} - (c+1)$. In step 9, $|Q^{(6)}_{j+1,1}| \geq \frac{|Q^{(5)}_{j+1,1}|}{[2d]^k}$, then $|Q^{(6)}_{j+1,1}| \geq \frac{\frac{|G_{j,1}| - h_j - c}{(2k+1)*(k+1)} - (c+1)}{[2d]^k}$.

Calculating function g is by setting $Q^{(6)}_{j+1,1} = x$ and $|G_{j,1}| = g(x)$, as follows: $g(x) \leq [x * (2d)^k + c + 1] * [(2k+1) * (k+1)] + h_j + c$.

The constant c depends on the length of $\beta_{j+1,v}$ and we showed that $h_j \leq |\alpha| * [3d*k]^j$. Therefore, we say that the function g depends on the parameters: d, k, j and $|\alpha|$.

■

4.5 The Violation Lemma

Lemma 4.6 [Violation] *For a long-lived k-solo-adaptive WT&S algorithm W with at most d MWMR registers, there is a large enough number $n_{d,k}$ s.t., if there are more than $n_{d,k}$ processes in W, then there is a run α and there are two processes p and q $(p \neq q)$ that are* eating *at the end of α.*

High level description of the Proof of Lemma 4.6: See Section 3.2.

Proof of Lemma 4.6.

Proof is similar to step $i = k+1$ in Lemma 4.5, but simpler. We first construct a run α and then we calculate the size of $n_{d,k}$.

Constructing run α:

1. We use Lemma 4.5, with e (that is calculated by the end of this proof) and the empty run. By definition, the empty run satisfies the alternate-run property with respect to all processes. By Lemma 4.5, there is a number $N(e,k)$ s.t., if there are $N(e,k)$ processes in W denoted by P, then there is a run β_k and a set of processes $G_k \subseteq P$ s.t.,
   (a) $|G_k| \geq e$,
   (b) all the MWMR registers written in all the pseudo-solo run segments of processes in G_k, after β_k, are covered in β_k,
   (c) run β_k satisfies the alternate-run property with respect to G_k.

2. Suppose that run $\beta_k = alone_1 \cdot clean_1 \ldots alone_c \cdot clean_c$ for some constant $c \geq 0$. We construct an extension $^{\sim}solo_p \cdot clean_{c+1} \cdot^{\sim} solo_q$ to run β_k by first constructing run segment $clean_{c+1}$, as follows: By step 1b, there are exactly k' $(0 \leq k' \leq k)$ processes $u_1, ..., u_{k'}$ that cover all the MWMR registers $r_1, ..., r_{k'}$, written in all the pseudo-solo run segments of processes in G_k, when starting after β_k. We construct a run segment $clean_{c+1}$ similarly to the definition of *clean* run segment in the alternate-run property (condition $P2$). First we let each process u_i to write in r_i and then we let all the processes $u_1, ..., u_{k'}$ to continue until thinking.

3. We select a sub-set of processes G'_k from G_k s.t., none of the participating processes $u_1, ..., u_{k'}$ in $clean_{c+1}$ reads the SWMR registers written by processes in G'_k. Since there are at most $|clean_{c+1}|$ read operations in $clean_{c+1}$, then there are at most $|clean_{c+1}|$ SWMR registers that are read. By the one-to-one correspondence between SWMR registers and the processes that are writing in them, we can remove these processes from G_k and the remained set of processes is denoted by G'_k. Thus $|G'_k| = |G_k| - |clean_{c+1}|$.

   To show that the set G'_k exists, we show that $clean_{c+1}$ has a constant length. Since W is an adaptive algorithm, then the number of operations that processes are allowed to perform in $clean_{c+1}$, depends on the number of active processes in the run. By step 1c, run β_k has at most a constant c of active processes, (because in each *alone* run segment in β_k, a new process is starting its execution). Thus, $|clean_{c+1}| = f(c)$, where $f(c)$ is a function of a constant c.

4. We construct a run segment $^\sim solo_p$ as follows: Select any process $p \in G'_k$ and a pseudo-solo run segment $^\sim solo_p$ s.t., $^\sim solo_p$ starts at the end of run β_k, and in $^\sim solo_p$ p writes in $r_1, ..., r_{k'}$ and starts eating.

   There exists $^\sim solo_p$ as above, because of the following:

   (a) By step 1c, β_k satisfies the alternate-run property with respect to G_k. Condition $P8$ of the alternate-run property says that the state by end of a run is transparent. Therefore, the state at the end of run β_k is transparent with respect to G_k. By this, every process in G_k has pseudo-solo run segments after run β_k and specifically $p \in G'_k \subseteq G_k$.

   (b) Again because β_k satisfies the alternate-run property with respect to G_k, then by condition $P8$, G_k satisfies the Multi-unique at the end of β_k. Thus, after run β_k all the MWMR registers written in all the longest pseudo-solo run segments of processes in G_k, are the same. Since $p \in G'_k$ and $G'_k \subseteq G_k$, then p writes in the same MWMR registers that are covered i.e., $r_1, ..., r_{k'}$.

