LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-462

AN ANALYSIS OF RABIN'S RANDOMIZED MUTUAL EXCLUSION ALGORITHM: PRELIMINARY REPORT

Nancy Lynch
Isaac Saias

December 1991

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

An Analysis of Rabin's Randomized Mutual Exclusion Algorithm: Preliminary Report*

Isaac Saias[†] Nancy Lynch[‡]

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

November 1, 1991

Abstract

In 1982, Michael Rabin published a randomized distributed algorithm implementing mutual exclusion for n processes using a read-modify-write primitive on a shared variable with $O(\log n)$ values. He claimed that this algorithm satisfied the following informally-stated *strong probabilistic no-lockout* property. Define the *adversary* to be the entity controlling the order in which processes take steps; then, for every adversary, any process competing for entrance to the critical section succeeds with probability $\Omega(1/m)$, where m is the number of competing processes.

In this paper we consider several different ways in which this property can be expressed formally. We express explicitly the dependency of the probability on the adversary and show that this dependency is so strong that the algorithm does not satisfy *any* of these conditions. In fact, the algorithm does not even satisfy a much weaker $\Omega(1/n)$ property.

Key words: mutual exclusion, randomized algorithm, no-lockout, bounded waiting, shared memory, read-modify-write, conditional probability.

*Research supported by research contracts ONR-N00014-91-J-1046, NSF-CCR-8915206 and DARPA-N00014-89-J-1988.

[†] e-mail: saias@theory.lcs.mit.edu

[‡] e-mail: lynch@theory.lcs.mit.edu

1 Introduction

The problem of *mutual exclusion* [2] involves allocating an indivisible, reusable resource among n competing processes. A mutual exclusion algorithm is said to guarantee *progress* if it continues to allocate the resource as long as at least one process is requesting it. It guarantees *no-lockout* if every process that requests the resource eventually receives it. A mutual exclusion algorithm satisfies *bounded waiting* if there is a fixed upper bound on the number of times any competing process can be bypassed by any other process. In conjunction with the progress property, the bounded waiting property implies the no-lockout property. All of these notions are defined with respect to an *adversary*, which is the entity controlling the order in which processes take steps.[1] In 1982, Burns et al.[1] considered the mutual exclusion algorithm in a distributed setting where processes communicate through a shared read-modify-write variable. For this setting, they proved that any *deterministic* mutual exclusion algorithm that guarantees progress and bounded waiting requires that the shared variable take on at least n distinct values. Shortly thereafter, as part of a general program of investigating the power of randomization in algorithms, Rabin published a *randomized* mutual exclusion algorithm [4] for the same shared memory distributed setting. His algorithm guarantees progress using a shared variable that takes on only $O(\log n)$ values.

It is quite easy to verify that Rabin's algorithm guarantees mutual exclusion and progress; in addition, however, Rabin claimed that his algorithm satisfies the following informally-stated *strong no-lockout* property. (In the statement of this property, a "trying round" refers to the interval between two successive allocations of the resource, and the "critical region" refers to the interval during which a particular process has the resource allocated to it. A "critical region" is also called a "critical section".)

"*If process i participates in a trying round of a run of a computation by the protocol and compatible with the adversary, together with $0 \leq m - 1 < n$ other processes, then the probability that i enters the critical region at the end of that round is at least c/m, $c \sim 2/3$.*" (*)

This property says that the algorithm guarantees an approximately equal chance of success to all processes that compete at the given round. Rabin argued strongly in [4] that a good randomized mutual exclusion algorithm should satisfy this strong no-lockout property, and in particular, that the probability of each process succeeding should depend inversely on m, the number of *actual* competitors at the given round. This dependence on m was claimed to be an important advantage of this algorithm over another algorithm developed by Ben-Or (also described in [4]); Ben-Or's algorithm is claimed to satisfy a weaker no-lockout property in which the probability of success is approximately c/n, where n is the total number of processes, i.e., the number of *potential* competitors.

Rabin's algorithm uses a randomly-chosen round number to conduct a competition for each round. Within each round, competing processes choose lottery numbers randomly, according to a truncated geometric distribution. One of the processes drawing the largest lottery number for the round wins. Thus, randomness is used in two ways in this algorithm: for choosing the round

[1] We give more formal definitions of these properties in Section 3.

numbers and choosing the lottery numbers. The detailed code for this algorithm appears in Figure 1.

Rabin's description of the strong no-lockout property was informal; in particular, the paper did not include a careful definition of the probability space of algorithm executions by which the probabilities of various events were to be measured. Thus, we attempted to provide the needed definitions and fill in the formal details of the probabilistic analysis. The surprising result of our project was the discovery that the algorithm does not have the properties claimed.

The difficulty that we are faced when attempting to prove a no-lockout statement is that there are actually *two* probability spaces: the probability space of the random inputs and the one of the random executions. The space considered in Rabin's statement (*) is the space of executions. The space considered by Rabin in his correctness proof is the mathematical space of the inputs. Of course the probability measure "$\mathbf{dP}'$" living on the random executions is derived from the probability $\mathbf{dP}$ living on the random inputs. But, as we will show, it is not derived from it as a simple image measure through a deterministic function f in such a way that $\mathbf{dP}' = f(\mathbf{dP})$. By its decisions, the adversary, (along with the code of the algorithm), determines the connection between these two measures. As a consequence the measure $\mathbf{dP}'$ is actually more correctly denoted by $\mathbf{dP}_{\mathcal{A}}$. This notation reflects precisely the fact that the adversary has some important control on the underlying probability space of statement (*). This statement expresses a lower bound of some event W' when measured by $\mathbf{P}_{\mathcal{A}}$: "$\mathbf{P}_{\mathcal{A}}(W' \ldots) \geq \ldots$" The proof of such a statement can be achieved only by expressing (actually lower-bounding) this probability in terms of the probability of some event W under the input probability $\mathbf{dP}$, over which the adversary has, by definition, no control. This was the implicit goal of Rabin when he proved his mathematical lemmas 3.2 and 3.3 (In our terms these results are of the type "$\mathbf{P}(W) \geq \ldots$") The problem that we uncovered is that such a connection *cannot* be established so that Rabin's claim (*) does not hold in spite of the fact that his mathematical results are correct. In Section 9.2 we provides a result whose proof shows in a striking way the problems at stake.

A related problem is that the random variables that Rabin considers in the proof sketches in his paper are *independent* whereas the formal versions of the no-lockout property involve *dependent* random variables (the values of the program variables) as well as the *adversary*. In particular, in the analysis of his algorithm, Rabin uses the following implicit argument: "The lottery value held by a process j having lost at its last round is stochastically smaller then a newly-chosen random value. Consequently, the probability that process i wins in round k is bounded from below by the probability obtained when assuming that (i.e., when conditioning on the fact that) all other participating processes hold new lottery values in round k." This argument is flawed because, as we show in theorem 7.2, the adversary can actually "manufacture" processes holding very high lottery values, even though they have lost at their last rounds.

We begin by presenting four different formal versions of the no-lockout property. These four statements involve conditional probabilities and give lower bounds on the (conditional) probability that a participating process wins the current round of competition. They differ by the nature of the events involved in the conditioning and by the values of the lower bounds. Described in this formal style, the strong no-lockout property claimed by Rabin involves conditioning over an event *posterior* to the point at which the probability is evaluated. We show in Theorem 6.1 that the adversary can use this fact in a simple way to lock out any process during any round.

We then go on to show in Theorem 7.2 that the algorithm suffers from a much more severe flaw, which bars it from satisfying even the weaker c/n no-lockout property claimed for Ben-Or's algorithm: we show that a set of linear size can be systematically locked out with probability exponentially close to 1.

We first remark that the role of the randomly-chosen round numbers is to make the lottery numbers drawn in previous rounds obsolete. The weak no-lockout property would hold if the set of round numbers used by the algorithm were infinite and if a previously-unused round number were used at each round; however, since the current round number always resides in the shared variable, it is not possible to use this strategy within the constraint of bounded size shared memory. An alternative is to use only a constant number of round numbers and to recycle them deterministically; however, in this case, the adversary can use its knowledge of the current round number and of the past run to lock out some processes. Rabin's algorithm makes the choice of the round numbers random in an attempt to prevent the adversary from collecting any significant knowledge about the past. However, our results demonstrate that this attempt does not succeed. In fact, the algorithm fails to satisfy the weak c/n no-lockout property for any *sub-linear* sized set of round numbers.

The algorithm of Ben-Or which is presented at the end of [4] is a modification of Rabin's algorithm that uses a shared variable of *constant* size. The methods that we develop in the analysis of Rabin's algorithm, in particular, those for Theorem 7.2, establish that Ben-Or's algorithm is similarly flawed, in particular, it does not satisfy the $1/2en$ no-lockout property claimed for it in [4].

The remainder of this paper is organized as follows. Section 2 recalls quickly the non-deterministic setting of probabilistic protocols. Section 3 contains a description of the mutual exclusion problem and formal definitions of the strong and weak no-lockout properties.

Section 4 presents Rabin's algorithm. In Section 5 we formalize Rabin's no-lockout property and express his proof in our formal terms. This proof is based on three inequalities that we disprove in the next three sections. In Section 6 we show that the strong $\Omega(1/m)$ no-lockout property does not hold. The proof stems from the fact that the conditioning of the strong no-lockout property provides the adversary with some a-priori knowledge of the execution to come. In Section 7 we prove that the algorithm does not even verify the weak $\Omega(1/n)$ property and that the algorithm will suffer from the same flaw as long as the size of the shared variable is kept sub-linear. The proof involves constructing a complex adversary that exploits fully the knowledge of the past run. It is worth noting that these two adversaries use completely different methods to achieve their goal. The first one uses the knowledge of m (the number of participating processes). The second uses a control it is able to exert on the lottery values. In Section 8 we show that the adversary is also able to infer (partially) the value of the round number, in opposition to Rabin's claim. In Section 9 we show precisely where Rabin's argument fails in the use of his mathematical Lemma 3.3. We finally give a positive result. We think that this result exemplifies very clearly the correct logical argumentation that has to be given while proving the correctness of a distributed probabilistic protocol. We finish with Section 10 that contains a general mathematical Lemma used in Section 8.

2 General Conventions about Distributed Stochastic Systems

A generic execution of a probabilistic distributed protocol is identified uniquely with two parameters ω and $\mathcal{A}$:

1. ω specifies any random choices of a particular execution.
2. $\mathcal{A}$ factors out all the other sources of nondeterminism in the protocol: $\mathcal{A}$ specifies at each point of the distributed execution which of the processes takes a next step within its own code.

We refer to such a generic execution as $\mathcal{E}(\omega, \mathcal{A})$.

For a given adversary $\mathcal{A}$, $\mathcal{E}(\omega, \mathcal{A})$ represents a random execution i.e., a stochastic process: the code description of a distributed protocols provides the definition of the 1-step transitions defining a distributed Markov process.

For any real function X defined on the set of executions and any adversary $\mathcal{A}$, $\omega \rightarrow X(\mathcal{E}(\omega, \mathcal{A}))$ is a real random variable.

3 Statement of the Mutual Exclusion Problem and Notations

We define here the problem of mutual exclusion in its generality, and describe what properties a solution to this problem is expected to have. The consideration of these properties lead us to specify the type of adversary that is suited to this model.

3.1 Statement of the Problem

The problem of mutual exclusion is the problem of the allocation of an indivisible resource among competing processes. A solution to this problem will be a probabilistic distributed algorithm described by a code $\mathcal{C}$ having the following properties:

1. All the processes involved run the same code $\mathcal{C}$.
2. $\mathcal{C}$ is partitioned into four regions Try, Crit, Exit and Rem. During any execution, any process progressing through the code goes cyclically through the regions Rem, Try, Crit, Exit, Rem, ..., in this order.
3. A process being in Crit is said to hold the resource. The indivisible property of the resource translates into the fact that at any point of any execution, at most one process can be in Crit.
4. For any process i, if i is in Crit at some point of the execution, i must undergo a change of region after this point (i.e., must go to Rem).

We will furthermore require any "good" solution to satisfy the additional properties of *progress* and *no-lockout*, that we will define. For this purpose we will define the notions of *run*, *round*, *adversary*, and *fair* adversary.

3.2 Definition of Runs, Rounds, and Adversaries

- A *run* is a finite or infinite sequence $\{(i_1, old_1, new_1), (i_2, old_2, new_2), \ldots, (i_t, old_t, new_t)\ldots\}$ where:
    - i_t is the label of a process,
    - old_t and new_t are taken from {Try, Crit, Exit, Rem}.
- Such a run is said to be the run of a (partial) execution $\mathcal{E}$ if i_t takes the t^{th} step in $\mathcal{E}$, and if process i_t undergoes the region change $old_t \rightarrow new_t$ during this step. Conceivably, $old_t = new_t$. (But, if new_t is different from old_t, it must follow old_t in the order indicated in 2. of Section 3.1.)
- If a run r is the run of a (partial) execution $\mathcal{E}$ then we say that $\mathcal{E}$ is *compatible* with r. Notice that there may be many executions (corresponding to different local behaviors of the processes) compatible with a given run.
- An *adversary* is a mapping $\mathcal{A}$ from the set of finite runs to the set $\{1, \ldots, n\}$ determining what process is to take its next step as a function of the current prefix of the current run.
- A run $\{(i_1, old_1, new_1), (i_2, old_2, new_2), \ldots\}$ is said to be *compatible* with an adversary $\mathcal{A}$ if $\mathcal{A}[\{(i_1, old_1, new_1), \ldots, (i_t, old_t, new_t)\}] = i_{t+1}$ for every t.
- An adversary $\mathcal{A}$ is said to be *fair* if, for every infinite run compatible with $\mathcal{A}$ and every process i, the following condition holds: if the last occurrence of i appears in the run as i_t, then $new_t =$ Rem. (That is, if i takes only a finite number of steps, then its last step leaves it in the remainder region.).
- We define a *round* of an execution to be a sequence of process steps from the time one process enters its critical region until the time the next process enters its critical region; formally, a *round* of an execution is a maximal execution fragment of the given execution, containing one transition Try → Crit at the end of the execution fragment, and containing no other transition Try → Crit.
- In a same way we define a round of a run to be a maximal run fragment of the given run, containing one transition Try → Crit at the end of the run fragment, and containing no other transition Try → Crit.
- We say that a process i *participates* in a round if i undergoes a region change Try → Crit, Rem → Try or Try → Try during this round.

In the definition of the adversary that we consider here, the adversary has no access to the values of the local and of the shared variables. The only thing that is visible to the adversary is the position of each process among the four regions Try, Crit, Exit, Rem that partition its local code.

3.3 A Definition of Progress and four Definitions of No-Lockout

Definition 3.1 An algorithm C that solves mutual exclusion satisfies *progress* or equivalently is *deadlock-free* if, for all fair adversaries, there is no infinite execution in which, from some point on, at least one process is in its Try region and no transition Try → Crit occurs.

Note that, so far, all the properties that we considered are deterministic.The no-lockout property is probabilistic. Its formal definition requires the following notation:

Let X denote any generic quantity whose value changes as the execution unfolds (e.g., a program variable). Then we will denote $X(k,\omega,\mathcal{A})$ the value held by X *just prior* to the last step (Try → Crit) of round k of the execution defined by $(\omega,\mathcal{A})$. We will usually omit the parameters $\omega,\mathcal{A}$, writing $X(k)$ in place of $X(k,\omega,\mathcal{A})$.

As a special case of this general notation, we define

- $\mathcal{P}(k)$ to be the set of participating processes in round k. (Set $\mathcal{P}(k) = \emptyset$ if $\mathcal{E}$ has less then k rounds.) The notation $\mathcal{P}(k)$ is consistent with the general notation because the set of processes participating in round k is updated as round k progresses: the definition of this set is complete only at the end of round k. This remark is at the core of our Theorem 6.1.

- For each i and k, we let $W_i(k)$ denote the set of executions in which process i enters the critical region at the end of round k.

We constantly use the probabilistic convention according to which, for any property $\mathcal{S}$, the set of executions $\{\mathcal{E} : \mathcal{E}$ has property $\mathcal{S}\}$ is denoted as $\{\mathcal{S}\}$. Then:

- For each k and $\mathcal{E}$ we let $\pi_k(\mathcal{E})$ denote the run compatible with the first k rounds of $\mathcal{E}$. Then for any k-round run ρ, $\{\pi_k = \rho\}$ represents the set of executions compatible with ρ. ($\{\pi_k = \rho\} = \emptyset$ if ρ has less then k rounds.)

- Similarly, for all $m \leq n$, $|\{\mathcal{P}(k)\}| = m$ represents the set of executions having m processes participating in round k.

The quantities $\{\pi_k = \rho\}$, $W_i(k)$, $\{|\{\mathcal{P}(k)\}| = m\}$ are sets of executions. Recall that an execution is uniquely characterized when the probabilistic choices ω and the adversary $\mathcal{A}$ are fixed. (Note also that in $\{\pi_k = \rho\}$ the adversary is characterized only up to round k.) For a given adversary, the executions depend only on the random choices and the previous sets can be viewed as random events. Their probability space is the space of executions endowed, for each given adversary $\mathcal{A}$, with the probability distribution $\mathbf{P}_{\mathcal{A}}$ induced by the random inputs.

The no-lockout property that we introduce gives a lower bound l on the probability of success of any participating process i during a round k (this event is $W_i(k)$). This lower bound must hold against *all* adversaries. This property takes different forms according to the point t_k of the execution at which the probability is estimated and according to the value l of the lower bound. (The subscript in t_k is used to emphasize that the probabilities are evaluated at different points for different rounds.)

At point t_k, the adversary has knowledge of the past run (up to t_k). Hence, in the game that is played between the user of the algorithm (running its code) and the adversary (taking decisions so as to minimize the probability of $W_i(k)$), the two players hold the knowledge $\{\pi_k = \rho\}$ and play accordingly. (Note that $\{\pi_k = \rho\}$ is part of the state of the system so that the algorithm "knows" it.) This means that the set of executions is restricted to $\{\pi_k = \rho\}$ i.e., that the probability of the no-lockout statement is conditioned on this event. To summarize, the conditioning of the no-lockout statement must include the knowledge of the adversary.

Conversely, consider the event over which conditioning is done in the no-lockout statement. (We argued that this event is a superset of $\{\pi_k = \rho\}$.) As the no-lockout property must hold for *all* adversary we can assume that we are dealing with the worst case adversary: this adversary takes decisions as if knowing the event we condition on.

This shows that there is a natural connection between the knowledge held by the adversary at the point t_k and the probabilistic conditioning present in the no-lockout property. This fact is crucially highlighted in Theorem 6.1. Of course, after the point t_k, as the execution proceeds into round k, the knowledge held by the adversary is updated with the nature of the current run.

The first version of the no-lockout property states that, at the beginning of each round k, each participating process has a (weak) c/n probability of success at round k.

Definition 3.2 (Weak, Run-knowing, Probabilistic No-lockout) A solution to the mutual exclusion problem satisfies *weak* probabilistic no-lockout for run-knowing adversary whenever there exists a universal (i.e., independent of k and n) constant c such that, for every fair adversary $\mathcal{A}$, every $k \geq 1$, every $(k-1)$-round run ρ compatible with $\mathcal{A}$, and every process i,

$$\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \geq c/n,$$

whenever $\{\pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\} \neq \emptyset$.

Definition 3.3 (Strong, Run-knowing, Probabilistic No-lockout) A solution to the mutual exclusion problem satisfies *strong* probabilistic for run-knowing adversary no-lockout whenever there exists a universal constant c such that, for every fair adversary $\mathcal{A}$, every $k \geq 1$, every $(k-1)$-round run ρ compatible with $\mathcal{A}$, every process i, and every $m \leq n$,

$$\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m\Big] \geq c/m,$$

whenever $\{\pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m\} \neq \emptyset$.

Due to our model of adversary, $\{\pi_{k-1} = \rho\}$ represents exactly the knowledge of the past execution that the adversary has available at the *beginning* of round k. Hence, conditioning $W_i(k)$ on $\{\pi_{k-1} = \rho\}$ means that the point t_k at which the probability is evaluated is the beginning of the kth round.

Another definition, which amounts to give the adversary more knowledge about the state of the system, is to condition on the complete value of the $k-1$-round execution compatible with $\mathcal{E}$, instead of conditioning on the value of the $k-1$-round run compatible with the execution $\mathcal{E}$. Formally, let $\pi'_k(\mathcal{E})$ denote the k-round execution compatible with $\mathcal{E}$. Then the corresponding weak no-lockout property is:

Definition 3.4 (Weak, Execution-knowing, Probabilistic No-lockout) There exists a universal (i.e., independent of k and n) constant c such that for every fair adversary $\mathcal{A}$, every $k \geq 1$, every $(k-1)$-round execution ϵ compatible with $\mathcal{A}$, and every process i,

$$\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \pi'_{k-1} = \epsilon,\ i \in \mathcal{P}(k)\Big] \geq c/n,$$

whenever $\{\pi'_{k-1} = \epsilon,\ i \in \mathcal{P}(k),\} \neq \emptyset$.

Some other alternative definitions are possible that consider an adversary of a still different strength. For instance we can consider the *average* case over all past executions, in which by definition no specific knowledge is assumed about the past.

Definition 3.5 (Strong, Without past knowledge, Probabilistic No-lockout) A solution to the mutual exclusion problem satisfies *strong* probabilistic *no-lockout* with no knowledge whenever there exists a universal constant c such that, for every fair adversary $\mathcal{A}$, every $k \geq 1$, every process i, and every $m \leq n$,

$$\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m\Big] \geq c/m,$$

whenever $\{i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m\} \neq \emptyset$.

This definition is tantamount to fix t_k as being the beginning of the execution.

By integration over ρ we see that an algorithm having the property of Definition 3.3 is stronger then one having the property of Definition 3.5. Equivalently, an adversary able to falsify Property 3.5 is stronger then one able to falsify Property 3.3.

4 Rabin's Algorithm

The algorithm that we analyze here was published in 1982 by Michael Rabin [4]. It is a probabilistic algorithm implementing mutual exclusion for n processes using a read-modify-write primitive on a shared variable with $O(\log n)$ values. The code is given in Figure 1.

This algorithm uses randomness in two ways: on the one hand, randomness allows a reduction of the size of the shared variable. This technique is very similar to the one developed by Flajolet and Martin [3] in their probabilistic approximate counting scheme. On the other hand, randomness is used to foil the knowledge accessible to the adversary about the state of the system.

It is easily verified that this algorithm verifies mutual exclusion (the entry of the critical section is guarded by a semaphore accessed in a read-modify-write fashion!), and progress. The core of the discussion and the purpose of this paper is to analyze the no-lockout property.