5. The state at the end of run $\beta_k \cdot^\sim solo_p \cdot clean_{c+1}$ may not be transparent with respect to G'_k, if processes in $clean_{c+1}$ read x SWMR registers written by active processes in run β_k, for some x ($x \geq 1$). Since run β_k satisfies the alternate-run property with respect to G_k and $G'_k \subseteq G_k$, then by Lemma 4.4 there is a run β^*_k s.t., all active processes in β^*_k are only from G'_k, $\beta_k \longrightarrow \beta^*_k$, β^*_k satisfies the alternate-run property with respect to a set of processes G''_k and $G''_k \subseteq G'_k$. In Lemma 4.4, the size of G''_k is calculated: $|G''_k| = |G'_k| - x - 1$. (We removed additional one, since we selected process p to participate in the run). Since there are at most c active processes in β_k, then $x \leq c$ and $|G''_k| \geq |G'_k| - c - 1$.

   Processes in $clean_{c+1}$ cannot read the SWMR register that p writes during $^\sim solo_p$ since $p \in G'_k$.

6. If by the end of run $\beta = \beta^*_k \cdot^\sim solo_p \cdot clean_{c+1}$ there is a process $q \in G''_k$ s.t., the state by the end of β is transparent with respect to q, then let $^\sim solo_q$ be the longest pseudo-solo run segment of q starting after β and in which q eats.

   Otherwise, select any process $q \in G''_k$. (Notice that q does not execute any pseudo-solo run segment after β.) Let Q be a set of processes s.t., q reads the SWMR registers belongs to processes in Q. Since there are maximum $c+1$ active processes in β and q can read up to k registers in its pseudo-solo run segment, then $|Q| \leq max(c+1, k) \leq c+1+k$. By removing Q from G''_k, then the remained set is G'''_k and $G'''_k = G''_k - Q - \{q\}$. Thus, $|G'''_k| \geq |G''_k| - (c+k+1) - 1$. In step 5, we constructed a new run β^*_k which satisfies the alternate-run property with respect to G''_k. We use Lemma 4.4 again on β^*_k and replace processes in Q with other processes from G'''_k s.t., the new run is β^{**}_k, $\beta^*_k \longrightarrow \beta^{**}_k$, and β^{**}_k satisfies the alternate-run property with respect to G^{IV}_k s.t., $G^{IV}_k \subseteq G'''_k$ and $|G^{IV}_k| = |G'''_k| - c$.

If process $p \in Q$ and p participates in $^{\sim}solo_p$, then we select another process $p' \in G^{IV}_k$ and its pseudo-solo run segment $^{\sim}solo_{p'}$ instead of $^{\sim}solo_p$ s.t., p' is writing in the same MWMR registers and p' eats, as p. Thus, $G^{IV}_k = \{p'\}$.

7. Finally, run $\beta^{**}_k \cdot^{\sim} solo_{p'} \cdot clean_{c+1} \cdot^{\sim} solo_q$ has two processes p' and q that are eating, concurrently.

Calculating the number $n_{d,k}$, as follows:

By Lemma 4.5, given a number e there is a number $N(e,k)$ s.t., G_k and run β_k fulfill all the properties in step 1 above. Below, we show that for $n_{d,k} = N(e,k)$, $n_{d,k}$ is a function of d and k.

The calculation the size of G_k $(= e)$ that is required to construct the violation run in this lemma, is as follows: In step 3, $|G'_k| = |G_k| - |clean_{c+1}|$. In step 5, $|G''_k| \geq |G'_k| - c - 1$. In step 6, $|G'''_k| \geq |G''_k| - c - k - 2$, $|G^{IV}_k| = |G'''_k| - c$ and $G^{IV}_k = \{p'\}$. We denote the length of β_k and $clean_{c+1}$ by h_k, similarly to the calculation done in Lemma 4.5 and $h_k \leq |\alpha| * [3 * d * k]^k$. Since α is the empty run, we define $|\alpha| = 1$ and $h_k \leq [3 * d * k]^k$. Therefore, since $|G_k| = h_k + c + 1 + c + k + 2 + c + 1$, then $e = |G_k| = [3 * d * k]^k + 3c + k + 4$. The constant c is the number of processes involved in β_k, which is bounded by the length of β_k. We calculate $h_k \leq [3 * d * k]^k$, thus $c \leq h_k$ depends only on d and k. Therefore, the size of G_k is also a function of d and k i.e., $e = |G_k| = 4 * [3 * d * k]^k + k + 4$.