4.1 Description of the Algorithm

The basic idea of Rabin's algorithm is that each round of competition for entry to the critical region consists of a lottery in which each process draws a number at random, and the process drawing the largest number is allowed to entry the critical region. The algorithm uses a read-modify-write primitive on a shared variable $V = (S, B, R)$ with three fields:

Shared variable: $V = (S, B, R)$, where

```
$S \in \{0,1\}$, initially 0                                  ** semaphore **
$B \in \{0,1,\ldots,\lceil\log n\rceil + 4\}$, initially 0    ** posted number **
$R \in \{0,2,\ldots,99\}$, initially *random*                 ** round number **
```

Code for i:

Local variables:

```
$B_i \in \{1,\ldots,\lceil\log n\rceil + 4\}$, initially 1    ** the chosen number of process $i$ **
$R_i \in \{0,1,\ldots,99\}$, initially $\perp$                ** the round number of lottery in which $i$ is
                                                                 currently participating **
```

Code:

```
while $V \neq (0, B_i, R_i)$ do                     ** i.e., while not winner or C is not available **
   if $(V.R \neq R_i)$ or $(V.B < B_i)$ then         ** i.e., not yet participated in lottery **
      $B_i \leftarrow random$                        ** random: truncated geometric **
      $V.B \leftarrow max(V.B, B_i)$
      $R_i \leftarrow V.R$
   unlock; lock;
$V \leftarrow (1, 0, random)$                        ** random: uniform **
unlock;                                              ** Try → Crit **

 ** Critical Region **

lock;                                                ** Crit → Exit **
$V.S \leftarrow 0$
$R_i \leftarrow \perp$
$B_i \leftarrow 1$
unlock;                                              ** Exit → Rem **

 ** Remainder Region **

lock;                                                ** Rem → Try **
```

Figure 1: Rabin's Randomized mutual exclusion algorithm.

1. $S \in \{0,1\}$ is used as a semaphore to ensure mutual exclusion as in simple mutual exclusion algorithms where a process enters the critical region only when the semaphore is set to 0, sets the semaphore to 1 upon entering the critical region, and resets the semaphore to 0 upon leaving;

2. $B \in \{0,\dots,b\}$ is used to post the largest number drawn by a process in the current lottery for entrance to the critical region; and

3. $R \in \{0,\dots,r\}$ is used to post a round number for the current round of competition for entry to the critical region.

In the algorithm b is chosen to be $\lceil \log n \rceil + 4$ and r to be a small constant such as 99.

The algorithm itself appears in Figure 1.

We introduced a few minor changes in the algorithm first published by Rabin. These changes are only semantic changes and do not alter in any way the algorithm described:

- Our code uses the lock/unlock commands that make explicit the atomic accesses to the shared read-modify-write variable. This allows for a less ambiguous interpretation of the code and for the economy of an **if** loop within the trying region.

- We do not make explicit in the code the changes of region (e.g., exit C) in the code: the region in which a process operates at a given point is implicitly defined by its position within its piece of code.

- In our algorithm the semaphore $V.S$ is set to 1 when the critical region is occupied.

- We reinitialize the local variables R_i and B_i in the exit region and not at the entrance of the critical region.

- We initialize the variable $V.R$ to a *random* value drawn from the same distribution as the one used at the entrance of the critical region, when the line of code $V \leftarrow (1, 0, \mathit{random})$ is performed. This prohibits the adversary from having more then a random knowledge of the state of the system at the first round.

We can characterize some general notion introduced in Section 3 in terms of the specifics of the code of figure 1. For instance, the set of processes participating in a round is the set of processes that go at least once through the **while** loop of their code during this round. Also, the transition Try → Crit corresponds to the atomic operation "lock; $V \leftarrow (1, 0, \mathit{random})$; unlock" of the code.

We first describe the essence of the algorithm. At the end of each round $k-1$, the winning process draws at random a new round number $V.R$. At the beginning of round k, the adversary has no knowledge of this number. (But, and this was overlooked in the original paper of Rabin, as the round proceeds, the adversary might get some partial knowledge of it!). Hence the adversary cannot influence "too much" the test "$V.R = R_i$?" performed by any process i it calls to participate in round k. (Due to our initialization of R this is true as well for round 1.) As a consequence, with "high" probability, the first time a process i participates in round k, the test "$V.R = R_i$?" returns false, at which point i chooses a new lottery random number B_i, and updates its round number: $R_i \leftarrow V.R$. Also, $V.B$ is always updated within a round to keep the highest value $\text{Max}_i B_i$ drawn so far in the round. The first process that checks that its lottery value is the highest obtained so far in the round (i.e., that $V.B = B_i$) at a point when the critical section is unoccupied (i.e., when $V.S = 0$) takes possession of the critical section. At this point the shared variable is reinitialized ($V.R$ is reset to random and $V.B$ is reset to 0) and a new round begins.

The algorithm has the following two features. First, any participating process i reinitializes its variable B_i at most once per round. Second, the process winning the competition takes at most two steps (and at least one) after the point f_k of the round at which the critical section becomes free. Equivalently, a process i that takes two steps after f_k and does not win the competition cannot hold the current maximal lottery value. (As i already took a step in round k it holds the current round number i.e., $R_i(k) = R(k)$. On the other hand the semaphore is set to 0 after f_k. If i held the highest lottery value it would pass all three tests in the code and enter the critical section.) We will take advantage of this last property in our constructions.

4.2 Notations Specific to Rabin's Algorithm

Let $\mathcal{A}$ be a fair adversary, $(\rho(k))_{k=1}^{\infty}$ an infinite sequence of numbers in $\{0, \ldots, 99\}$ and let $(\beta_i(k))_{k=1}^{\infty}$, $i = 1, \ldots, n$ be n infinite sequences of numbers in $\{1, \ldots, b\}$. We then define $\mathcal{E}(\rho, \beta_1, \ldots, \beta_n, \mathcal{A})$ to be the execution of the algorithm with adversary $\mathcal{A}$, in which the successive choices of round numbers are drawn from ρ and the successive choices of the B_i's are drawn from β_i. More specifically, we saw that a process i participating in a round k draws a new lottery number at most once during that round. If it does, we can assume that it uses $\beta_i(k)$ for that purpose. In the same way, the algorithm uses $\rho(k)$ as the round number for round k; for $k = 1$ this number is chosen in the initialization, while for $k \geq 2$ it is chosen at the very end of round $(k-1)$ by the process that wins round $k-1$ at the point at which it executes the code line $V \leftarrow (1, 0, \mathit{random})$.

In the spirit of Section 2 we let ω denote $(\rho, \beta_1, \ldots, \beta_n)$ and identify an execution $\mathcal{E}(\omega, \mathcal{A})$ with its defining parameters $\omega, \mathcal{A}$. In this section we will fix the adversary $\mathcal{A}$. Any ω then defines a unique execution, allowing us to consider the set of executions as being a probability space endowed with the image measure under $\omega \rightarrow \mathcal{E}(\omega, \mathcal{A})$. We will use this fact implicitly when estimating "the probability of a set of executions".

Even though the time parameterization, which is incremented whenever any process takes a step, allows for a complete parameterization of the executions, the less refined one, induced by the round numbers, is easier to handle and will suffice for our purpose.

Following the general conventions that we set out in Section 3, for any quantity X, we let $X(k)$ denote the value of X *just prior* to the last step of round k of the execution. This step "$V \leftarrow (1, 0, \mathit{random})$" is taken by the process that wins the lottery and takes possession of the critical region. It corresponds to a reinitialization of the shared variable in preparation for round $k+1$.

We will for instance consider the quantities held by a program variable:

- $R(k)$ is the round number used during round k. (It is set at the very end of round $k-1$.)
- $R_i(k)$ is the round number process i ends up with at the end of round k.

- $B_i(k)$ represents the lottery number process i ends up with at the end of round k; it is conceivable that process i does not set $B_i(k)$ to $\beta_i(k)$ (so that $B_i(k) = B_i(k-1)$), *even* if it participates in round k. This explains why we have chosen a different notation $\beta_i(k)$ to deal with the actual lottery numbers.

- $B(k)$ represents the lottery value of the process that ends up winning round k. Hence the content of the program variable $V.B$ grows, within round k, from the value 0 it holds at the beginning of the round, to the value $B(k)$ it assumes just prior to the reinitialization done of the end.

We also introduce and define some sets of executions that will appear in the course of the analysis.

- For each i and k define $\mathcal{N}_i(k) \overset{\text{def}}{=} \{i$ reinitializes its program variable B_i with a new value $\beta_i(k)$ during round $k\}$. ($\mathcal{N}$ stands for New-values.)

- For each k define $\mathcal{N}(k) \overset{\text{def}}{=} \{$All the processes j participating in round k reinitialize their program variables B_j with a new value $\beta_j(k)$ during round $k\}$.

- For each k define $\mathcal{U}(k) \overset{\text{def}}{=} \{$ There is exactly one process j participating in round k such that $B_j(k) = B(k)$ and $R_j(k) = R(k)\}$. ($\mathcal{U}(k)$ stands for Uniquemax.) $\mathcal{U}(k)$ is the set of executions in which at most one process ends up with the highest lottery value in round k. Recall also the three following general notations:

- $\mathcal{P}(k)$ denotes the set of participating processes in round k.

- For each k and $\mathcal{E}$, $\pi_k(\mathcal{E})$ denotes the k-round run compatible with $\mathcal{E}$.

- For each i and k, $W_i(k)$ denotes the set of executions in which process i enters the critical region at the end of round k.

We can relate some of the preceding events by

$$\mathcal{N}(k) = \bigcap_{i \in \mathcal{P}(k)} \mathcal{N}_i(k).$$

Note that the quantity $|\mathcal{P}(k)|$ is not completely under the control of the adversary. For instance, the adversary cannot ensure with certainty that at least 2 new processes enter the competition of round k if the semaphore S is set to 0: the first process j doing so could go right through into the critical region![2]

We now turn to the probabilistic aspects of the model, introduce the random inputs of the algorithm and discuss the probability induced on the space of executions.

[2] This happens exactly when the test "$V.R = R_j$ and $V.B = B_j$" is true. Recall that the adversary does not have knowledge of the local variables and hence cannot control the test.

4.3 The Random Inputs

Recall that we set $b = \lceil \log n \rceil + 4$. The sequence ρ is obtained as a sequence of iid (independent identically distributed) uniform random variables and the sequences $(\beta_i(k))_{k=1}^{\infty}$ are constructed as sequences of iid truncated geometric random variables:

$$\begin{aligned} \mathbf{P}[\rho(k) = l] &= \tfrac{1}{100}, \quad 0 \leq l \leq 99, \\ \mathbf{P}[\beta_i(k) = l] &= \begin{cases} \frac{1}{2^l} & \text{if } 1 \leq l < b, \\ \frac{1}{2^{l-1}} & \text{if } l = b. \end{cases} \end{aligned}$$

As discussed previously, for any *given* adversary, the product measures controlling ρ and the β_i's endow the set of executions with a probability measure. This is the probability space that we consider in the definition of probabilistic no-lockout. In this setting, the quantities $\mathcal{P}(k)$, $R(k)$, $B(k), R_i(k) \ldots$ can then be viewed as random variables, and the events $\mathcal{U}(k), \mathcal{N}(k), \mathcal{N}_i(k)$, $\{\pi_k = \rho\}$, $W_i(k) \ldots$ previously defined can be viewed as random events.