Finally, the number $n_{d,k}$, is the number of processes required for constructing run β_k and the violation run, which means $n_{d,k} = N(e,k) = F_k(d, k, |\alpha|, e)$, where $e = |G_k|$ and $N(e,k), F_k(d, k, |\alpha|, e)$ are as defined in Lemma 4.5. We calculate e as a function of d and k and $|\alpha| = 1$. Hence, $n_{d,k}$ is a function of d and k.

■

5 Concluding Remarks

In the run we constructed in the main theorem's proof, each process participates at most once. I.e., no process in the constructed run invokes a WT&SET operation twice. One might think that therefore the proof is for single-shot WT&S objects, but no. In the constructed run processes that attempt to capture the WT&S bit and fail are assumed to leave the WT&S object in an idle state. That is, the object may be captured and released several times, namely is long-lived. Processes, however, access it each at most one time. Perhaps a more sophisticated construction would reuse the processes and prove a stronger bound for long-lived objects in which the same set of processes access the object multiple times.

In a single shot WT&S object, the object may be captured at most one time. Furthermore, if some processes attempted to capture the WT&Set bit and failed, then no later process will succeed in capturing the bit. Notice that such an object has an adaptive implementation with just two MWMR registers following the X and Y MWMR registers construction used by Lamport in [16] (and which is the basis of the splitter object [17]).

There is an implementation of adaptive collect with a function of n MWMR registers, but the most space efficient adaptive renaming object has an implementation with $N >> n$ MWMR registers, where N is the total number of processes in the system, while n is the maximum number of concurrent processes. This leaves an open question, if there is an adaptive WT&S algorithm for implementing renaming object in the Read/Write Shared Memory model that uses a function of n MWMR registers?

References

[1] Y. Afek, P. Boxer, and D. Touitou. Bounds on the shared memory requirements for long-lived and adaptive objects. In *Proc. of the Nineteenth Ann. ACM Symp. on Principles of Distributed Computing (PODC)*, July 2000.

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

[3] H. Attiya and A. Fouren. Adaptive long-lived renaming with read and write operations. Technical Report 0956, Faculty of Computer Science, Technion, Haifa, 1999. http://www.cs.technion.ac.il/∼hagit/pubs/tr0956.ps.gz.

[4] Y. Afek, G. Stupp, and D. Touitou. Long lived adaptive collect with applications. In *Proc. of the 40th IEEE Ann. Symp. on Foundation of Computer Science*, pages 262–272, October 1999.

[5] Y. Afek, H. Attiya, G. Stupp, and D. Touitou. Adaptive long-lived renaming using bounded memory. Submitted to DISC99. ftp://ftp.math.tau.ac.il/pub/stupp/PAPERS/name99.ps.gz, 1999.

[6] James H. Anderson and Jae-Heon Yang. Time/contention trade-offs for multiprocessor synchronization. *Information and Computation*, 124(1):68–84, 10 January 1996. Also in STOC94, 224-233.

[7] Y. Afek, E. Gafni, and M. Merritt. Adaptive algorithms utilyzing adaptive collect and snapshot. Submitted for publication, ftp://ftp.math.tau.ac.il/pub/afek/1shotRn.ps, 1999.

[8] J. Anderson and J. Yang. Time/contention trade-offs for multiprocessor synchronization. In *Proc. of the 26th Ann. ACM Symp. on Theory of Computer*, pages 224–233, May 1994.

[9] M. Merritt and G. Taubenfeld. Knowledge in shared memory systems. In *Proc. of the 10th Ann. ACM Symp. on Principles of Distributed Computing*, pages 189–200, August 1991.

[10] L. Lamport. The mutual exclusion problem, part I: A theory of interprocess communication. *J. of the ACM*, 33(2):313–326, April 1986.

[11] Jae-Heon Yang and James H. Anderson. Time bounds for mutual exclusion and related problems (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 224–233, Montréal, Québec, Canada, 23–25 May 1994.

[12] G. L. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12(3):115–116, 1981.

[13] Y. Afek, E. Gafni, J. Tromp, and P. M. B. Vitányi. Wait-free test-and-set. In *Proc. of the 6th Int. Workshop on Distributed Algorithms: Lecture Notes in Computer Science, 647*, pages 85–94. Springer Verlag, November 1992.

[14] H. Attiya and A. Fouren. Adaptive long-lived renaming with read and write operations. Extended Abstract, November 1998.

[15] Y. Afek, H. Attiya, A. Fouren, G. Stupp, and D. Touitou. Long-lived renaming made adaptive. In *Proc. 18th Annual ACM Symp. on Principles of Distributed Computing*, pages 91–103, May 1999.

[16] L. Lamport. A fast mutual exclusion algorithms. *ACM Trans. on Computer Systems*, 5(1):1–11, February 1987.

[17] M. Moir and J. Anderson. Wait-free algorithms for fast, long-lived renaming. *Science of Computer Programming*, 25(1):1–39, October 1995. Also in Proc. 8th Int. Workshop on Distributed Algorithms, September 1994, 141-155.