A difficulty though, which will give rise to the discussion of Sections 8 and 7, is that the adversary is not "given" independently of the random choices: it interacts with randomization in the sense that it actually sets its decisions based on its knowledge (of the randomized state of the system). As a consequence, the probability distribution on the set of executions depends on $\mathcal{A}$, which justifies our notation $\mathbf{P}_{\mathcal{A}}$. In terms of the code, this translates into the fact that the various (random) variables involved in the code are not independent. The $R(k), k = 1 \ldots$ are iid, since each $R(k)$ is chosen to equal $\rho(k)$ and the $\rho(k)$'s are iid. However, the random variables $B_i(k)$ do not have the same distribution as the $\beta_i(k)$. In particular the $B_i(k)$'s are not iid whereas the $\beta_i(k)$ are. Similarly, the variables $(B_i(k))_k^{\infty}$ and $(R_j(k))_k^{\infty}$ for arbitrary i and j's are not independent.

5 Rabin's Algorithm and the No-lockout Property

To begin, we show that the no-lockout property of Definition 3.4 does not hold.

5.1 A No-Lockout Property that does not hold

We will give a counterexample proving that the algorithm does not satisfy Property 3.4. For this we take n big enough so that $n > 100b + 1$ and set $k = 100b + 1$. We then consider an execution ϵ having the properties that, for all $(r, s) \in \{0, \ldots, 99\} \times \{1, \ldots, b\}$, there is a process j whose local variables R_j and B_j hold r and s respectively by the beginning of round k, i.e., such that $(R_j(k-1), B_j(k-1)) = (r, s)$.

An example of such an execution is an execution such that:

- In round 1, process 1 participates and loses with its private variables being $R_1(1) = 1$, $B_1(1) = 1$. Process 1 does not participate in rounds $2, \ldots, k-1$.

- In round 2, process 2 participates and loses with its private variables being $R_2(2) = 1$, $B_2(2) = 2$. Process 2 does not participate in rounds $3, \ldots, k-1$.

$$\vdots$$

- In round $100b$ process $100b$ participates and loses with its private variables being $R_{100b}(100b) = b$, $B_{100b}(100b) = b$.

Consider then an adversary $\mathcal{A}_0$ that behaves as follows in round k $(= 100b + 1)$. $\mathcal{A}_0$ gives one step to process $100b + 1$ whereas the critical region is still occupied. It then waits for the critical section to become free and proceeds at this point with the schedule $1, 2, \ldots, 100b+1$. But, whatever R_{100b+1} and B_{100b+1} are, there is one process among the processes $1, \ldots, 100b$ that holds the same values. This process takes then possession of the critical section so that $\mathbf{P}_{\mathcal{A}_0}\Big[W_{100b+1}(100b+1) \mid \pi'_{k-1} = \epsilon,\ 100b + 1 \in \mathcal{P}(k)\Big] = 0$!

5.2 Rabin's No-Lockout Statement

In his original paper [4], Rabin claimed that the algorithm verifies the property (*) given in Section 1.

One of the problem with such a statement is that it is not formally expressed. A few natural questions arise in the pursuit of its analysis:

- At which point t_k of the execution do we want to evaluate the probability of the (future) success of process i in round k? This has to be before round k ends: by the time round k ends, its outcome is known with probability one! But then, when and how is m defined? In effect, recall that m denotes $|\mathcal{P}(k)|$. And, as we emphasized in Sections 3 and 4.2, $|\mathcal{P}(k)|$ is finally set only by the *end* of round k!
- We saw in Section 3.3 that the worst-case adversary knows *implicitly* the set over which the conditioning is done in the statement of the no-lockout property, even though, by definition, the knowledge that it is effectively provided with is only the past run. Call $S(t_{k-1})$ the set of past executions that we want to condition on in the no-lockout property. (Note that we can a priori consider any set $S(t_{k-1})$ for which there is a run ρ and and an execution ϵ such that $\{\pi'_{k-1} = \epsilon\} \subseteq S(t_{k-1}) \subseteq \{\pi_{k-1} = \rho\}$.)

  The more precise (i.e., small) we can make $S(t_{k-1})$, the stronger the algorithm is: this means in effect that in spite of an accrued knowledge of the state of the system by the adversary, the algorithm is still able to preserve the no-lockout property.

The previous discussion leads us to propose the following solutions:

- We select t_k to be the beginning of round k, so that, in the no-lockout property, the probability that i takes possession of the critical region by the end of round k is estimated at the *beginning* of round k.

- We translate formally the fact that we restrict ourselves to the case of m participating processes, by conditioning the probability on the event $\{|\mathcal{P}(k)| = m\}$.
- About the choice of the set $S(t_{k-1})$: we will consider the weakest condition where $S(t_{k-1})$ is taken to be $\{\pi_{k-1} = \rho\}$.

Bringing together these choices, the formal transcription of the no-lockout property sought by Rabin is the one of Definition 3.3.

5.3 An Introduction to Rabin's Proof and its Problems

We know embark in the program of trying to transcribe the informal proof given by Rabin in his paper and of filling the missing gaps. A *correct* and formal way to proceed following the line of Rabin's argument is to consider the inclusion

$$\{W_i(k),\ i \in \mathcal{P}(k)\} \supseteq \{i \in \mathcal{P}(k),\ \mathcal{N}_i(k),\ \mathcal{U}(k),\ B_i(k) = B(k)\}$$

and the corresponding inequality:

$$\begin{aligned}
&\mathbf{P}_{\mathcal{A}}[W_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m] \geq \\
&\quad \mathbf{P}_{\mathcal{A}}[\mathcal{N}_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m] \\
&\quad \mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}_i(k)] \\
&\quad \mathbf{P}_{\mathcal{A}}[B_i(k) = B(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}_i(k),\ \mathcal{U}(k)].
\end{aligned} \tag{1}$$

At this point Rabin uses the three lower bounds

$$\mathbf{P}_{\mathcal{A}}[\mathcal{N}_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m] \ \geq\ .99, \tag{2}$$

$$\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}_i(k)] \sim 2/3, \tag{3}$$

$$\mathbf{P}_{\mathcal{A}}[B_i(k) = B(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}_i(k),\ \mathcal{U}(k)] \geq 1/m. \tag{4}$$

In his paper, Rabin gives only some very informal justification for Equations (2) and (4) and gives a mathematical property about *independent* variables that is related to Equation (3). But this result is *not* the one of Equation (3) which deals with *dependent* program variables. Upon careful analysis, we find that all three inequalities are incorrect for different reasons.

The major difficulty arising in the analysis of the no-lockout property is the interaction in between the probabilistic aspect of the algorithm and the adversary. On the one hand one of the two roles of randomness in the algorithm is to limit the knowledge of the current state held by the adversary. On the other hand, we argued in Section 3.3 that probabilistic conditioning enhances the knowledge held by the adversary: conditioning reduces the random space. (This is the nature of conditioning!) An adversary can take advantage of the reduction of randomness incumbent to probabilistic conditioning in at least two ways.

It can gather knowledge as the execution unfolds, wait for a "lucky" event within a round and then act against the algorithm. This problem is addressed in Section 8 and is a semi refutation of Equation (2) above.

But the most striking counterexamples that we bring stem from the fact the adversary can use directly the knowledge provided through the conditioning of the no-lockout statement. The events involved in this conditioning are of two very different nature:

- The event $\{|\mathcal{P}(k)| = m\}$ is *posterior* to the point t_k at which the probability is evaluated. We show in Theorem 6.1 that the adversary can use this fact in a simple way to lock out any process during any round. This fact is in contradiction with Equation (4) above. Furthermore the power derived by the adversary from the knowledge of $\{|\mathcal{P}(k)| = m\}$ is so significant that it does not need to use its knowledge of the current run!

- The event $\{\pi_{k-1} = \rho\}$ is *prior* to the point t_k. If the random variable was acting as a real eraser of the past, the adversary could not be able to use this fact to its advantage. But in Theorem 7.1 we will prove that the adversary can prepare before round k a host of processes holding high lottery values, and use them to lockout a linear fraction of the processes. This result constitute a very strong refutation of Equation (3): it actually demonstrate that $\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}_i(k)] = o(1/n)$.

We will discuss further this problem in Section 6.2.

Rabin actually proved a result different from the one of Equation (3). His result (his Lemma 3.3) can be restated in our language as being:

$$\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}(k)] \sim 2/3.$$

Conditioned on $\mathcal{N}(k)$, $\mathcal{U}(k)$ is independent of $\{\pi_{k-1} = \rho\}$. Furthermore, as all of the processes draw new values, they all assume a symmetric role, so that, conditioned on $\mathcal{N}(k)$, the probability of $\mathcal{U}(k)$ does not depend on $i \in \mathcal{P}(k)$ (for $m \geq 2$).

Hence the previous equation can be reduced to:

$$\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid\ |\mathcal{P}(k)| = m,\ \mathcal{N}(k)] \sim 2/3, \tag{5}$$

in which Rabin's result can be more easily recognized.

The problem we are then facing is to reexpress Inequality (1) so as to introduce

$$\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}(k)],$$

instead of

$$\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m,\ \mathcal{N}_i(k)].$$

We will address the issue in Section 9 and show how and where it leads to problems.

6 Refutation of Equation 4

The result of this section states that the strong $\Omega(1/m)$ result claimed by Rabin is incorrect.

6.1 The Refutation

Theorem 6.1 *The algorithm does not have the strong no-lockout property of Definition (3.5) (and hence of Definition 3.3). Indeed there is an adversary $\mathcal{A}_1$ such that, for any $m \le n-1$,* $\mathbf{P}_{\mathcal{A}_1}\Big[W_1(1) \mid 1 \in \mathcal{P}(1),\ |\mathcal{P}(1)| = m\Big] = 0$.

As we already remarked, the worst case adversary acts as if it knows the events on which conditioning is done. Knowing beforehand that the total number of participating processes in the round is m allows the adversary to design a schedule where processes take steps in turn, where process 1 begins and where process m takes possession of the critical section.

Specifically, consider the adversary $\mathcal{A}_1$ whose $2n+1$ first allocations of steps are defined in Figure 2.

$\mathcal{A}_1$ is described as a function from the set of finite runs to the set of processes. Recall that a run is a sequence of terms (i_t, old_t, new_t). $\mathcal{A}_1$ does not use its knowledge about ρ and can be more simply described as a function $\mathcal{A}_1[i_1, i_2, \ldots, i_t] = i_{t+1}$. At the beginning of the execution all processes are in their initial configuration as described in the code of Figure 1. In particular all the variables B_i are set to 0.

$$
\begin{aligned}
&\mathcal{A}_1[\emptyset] = 1\\
&\mathcal{A}_1[1] = 2\\
&\mathcal{A}_1[12] = 2\\
&\mathcal{A}_1[122] = 3\\
&\mathcal{A}_1[1223] = 3\\
&\ldots\\
&\mathcal{A}_1[122\ldots ii] = i+1\\
&\mathcal{A}_1[122\ldots ii(i+1)] = i+1\\
&\ldots\\
&\mathcal{A}_1[122\ldots nn] = 1.
\end{aligned}
$$

Figure 2: Adversary $\mathcal{A}_1$ "locking out" process 1 in round 1.

Lemma 6.2 *The allocation of steps described in Figure 2 brings round* 1 *to its end.*

Proof: The program variable S is 0 at the beginning of round 1: the critical section is free. Looking for a contradiction, assume that round 1 is not over after the $2n+1$ steps described in Figure 2. Recall that $B(1)$ is the maximal value of the $B_i(1)$'s drawn in round 1:

$$B(1) = \operatorname*{Max}_{i \in \mathcal{P}(1)} B_i(1).$$

Consider the first process j drawing $B(1)$ when the adversary of Figure 2 is implemented. While taking its second step, process j verifies that $V = (0, B_j(1), R_j(1))$ and hence takes possession of the critical section; but this finishes round 1. ■

Lemma 6.3 *Let $B_1(0)$, $B(0)$ denote the initial values held by the program variables B_1 and B. With the adversary $\mathcal{A}_1$ described in Figure 2, process* 1 *can take possession of the critical region during round* 1 *only if $B_1(1)$ is the maximal value among all the n values $B_i(1)$.*

Proof: 1 does not take control of the critical region at its first step, because $B(0) < B_1(0)$ at this point. Hence it has then to wait for its second step within round 1. But, due to $\mathcal{A}_1$, this second step will come (if it comes) only when all the other processes will have been given two steps. Adopting the idea of the proof of the previous claim, we see that 1 gets then control of the critical region only if $B_1(1)$ is maximal among all the $B_i(1)$ (and is actually the only such maximal value). ■

Lemma 6.4 *With the adversary described in Figure 2,* 1 *cannot take possession of the critical section if $|\mathcal{P}(1)| \leq n - 1$.*

Proof: As we just saw, 1 can win round k only if it wins at its second step, in which case $|\mathcal{P}(1)|$ is n. ■

Proof of Theorem 6.1: For $m \leq n - 1$, lemma 6.4 tells us that $\{|\mathcal{P}(1)| = m\} \cap W_1(1) = \emptyset$. On the other hand $\mathbf{P}_{\mathcal{A}_1}[|\mathcal{P}(1)| = m] \neq 0$. Hence

$$\begin{aligned} \mathbf{P}_{\mathcal{A}_1}\Big[W_1(1) \mid 1 \in \mathcal{P}_1,\ |\mathcal{P}(1)| = m\Big] &= \frac{\mathbf{P}_{\mathcal{A}_1}\Big[\{|\mathcal{P}(1)| = m\} \cap W_1(1) \mid 1 \in \mathcal{P}(1)\Big]}{\mathbf{P}_{\mathcal{A}_1}[|\mathcal{P}(1)| = m]} \\ &= 0. \end{aligned}$$

■

6.2 A Priori versus A Posteriori Conditioning

The counterexample of Section 6.1 illustrates the general principle that we noticed previously: conditioning on an event $\mathcal{S}$ amounts to provide the adversary with the knowledge of $\mathcal{S}$. But the event $\mathcal{S} = \{|\mathcal{P}(k)| = m\}$ is different in a significant way from the other ones (e.g., $\pi_{k-1} = \rho$, $\pi'_{k-1} = \epsilon$) which appear in the various definitions of no-lockout.

In effect, these events are events that are described by properties of the execution at points which are *prior* to the point t_k of the execution at which the probability is evaluated. Conditioning on them provides the adversary only with an *a posteriori* knowledge of the execution (the knowledge of the *past* run). As a consequence, the adversary is able to act against the algorithm only in specific rounds k after specific runs.

But $\{|\mathcal{P}(k)| = m\}$ is an event which is described by a property of the execution at the *end* of round k. This point is *posterior* to the point t_k at which the probability is evaluated, so that conditioning on $\{|\mathcal{P}(k)| = m\}$ amounts to provide the adversary with the *a priori* knowledge that the number of participating processes will be m, *whatever* it does.

Furthermore, as stated in Theorem 6.1, the adversary is even strong enough to counter the Property 3.5 (and not only the weaker Property 3.3). Even worse: the adversary is so strong that it is able to do so at any round k, deciding the schedule ahead of time and using its knowledge of the run only in a marginal way. In effect we can adapt the example of Figure 2 for the case of k arbitrary: in the first $k-1$ rounds $\mathcal{A}_1$ gives steps in any arbitrary way. It then waits for the process in control of the critical region to leave (This is the only point where the adversary uses its knowledge about the run of the execution. Note also that it does so after the point t_k at which the probability is evaluated.), and then proceeds with the $2n+1$ allocation of steps defined in Figure 2. The same argument as before shows that, with any such adversary $\mathcal{A}_1$,

$$\mathbf{P}_{\mathcal{A}_1}\Big[W_1(k) \mid 1 \in \mathcal{P}(k),\ |\mathcal{P}(k)| = m\Big] = 0.$$

We just went over a subtle point about the interplay of the probabilistic conditioning and the power of the adversary. This point is most apparent in the property of Definition 3.5. Definition 3.5 expresses formally that the point t_k of the execution at which the probability is evaluated is the very beginning of the execution (and that, evaluated from this point, the probability of i succeeding at the end of round k is c/m). Hence, in this case, no knowledge of the type $\pi_{k-1} = \rho$ is assumed. Nevertheless this last fact does not preclude the adversary from getting this knowledge in an on-line fashion as the execution unfolds.

6.3 Towards Fixing the Algorithm

The previous counterexample is based on the fact that the code of the trying section has an "unlock;lock", and that a process i which draws a lottery number B_i such that $B_i = V.B$ has to be given a second step before taking possession of the critical section (assuming that, by the time it receives this step, the round is not already over and that i still holds the highest lottery number). This fact provides the adversary with an edge allowing it to force a given participating process i to compete against *all* other processes before being able to take possession of the critical region. We will say that the adversary has the total-tournament feature with respect to i.

We might think that the algorithm just needs the following simple fix to satisfy the property of Definition (3.5): we remove the cut in the thread of executions and ensure that a process i takes immediately possession of the critical section after having drawn the highest lottery number so far in the round. This change affects only the code of the trying region and is described in Figure 3.

But this modification does not work. In effect, we can then construct the following similar adversary $\mathcal{A}_2$. $\mathcal{A}_2$ lets only process n take steps (and therefore win) in round 1. In round 2 it gives one step to process 1 while the critical section is still under control of process n, waits for the critical region to be freed, and then gives in turn one step to processes $2, 3, \ldots, n, 1$. This adversary still holds in round 2 the total-tournament feature with respect to process 1. The same argument as the one used with $\mathcal{A}_1$ shows that the strong version of Rabin's Theorem does not hold with the modified code $\mathcal{C}_1$ of Figure 3.

To remove the total-tournament feature of the adversary and to obtain the strong version of the no-lockout property we have to ensure that a process cannot be given a step when the critical section is under the control of another process. We can achieve this by modifying the code as indicated in Figure 4.

In the code $\mathcal{C}_2$ the shared variable is locked whenever and as long as a process is in the critical section. At this point, the algorithm does satisfy the strong version of the no-lockout property.

```
    while $V \neq (0, B_i, R_i)$ do
        if $(V.R \neq R_i)$ or $(V.B < B_i)$ then
            $B_i \leftarrow random$
            $V.B \leftarrow max(V.B, B_i)$
            $R_i \leftarrow V.R$
        if $(V.B = B_i)$ and $(V.S = 0)$ then go to (A)
        unlock; lock
(A) $V \leftarrow (1, 0, random)$
    unlock;
```

Figure 3: Modified code $\mathcal{C}_1$ for the trying region of the code for i

```
    while $V \neq (0, B_i, R_i)$ do
        if $(V.R \neq R_i)$ or $(V.B < B_i)$ then
            $B_i \leftarrow random$
            $V.B \leftarrow max(V.B, B_i)$
            $R_i \leftarrow V.R$
        if $(V.B = B_i)$ then go to (A)
        unlock; lock
(A) $V \leftarrow (1, 0, random)$

  ** Critical Region **

    $V.S \leftarrow 0$
    $R_i \leftarrow 0$
    $B_i \leftarrow 0$
    unlock;

  ** Remainder Region **

    lock;
```

Figure 4: Modified code $\mathcal{C}_2$ for i

Unfortunately it is by now meaningless: $\mathcal{P}(k)$ is always one. The first process participating in round k takes possession of the critical section!

7 The Algorithm does not verify the $1/n$ No-Lockout Property!

This section stems from a refutation of the fact claimed by Rabin "that with probability 2/3, only one process wins the lottery". (This statement corresponds to Inequality 3.) The problem that our refutation uncovers is actually so important that it gives to the weak adversary of Definition 3.2 the possibility of locking out with *exponential* probability any process at will. We are very far now from the strong $1/m$ property of Definition 3.3!

7.1 The Refutation

Theorem 7.1 *The algorithm does not verify the Weak, Run-knowing No-Lockout property.*[3]

This theorem is a consequence of the next one:

Theorem 7.2 *$\exists c < 1$, for n big enough (e.g. $n \geq 20$), $\exists$ adversary $\mathcal{A}$, $\exists$ a round ρ such that:*

$$\mathbf{P}_{\mathcal{A}}[W_1(k) \mid \pi_{k-1} = \rho, 1 \in \mathcal{P}(k)] \leq e^{-32} + (1 - e^{-32})c^n.$$

In the previous theorem, $k - 1$ is implicitly assumed to be the number of rounds in ρ.

Proof: We will define an adversary that, during a preparation phase ρ, selects and stores some "strong" processes. As soon as these $n/2$ processes are ready, phase k begins and the adversary is able to give a turn and lock out with high probability process 1 (say).

We begin with a convenient definition.

Definition 7.1 Let l be a round. Assume that, during round l, the adversary elects to adopt the following strategy. It first waits for the critical section to become free, then gives one step to process i_0 and then *two* steps (in any order) to k other processes. Assume that at this point the critical section is still available (so that round l is not over). We then say that process i_0 is a *k-survivor* (at round l).

We will say in the sequel that the adversary "selects a k-survivor". By this we will mean that the adversary selects any process i_0 and then adopts the strategy described in Definition 7.1. Of course the adversary has no control on the fact that the critical section is still free after having given turns to all k processes. Hence, to select a k-survivor, the adversary will actually repeat the same schedule along possibly many rounds until the situation described in Definition 7.1 concretizes. At this point, $\mathcal{A}$ stores process i_0, i.e., does not give any step to process i_0 until round k begins.

In the preparation phase $\mathcal{A}$ selects strong processes in the following way.

- It first selects $6/20n$ $2^{\log_2 n - 1}$-survivors. (Note that $6/20n + n/2 < n$, so that this selection is indeed possible).

[3] This property is defined in Section 3.3

- It then selects $n/20$ $2^{\log_2 n-2}$-survivors.

$\vdots$

- It then selects $n/20$ $2^{\log_2 n-5}$-survivors.

At this point $\mathcal{A}$ selected a total of $n/2$ processes that we label $n/2+1, \ldots, n$ for convenience. These $n/2$ processes are partitioned among ten sets $S_{-5}, S_{-4}, \ldots, S_4$ characterized by the three properties:

- $\forall l = -5, \ldots, 4, |S_l| = n/20$,
- $\forall l = -2, -3, -4, -5,\ |S_l| \stackrel{\text{def}}{=} \{2^{\log_2 n+l}\text{-survivors}\}$,
- $\{2^{\log_2 n-1}\text{-survivors}\} \stackrel{\text{def}}{=} S_{-1} \cup S_0 \cup \ldots \cup S_4$.

As soon as this is over, the adversary has in store all the processes it needs and round k begins. $\mathcal{A}$ then adopts the strategy of Figure 5.

- While Crit is *still occupied*, $\mathcal{A}$ gives one step to processes $1, 2, \ldots, n/2$.
- $\mathcal{A}$ then waits for the transition $Crit \rightarrow Exit$ to occur.
- $\mathcal{A}$ then gives two steps (in any order) to all processes in $S_{-5} \cup S_{-4} \cup \ldots \cup S_4$.

Figure 5: Strategy used in phase k locking out $n/2$ processes.

We will use in the sequel the following convenient notation:

Definition 7.2 For any sequence $(a_i)_{i\in\mathcal{N}}$ we denote $Max_k a_i \stackrel{\text{def}}{=} Max\{a_1, a_2, \ldots, a_k\}$.

Claim 7.3

$$\begin{array}{llll} \forall l = -1, \ldots, -5, & \forall i \in S_l, & \mathbf{P}_{\mathcal{A}}[B_i = log_2 n + l] & \geq\ 0.25 \\ \forall l = 0, \ldots, 4, & \forall i \in S_l, & \mathbf{P}_{\mathcal{A}}[B_i = log_2 n + l] & \geq\ 0.05\ . \end{array}$$

Proof: (of Claim 7.3.)

In the preparation phase ρ, whenever $\mathcal{A}$ goes through the process of selecting a k-survivor i_0, it picks at random k processes that i_0 has to withstand. By this we mean that the lottery number of i_0 is bigger then the one of these k processes:

$$B_{i_0} \leq \text{Max}_k B_i \sim \text{Max}_k \beta_i\ .$$

The result is then a consequence of Corollary 7.8.

■

We know turn to the analysis of round k. When the critical section becomes free, due to the strategy of the adversary, the variable $V.B$ holds the maximum of the $n/2$ variables $1, 2, \ldots, n/2$. By Corollary 7.6 we then deduce that

$$\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_{n/2}B_i \geq \log_2 n - 5] \sim (1 - e^{-32}).$$

Let $d' \overset{\text{def}}{=} \mathrm{Inf}(0.25, 0.05) = 0.05$ and $d \overset{\text{def}}{=} d'/100$.

Then the winner of the competition at round k is one of the $n/2$ proceses $1, \ldots, n/2$ only if:

their maximum lottery number is less then $\log_2 n - 5$

or if their maximum lotteryy number is some number $l \geq \log_2 n - 5$
and if the $n/20$ processes in S_l do not pass the test "$(V.R = R_i)$ and $(V.B = B_i)$?".

But a given process in S_l does not pass this test with probability at most $(1 - d'/100)$. From this we deduce the inequality

$$\mathbf{P}_{\mathcal{A}}[W_1(k) \cup W_2(k) \cup \ldots \cup W_{n/2}(k) \mid \pi_{k-1} = \rho] \leq e^{-32} + (1 - e^{-32})(1 - d)^{n/20}.$$

■

This result means that there is an adversary that can lockout, with probability exponentially close to 1, an arbitrary set of $n/2$ processes during *some* round. With a slight improvement we can actually derive an adversary that will succeed in *always* locking out a given set S of (say) $n/100$ processes: we just need to remark that the adversary can do without this set S during the preparation phase ρ. The adversary would then alternate preparation phases $\rho_1, \rho_2, \ldots$ with rounds $k_1, k_2, \ldots$ The set S of processes would be given steps only during rounds $k_1, k_2, \ldots$ and would be locked out at each time with probability exponentially close to 1.

7.2 What happens if the domain of the shared variable is increased?

In view of our counterexamples we might think that allowing an increase in the size of the shared variable might bring a solution. For instance if b is set to be 2 $\log_2 n$ instead of $\log_2 n + 4$, then $\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k \beta_i = log_2 k + l]$ cannot be given a good lower bound for l big. This is to be put into perspective with the second inequality of Corollary 7.8 and of Claim 7.3. In a similar way it is natural to wonder whether for instance taking $\log_2 n$ possible round number values instead of 100 might help. The answer is: no!

Theorem 7.4 *Assume that the set of possible round numbers used by the algorithm has size r and that the set of possible lottery numbers has size b ($log_2 n + 4 \leq b \leq n$). Then there exists a constant d' ($d' = 0.05$), an adversary $\mathcal{A}$, and a run ρ such that:*

$$\mathbf{P}_{\mathcal{A}}[W_1(k) \mid \pi_{k-1} = \rho,\ 1 \in \mathcal{P}(k)] \leq e^{-32} + (1 - e^{-32})(1 - d'/r)^{n/20} + 2\frac{r}{n^2}.$$

The previous result can be simplified into:

$$\mathbf{P}_{\mathcal{A}}[W_1(k) \mid \pi_{k-1} = \rho,\ 1 \in \mathcal{P}(k)] \leq e^{-32} + e^{-c_1 n/r} + c_2\frac{r}{n^2},$$

where c_1 and c_2 are some positive constants. This shows that, if r is sub-linear (and this is *exactly* the theoretical result that the algorithm was trying to achieve!), there is an adversary capable of bringing the probability to a $o(1/n)$. Furthermore, the remark made at the end of the previous section applies here also: a set of processes of linear size can be systematically locked out with probability arbitrary close to 1.

Proof: As in Section 7.1, for $l = -5, \ldots, -2$, the adversary prepares a set S_l of $2^{\log_2 n+l}$-survivors, each of size $n/20$, and a set S_{-1} of $2^{\log_2 n-1}$-survivors; the size of S_{-1} is $6/20n$. (We can as before think of this set as being partitioned into six different sets.) We let η stand for 6/20 in the sequel.

For simplicity, assume that the adversary wants to lockout process 1. Let p_m denote the probability that process 1 holds m as its lottery value after having taken a step in round k. For any process j in S_{-1} let also q_m denote the probability that process j holds m as its lottery value at the end of the preparation phase ρ.

The same reasoning as in Section 7.1 then leads to the inequality:

$$\mathbf{P}_{\mathcal{A}}[W_1(k) \mid \pi_{k-1} = \rho,\ 1 \in \mathcal{P}(k)] \leq e^{-32} + (1 - e^{-32})(1 - d'/r)^{n/20} + \sum_{m \geq \log_2 n+5} p_m(1 - \frac{q_m}{r})^{\eta n}.$$

Write $m = \log_2 n + x - 1 = \log_2(n/2) + x$. Then, as is seen in the proof of Corollary 7.8, $q_m = e^{-2^{1-\zeta}} 2^{1-\zeta}$ for some $\zeta \in (x, x+1)$. For $m \geq \log_2 n + 5$, x is at least 6 and hence $q_m \sim 2^{1-\zeta} \geq 2^{1-x}$. On the other hand $p_m \sim 2^{-m} = 2^{-x}/n$.

Define $\psi(x) \overset{\text{def}}{=} e^{-2^{1-x}\eta n/r}$ so that $\psi'(x) = e^{-2^{1-x}\eta n/r} 2^{1-x}\eta n/r$. Then:

$$\begin{aligned}
\sum_{m \geq \log_2 +5} p_m(1 - \frac{q_m}{r})^{\eta n} &\leq 1/n \sum_{x \geq 6} 2^{-x}(1 - \frac{2^{1-x}}{r})^{\eta n} \\
&\leq 1/n \sum_{x \geq 6} 2^{-x} e^{-(\frac{2^{1-x}}{r}\eta n)} \\
&= 1/(2n) \sum_{x \geq 6} 2^{1-x} e^{-(\frac{2^{1-x}}{r}\eta n)} \\
&= \frac{r}{2\eta n^2} \sum_{x \geq 6} \psi'(x) \\
&\leq \frac{r}{2\eta n^2} \int_5^\infty \psi'(x) dx \\
&= \frac{r}{2\eta n^2} [\psi]_5^\infty \\
&= \frac{r}{2\eta n^2} [1 - e^{-2^5 \eta n/r}] \\
&\leq \frac{r}{2\eta n^2}.
\end{aligned}$$

7.3 Mathematical Considerations

Consider a sequence of iid random variables $\beta_1, \beta_2, \ldots$, whose law is given by:

$$\mathbf{P}[\beta_i = l] \; = \frac{1}{2^{l-\epsilon_b(l)}},$$

where $\epsilon_b(l) = 1$ if $l = b$ and 0 otherwise.

Theorem 7.5

$$\mathbf{P}_{\mathcal{A}}[Max_k \beta_i \geq log_2 k + x] \sim 1 - e^{-2^{1-x}}.^4$$

Proof: Recall that, by definition, for all $j \leq b$, $\mathbf{P}_{\mathcal{A}}[\beta_i \geq j] = (1/2)^{j-1}$ so that $\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k \beta_i < j] = (1 - 2^{1-j})^k$. Setting $j = \log_2 k + x$ gives:

$$\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k \beta_i < \log_2 k + x] = (1 - \frac{2^{1-x}}{k})^k \sim e^{-2^{1-x}}.$$

■

The approximation in the previous theorem is actually *very* tight when k is big and x bigger then a given negative constant. This is the case in our construction where we chose $k \sim n/2$ and $x \geq -5$. As an illustration of the theorem we deduce the two following results.

Corollary 7.6

$$\mathbf{P}_{\mathcal{A}}[Max_k \beta_i < log_2 k - 4] \leq e^{-32}.$$

Proof: Immediate.

■

Corollary 7.7

$$\mathbf{P}_{\mathcal{A}}[Max_k \beta_i \geq log_2 k + 8] \leq 0.01 \; .$$

Proof: $1 - e^{-2^{1-8}} \sim 2^{-7} < 0.01$. ■

Theorem 7.5 implies that the maximum of k random variables β_i is concentrated tightly around $\log_2 k$: Corollary 7.6 shows that the maximum is with overwhelming probability at least as big as $\log_2 k$, whereas Corollary 7.7 shows that with probability 99% this maximum is at most $\log_2 k + 7$.

Corollary 7.8

$$\begin{array}{rl} & \mathbf{P}_{\mathcal{A}}[Max_k \beta_i = log_2 k] \geq 0.25 \\ \forall l = 1, \ldots, 4 & \mathbf{P}_{\mathcal{A}}[Max_k \beta_i = log_2 k + l] \geq 0.05 \; . \end{array}$$

[4] See Definition 7.2 for the definition of Max_k

Proof: Let $\phi(x) \overset{\text{def}}{=} 1 - e^{-2^{1-x}}$. Then $\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k\beta_i = log_2k + x] = \phi(x) - \phi(x+1)$. This is equal to $-\phi'(\zeta) = e^{-2^{1-\zeta}}2^{1-\zeta}$ for some $\zeta \in (x, x+1)$. In particular $\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k\beta_i = log_2k] = e^{-2^{1-\zeta}}2^{1-\zeta}$ for some $\zeta \in (0,1)$. We check immediately that ϕ'' is negative on $(-\infty, 1)$ and and positive on $(1, \infty)$. This allows us to write that

$$\mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k\beta_i = log_2k] \geq -\phi'(0) \geq 0.25 \,.$$

The same argument gives also that

$$\begin{aligned} \forall l = 1, \ldots, 4,\ \mathbf{P}_{\mathcal{A}}[\mathrm{Max}_k\beta_i = log_2k + l] &\geq \phi'(-5) \sim e^{-2^{-4}}2^{-4} \\ &\geq 1/20. \end{aligned}$$

■

8 Refutation of Equation 2

In his proof Rabin argued that the randomization of the round number was making it impossible to derive any knowledge about the private round numbers. He specifically claimed that: "with probability 0.99 we have that $R_i \neq R$".

We disprove that fact in this section. Our refutation is based on the fact that the adversary is able to use its knowledge of the run of the *on-going* execution during round k, and deduce some information about the current round number.

Assume that ρ indicates that processes $1, 2, 3, 4$ participated *only* in round l before round k, and that process 5 never participated before round k. More precisely, assume that during round l the following pattern happened: $\mathcal{A}$ waited for the critical region to become free, then allocated one step in turn to processes $1, 2, 2, 3, 3, 4, 4$; at this point 4 entered the critical region. Assume also that the current round k is under way and that the partial run ρ' at round k indicates that the critical region became free before any competing process was given a step, and that the adversary then allocated one step in turn to processes $5, 3, 3$, and that, after 3 took its last step, the critical section was still free. I claim that at this point:

"$R_2 \neq R$ with probability strictly less then 0.99".

Call ρ' the partial run of round k indicated above. Then the claim can be reformulated more precisely as:

Claim 8.1

$$\mathbf{P}_{\mathcal{A}}[R(k) \neq R_2(k-1) \mid \rho,\ \rho'] < .99$$

Proof: Bayes' rule allows us to write:

$$\mathbf{P}_{\mathcal{A}}[R(k) \neq R_2(k-1) \mid \rho,\ \rho'] = \frac{\mathbf{P}_{\mathcal{A}}[R(k) \neq R_2(k-1) \mid \rho]\ \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) \neq R_2(k-1)]}{\mathbf{P}_{\mathcal{A}}[\rho' \mid \rho]}. \tag{6}$$

In the numerator, the first term $\mathbf{P}_{\mathcal{A}}[R(k) \neq R_2(k-1) \mid \rho]$ is equal to 0.99 because $R(k)$ is uniformly distributed and independent from $R_2(k-1)$ and ρ. We will use this fact another time while expressing the value of $\mathbf{P}_{\mathcal{A}}[\rho' \mid \rho]$:

$$\begin{aligned} \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho] &= \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) \neq R_2(k-1)]\ \mathbf{P}_{\mathcal{A}}[R(k) \neq R_2(k-1) \mid \rho] \\ &\quad + \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) = R_2(k-1)]\ \mathbf{P}_{\mathcal{A}}[R(k) = R_2(k-1) \mid \rho] \\ &= 0.99\ \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) \neq R_2(k-1)] + 0.01\ \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) = R_2(k-1)]. \end{aligned} \tag{7}$$

By assumption l is the last round before round k where processes $1, 2, 3$ and 4 last participated. Hence $R_1(k-1) = R_2(k-1) = R_3(k-1)$.

- Consider first the case when $R(k) \neq R_2(k-1)$. Then process 3 gets a yes answer when going through the test "$(V.R \neq R_3)$ **or** $(V.B < B_3)$", and consequently chooses a new value $B_3(k) = \beta_3(k)$. Hence

$$\mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) \neq R_2(k-1)] = \mathbf{P}[\beta_3(k) < \beta_5(k)]. \tag{8}$$

- Consider now the case $R(k) = R_2(k-1)$ so that we also have $R(k) = R_3(k-1)$. By hypothesis, process 5 never participated in the computation before round k and hence draws a new number $B_5(k) = \beta_5(k)$. Hence

$$\mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) = R_2(k-1)] = \mathbf{P}_{\mathcal{A}}[B_3(k) < \beta_5(k) \mid \rho, R(k) = R_2(k-1)]. \tag{9}$$

As processes $1, \ldots, 4$ participated *only* in round l up to round k, the knowledge provided by ρ about process 3 is (exactly) that, in round l, process 3 lost to process 1 along with process 2, and that process 1 lost in turn to process 4, i.e., that $\beta_3(l) < \beta_1(l)$, $\beta_2(l) < \beta_1(l)$ and $\beta_1(l) < \beta_4(l)$. For the sake of notational simplicity, we let X denote for the rest of this paragraph a random variable whose law is the law of $\beta_1(l)$ conditioned on $\{\beta_1(l) > \mathrm{Max}\{\beta_2(l), \beta_3(l)\}, \beta_1(l) < \beta_4(l)\}$. This means that, for instance,

$$\forall x \in \mathbf{R},\ \mathbf{P}[X \geq x] = \mathbf{P}\Big[\beta_1(l) \geq x \mid \beta_1(l) > \mathrm{Max}\{\beta_2(l), \beta_3(l)\},\ \beta_1(l) < \beta_4(l)\Big].$$

When 3 takes its first step within round k, $V.B$ holds the value $\beta_5(k)$. As a consequence, 3 chooses a new value when and exactly when $B_3(k-1)(= \beta_3(l))$ is strictly bigger then $\beta_5(k)$. (Recall that the case $\beta_3(l) = \beta_5(k)$ would lead 3 to take possession of the critical section at its first step, and that the case $\beta_3(l) = \beta_5(k)$ would lead 3 to keep its "old" lottery value $B_3(k-1)$.) From this we deduce that:

$$\begin{aligned} \mathbf{P}_{\mathcal{A}}[B_3(k) < \beta_5(k) \mid \ \rho, R(k) = R_2(k-1)] &= \mathbf{P}[\beta_3(l) < \beta_5(k) \mid \ \beta_3(l) < X] \\ &\quad + \mathbf{P}[\beta_3(l) > \beta_5(k),\ \beta_3(k) < \beta_5(k) \mid \ \beta_3(l) < X]. \end{aligned} \tag{10}$$

Using Lemma 10.1 we derive that:

$$\mathbf{P}[\beta_3(l) < \beta_5(k) \mid \ \beta_3(l) < X] \geq \mathbf{P}[\beta_3(l) < \beta_5(k)].$$

On the other hand $\mathbf{P}[\beta_3(l) < \beta_5(k)] = \mathbf{P}[\beta_3(k) < \beta_5(k)]$ because all the random variables $\beta_i(j), i = 1, \ldots, n,\ j \geq 1$ are iid. Taking into account the fact that the last term of equation 10 is non zero, we have then established that:

$$\mathbf{P}_{\mathcal{A}}[B_3(k) < \beta_5(k) \mid\ \rho, R(k) = R_2(k-1)] > \mathbf{P}[\beta_3(k) < \beta_5(k)]. \tag{11}$$

Using Equations 8, 9 and 11 we deduce from this that

$$\mathbf{P}_{\mathcal{A}}[\rho' \mid \rho, R(k) = R_2(k-1)] > \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho,\ R(k) \neq R_2(k-1)].$$

Equation 7 then shows that $\mathbf{P}_{\mathcal{A}}[\rho' \mid \rho] > \mathbf{P}_{\mathcal{A}}[\rho' \mid \rho,\ R(k) \neq R_2(k-1)]$. Plugging this result into Equation 6 finishes the proof.

■

The event ρ' that we considered for the sake of this counter-example is of low probability. Nevertheless the example demonstrates that the adversary can be stronger then what Rabin implicitly assumes: it can conceivably implement a strategy where it uses *all* the knowledge it gathers up to round k as well as the one from the current run in order get some control on the algorithm.

9 Is it possible to use Rabin's Lemma 3.3?

9.1 Rabin's Lemma is not useful as such.

As we mentioned in Section 5.3, Rabin's lemma 3.3 is a general mathematical tool that is not directly related with our problem. We begin by restating it.

Lemma 9.1 (Rabin) $\forall m, 1 \leq m \leq n$

$$\mathbf{P}_{\mathcal{A}}[\exists!\ i\ ;\ 1 \leq i \leq m,\ \beta_i(k) = \underset{j\ \in\ \{1,2,\ldots,m\}}{Max} \beta_j(k)] \sim 2/3.$$

This statement is stated slightly more precisely then the one of Rabin to take into account the fact that the distribution of the $\beta_j(k)$ depends apriori on n through the truncation at $b = \log_2 n + 4$. As we already saw, this mathematical result translates into:

$$\begin{aligned} \mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid\ |\mathcal{P}(k)| = m,\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ \mathcal{N}(k)] &= \mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid\ |\mathcal{P}(k)| = m,\ \mathcal{N}(k)] \\ &\sim 2/3, \end{aligned}$$

where we used the fact that, conditioned on $\mathcal{N}(k)$, $\mathcal{U}(k)$ is independent of $\{\pi_{k-1} = \rho\}$, and that then $\mathcal{U}(k)$ is independent of $i \in \mathcal{P}(k)$ because of the symmetric role played by all processes i.

As our Section 7 demonstrates, there is no hope to find any proof that would even prove the weak $\Omega(1/n)$ result. Nevertheless, it is interesting to try to work our way rigorously towards a lower bound of $\mathbf{P}_{\mathcal{A}}[W_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)]$, and try to introduce Rabin's quantity $\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ \mathcal{N}(k)]$ instead of the quantity $\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ \mathcal{N}_i(k)]$ that the rigorous probabilistic conditioning of Equation 1 was introducing. (For convenience we will refer to these two quantities as "Rabin's" quantity and "our" quantity.) For this purpose we will write what seems to be a normal substitute for inequality 1. The "missing link" in our argument will be in Inequality 12 which is flawed. We will give what seems to be a justification for it and then the reason for its being wrong.

Consider then the following substitute for inequality 1:

$$\begin{aligned}
&\mathbf{P}_{\mathcal{A}}[W_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)] \\
&\geq\ \mathbf{P}_{\mathcal{A}}\Big[W_i(k), \mathcal{N}_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \\
&=\ \mathbf{P}_{\mathcal{A}}\Big[\mathcal{N}_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big]\ \mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{N}_i(k)\ ,\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \\
&\geq\ \mathbf{P}_{\mathcal{A}}\Big[\mathcal{N}_i(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big]\ \mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{N}(k), \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big]. \qquad (12)
\end{aligned}$$

The informal (but flawed) justification for inequality 12 is as follows:

Consider a participating process j competing with i for access of the critical region during round k.

If j draws a new value in round k then of course this value is $\beta_j(k)$, so that:

$$\mathbf{P}_{\mathcal{A}}\Big[B_j(k) \geq l \mid \mathcal{N}_j(k), \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big] = \mathbf{P}_{\mathcal{A}}\Big[\beta_j(k) \geq l \mid \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big].$$

If j does not draw a new value it then holds an old value $B_j(k)$. This value having lost previously against other values tends to be smaller then if it had not, i.e., if it was new. This translates in formal terms into:

$$\begin{aligned}
\mathbf{P}_{\mathcal{A}}\Big[B_j(k) \geq l \mid \neg\mathcal{N}_j(k), \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big] &\leq\ \mathbf{P}_{\mathcal{A}}\Big[\beta_j(k) \geq l \mid \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big] \qquad (13) \\
&=\ \mathbf{P}_{\mathcal{A}}\Big[B_j(k) \geq l \mid\ \mathcal{N}_j(k), \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big].
\end{aligned}$$

The two previous equations then lead to:

$$\mathbf{P}_{\mathcal{A}}\Big[B_j(k) \geq l \mid\ \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big] \leq \mathbf{P}_{\mathcal{A}}\Big[B_j(k) \geq l \mid \mathcal{N}_j(k),\ \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big].$$

This means that a competitor j is always "better off" by choosing a new value and this justifies Equation 12.

Unfortunately Equation 13 does not hold. In effect, if j is a ($\log n - 1$)-survivor[5] then we actually have:

$$\mathbf{P}_{\mathcal{A}}\Big[B_j(k) \geq l \mid \neg\mathcal{N}_j(k), \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big] > \mathbf{P}_{\mathcal{A}}\Big[\beta_j(k) \geq l \mid \pi_{k-1} = \rho,\ j \in \mathcal{P}(k)\Big]\ !!$$

We can give a formal translation of this phenomenon.

[5] See Definition 7.1.

Rabin's quantity is derived from our's by further conditioning. On the other hand, the proof that we gave in Section 7 established that our quantity can be made arbitrary small (for some instances of ρ). These two facts along with Rabin's result say that the (conditional) probability of $\mathcal{N}(k)$ is very small. Precisely:

$$\begin{aligned}
&\mathbf{P}_{\mathcal{A}}\big[\mathcal{N}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ \mathcal{N}_i(k)\big] \\
&\quad= \frac{\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ \mathcal{N}_i(k)]}{\mathbf{P}_{\mathcal{A}}[\mathcal{U}(k) \mid \pi_{k-1} = \rho,\ i \in \mathcal{P}(k),\ \mathcal{N}(k)]} \\
&\quad\leq \frac{o(1/n)}{2/3} \\
&\quad= o(1/n).
\end{aligned}$$

This fact explains why the algorithm fails to achieve the weak lockout property of Definition 3.2. Recall that the the algorithm uses randomness in two different ways. The first use of randomness is within the mathematical procedure which selects with high probability a unique element from within a set of at much n elements. This procedure is based on the use of a distribution whose support is log n. Recall also that the lower bound of [1] in essence expresses that, in a distributed system, the selection of a unique process requires at least n values (of the shared variable) in presence of an adversary holding *complete* knowledge of the system. In order to incorporate the mathematical tool within a protocol, and use it to beat the lower bound, another use of randomness was devised to bar the adversary from holding too much knowledge.

What the previous result demonstrates, is that the conception of the algorithm actually did not succeed in preventing the access of the adversary to that knowledge: in some instances of ρ, in spite of the fact that the program variable R is supposed to erase the influence of the past, the adversary is able to bring surviving ghost values and defeat the algorithm.

9.2 At Last, a Lower Bound!

In this section we will exceptionally bring into light the different implicit probability spaces involved in the analysis. Recall that $\mathbf{P}_{\mathcal{A}}$ denotes the probability measure on the set of executions; the subscript $\mathcal{A}$ refers to the fact that these executions are not only determined by the random inputs, but also by the adversary. As discussed in page 2 this fact is crucial when we want to get a lower bound and reduce our analysis to the *sole* consideration of probabilistic facts. Recall also that the notation $\mathbf{P}$ denotes the probability controlling the mathematical random input variables. These random variables exist independently of the consideration of any adversary.

At this point we have basically established that the adversary can kill the algorithm due to the fact that the algorithm cannot enforce a good probability on $\mathcal{N}(k)$. An interesting question is then to determine whether the algorithm has the weak lockout property with a *good* (i.e., not too small) constant c, when we enforce that (i.e., when we condition on the fact that) $\mathcal{N}(k)$ holds. The answer is yes, but the formal proof is still tricky as we will see. The reason for asking for a good constant is that an obvious lower bound is obtained by considering the case where i picks the maximal lottery value b (and is the only one to do so). But this probability is bounded by $2^{-b+1} \leq 2^{-3}/n$. And if b was brought to $2\log_2 n$ it would become a $o(1/n)$!

Let us introduce the event $\mathcal{U}'(k)$:

$\mathcal{U}'(k) \overset{\text{def}}{=} \{$There is exactly one index $i \in \{1,2,\ldots,n\}$ such that $\beta_i(k) = \underset{j \in \{1,2,\ldots,n\}}{\text{Max}} \beta_j(k)\}$.

Note that Rabin's lemma implies that $\mathbf{P}[\mathcal{U}'(k)] \geq 2/3$.

Theorem 9.2 For every process $i = 1,\ldots,n$, for every round $k \geq 1$, for every adversary $\mathcal{A}$ and for every $(k-1)$-round run ρ,

$$\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \geq \frac{2}{3n}.$$

Proof:

We begin by reducing our study to within $\mathcal{U}'(k)$.

$$\begin{aligned}
&\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \\
&\quad = \ \mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{U}(k),\ \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big]\ \mathbf{P}_{\mathcal{A}}\Big[\mathcal{U}(k) \mid \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \\
&\quad \geq \ 2/3\ \mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{U}(k),\ \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big].
\end{aligned}$$

Remark that $\mathcal{U}'(k) \cap \mathcal{N}(k) \subseteq \mathcal{U}(k)$. Remark also that, in $\mathcal{U}(k)$, the event $W_i(k)$ is the same as the event $\{B_i(k)$ is the *unique* maximum among all the values $B_j(k)$ drawn by the participants of round $k\}$. One fundamental consequence of this fact is that, in $\mathcal{U}(k)$, the event $W_i(k)$ depends *only* of the values of the local variables $B_j(k)$; $j = 1,\ldots,n$. Hence:

$$\begin{aligned}
&\mathbf{P}_{\mathcal{A}}\Big[W_i(k) \mid \mathcal{U}'(k), \mathcal{N}(k), \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \\
&\quad = \ \mathbf{P}_{\mathcal{A}}\Big[B_i(k) = \underset{j \in \mathcal{P}(k)}{\text{Max}} B_j(k) \mid \mathcal{U}'(k),\ \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big] \\
&\quad = \mathbf{P}_{\mathcal{A}}\Big[\beta_i(k) = \underset{j \in \mathcal{P}(k)}{\text{Max}} \beta_j(k) \mid \mathcal{U}'(k),\ \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big]
\end{aligned}$$

(because for all j, $B_j(k) = \beta_j(k)$ in $\mathcal{N}(k)$),

$$\geq \mathbf{P}_{\mathcal{A}}\Big[\beta_i(k) = \underset{j \in \{1,2,\ldots,n\}}{\text{Max}} \beta_j(k) \mid \mathcal{U}'(k),\ \mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\Big]$$

(because $\{\beta_i(k) = \underset{j \in \mathcal{P}(k)}{\text{Max}} \beta_j(k)\} \supseteq \{\beta_i(k) = \underset{j \in \{1,2,\ldots,n\}}{\text{Max}} \beta_j(k)\}$),

$$= \mathbf{P}_{\mathcal{A}}\Big[\beta_i(k) = \underset{j \in \{1,2,\ldots,n\}}{\text{Max}} \beta_j(k) \mid \mathcal{U}'(k)\Big]$$

(because $\{\beta_i(k) = \underset{j \in \{1,2,\ldots,n\}}{\text{Max}} \beta_j(k)\}$ is independent from the event $\{\mathcal{N}(k),\ \pi_{k-1} = \rho,\ i \in \mathcal{P}(k)\}$),

$$= \mathbf{P}\Big[\beta_i(k) = \operatorname*{Max}_{j \in \{1,2,\dots,n\}} \beta_j(k) \mid \mathcal{U}'(k)\Big]$$

(because $\{\beta_i(k) = \operatorname*{Max}_{j \in \{1,2,\dots,n\}} \beta_j(k)\}$ and $\mathcal{U}'(k)$ are pure mathematical events that do not depend on $\mathcal{A}$).

But, in $\mathcal{U}'(k)$, the maximum $\operatorname*{Max}_{j \in \{1,2,\dots,n\}} \beta_j(k)$ is attained equiprobably by all indices i:

$$\forall i,\ \mathbf{P}\Big[\beta_i(k) = \operatorname*{Max}_{j \in \{1,2,\dots,n\}} \beta_j(k) \mid \mathcal{U}'(k)\Big] = 1/n.$$

■

The preceding proof calls for the some important comments:

It is *not* true that $\mathbf{P}_{\mathcal{A}}\Big[\beta_i(k) = \operatorname*{Max}_{j \in \mathcal{P}(k)} \beta_j(k) \mid \mathcal{U}'(k)\Big] = \mathbf{P}\Big[\beta_i(k) = \operatorname*{Max}_{j \in \mathcal{P}(k)} \beta_j(k) \mid \mathcal{U}'(k)\Big]$ i.e., that the adversary has no control over the event $\{\beta_i(k) = \operatorname*{Max}_{j \in \mathcal{P}(k)} \beta_j(k)\}$. Actually the former probability does not make sense because $|\mathcal{P}(k)|$ is a quantity that depends on the adversary. What we really mean is that it is not true that

$$\mathbf{P}_{\mathcal{A}}\Big[\beta_i(k) = \operatorname*{Max}_{j \in \mathcal{P}(k)} \beta_j(k) \mid \mathcal{U}'(k),\ |\mathcal{P}(k)| = m\Big] = \mathbf{P}\Big[\beta_i(k) = \operatorname*{Max}_{j \in \{1,\dots,m\}} \beta_j(k) \mid \mathcal{U}'(k)\Big].$$

Indeed the former probability is equal to $1/m$ whereas we proved in Section 6 that the latter is 0 when $m \leq n-1$ and when the adversary is $\mathcal{A}_1$!!

10 A General Mathematical Result

The result of this section is used in Section 8, where we used Bayes'Theorem in order to gain some a-priori knowledge of the state of the system based on some a-posteriori observations.

Lemma 10.1 *Let B and A be any real-valued random variables. Then*

$$\forall x \in \mathbf{R},\ \mathbf{P}[B \geq x \mid B \leq A] \leq \mathbf{P}[B \geq x].^{6}$$

Proof: The result is obvious if $\mathbf{P}[B \leq A] = 0$. We can therefore restrict ourselves to the case $\mathbf{P}[B \leq A] > 0$. Assume first that A is a constant a.

If $a < x$ the result is trivially true, so we assume that $a \geq x$. Set $\rho = \mathbf{P}[B > a]$ and $\beta = \mathbf{P}[B \geq x]$. Then:

$$\begin{aligned} \mathbf{P}[B \geq x \mid B \leq a] &= \frac{\mathbf{P}[x \leq B \leq a]}{\mathbf{P}[B \leq a]} \\ &= \tfrac{\beta-\rho}{1-\rho} \overset{\text{def}}{=} \phi_\beta(\rho). \end{aligned}$$

For $\beta \in [0,1]$, The function ϕ_β is not increasing on $[0,1)$. Hence:

$$\begin{aligned} \mathbf{P}[B \geq x \mid B \leq a] &\leq \phi_\beta(0) \\ &= \mathbf{P}[B \geq x]. \end{aligned}$$

[6] Without of generality we set $0/0 = 0$ whenever this quantity arises in the computation of conditional probabilities.

We then turn to the case where A is a general random variable. (Note that we cannot simply extend the previous proof in this case: A can sometimes be less then x and then it is not true anymore that $\{B > A\} \subseteq \{x \leq B\}$. We used this when saying that $\mathbf{P}[x \leq B \leq a] = \mathbf{P}[x \leq B] - \mathbf{P}[B > a]$.) We will let dP_A denote the distribution of A so that, for any measurable set U, $dP_A[U] = P[A \in U]$. Then:

$$\begin{array}{rcl} \mathbf{P}[A \geq B \geq x] & = & \int_a \mathbf{P}[a \geq B \geq x]\ dP_A(a) \\ & = & \int_a \mathbf{P}[B \geq x \mid B \leq a]\ \mathbf{P}[B \leq a]\ dP_A(a) \\ & \leq & \int_a \mathbf{P}[B \geq x]\ \mathbf{P}[B \leq a]\ dP_A(a) \end{array}$$

(We use here the result valid for $A = a$ constant),

$$\begin{array}{l} = \mathbf{P}[B \geq x]\ \int_a \mathbf{P}[B \leq a] dP_A(a) \\ = \mathbf{P}[B \geq x]\ \mathbf{P}[B \leq A]. \end{array}$$

Then we just need to divide by $\mathbf{P}[B \leq A]$ to get the result we are after. ■

Note that, if A is a discrete variable, its distribution is absolutely continuous with respect to the counting measure, so that the expression $\int_a \mathbf{P}[a \geq B \geq x] dP_A(a)$ reduces to $\sum_a \mathbf{P}[a \geq B \geq x]\mathbf{P}[A = a]$.

References

[1] Burns J., Fischer M., Jackson P., Lynch N. and Peterson G. Data requirements for implementation of n- process mutual exclusion using a single shared variable. *Journal of the ACM*, 29:183-205, (1982).

[2] E. Dijkstra. Solution of a Problem in Concurrent Programming Control. *Communications of the ACM*, 321, (1966)

[3] Philippe Flajolet and Nigel Martin. Probabilistic Counting Algorithms for Data Base Applications. *Journal of Computer and System Sciences*, 31:182–209, (1985).

[4] Michael Rabin. N-process mutual exclusion with bounded waiting by 4 log N- shared variable. *Journal of Computation and System Sciences*, 25:66–75 (1982).