RANDOMIZED DINING PHILOSOPHERS WITHOUT FAIRNESS ASSUMPTION

Marie Duflot, Laurent Fribourg and Claudine Picaronny
LSV, CNRS & ENS de Cachan,
61 av. du Prés. Wilson, 94235 Cachan cedex, France
{duflot,fribourg,picaro}@lsv.ens-cachan.fr

Abstract We consider Lehmann-Rabin's randomized solution to the well-known problem of the dining philosophers. Up to now, such an analysis has always required a "fairness" assumption on the scheduler: if a philosopher is continuously hungry then he must eventually be scheduled. In contrast here, we modify the algorithm in order to get rid of the fairness assumption. We claim that the spirit of the original algorithm is preserved. We prove that, for *any* (possibly unfair) scheduler, the modified algorithm converges: every computation reaches with probability 1 a configuration where some philosopher eats. Furthermore, we are now able to evaluate the expected time of convergence as a number of transitions. We show that, for some "malicious" scheduler, this expected time is at least exponential in the number N of philosophers.

1. Introduction

Recently, due to the rising risk of traffic congestion, there has been an increasing interest in providing differentiated Internet services, departing from the traditional notion of fairness for bandwidth allocations [1, 6]. This motivates reconsidering the need for the fairness assumption, which is classical in resource-allocation algorithms (see [9] chap 11). Here we consider Lehmann-Rabin's randomized solution to a special case of resource-allocation problem: the dining philosophers. N philosophers, $P_1, \cdots, P_N$, are seated around a table, and variously think or try to eat by using some shared forks. The problem is to find a distributed protocol guaranteeing that some philosopher will eventually eat. A philosopher is only able to execute a step provided he is selected by a general mechanism called *scheduler*. When he is selected by the scheduler, he executes exactly one action (and nothing is done by the others). Let $\mathcal{L}$ be the set of configurations, called here "legitimate", where some philosopher eats. We show here that the algorithm reaches $\mathcal{L}$ within a finite time with probability 1. In the following, we will call this property *convergence*. (This is sometimes called *progress* in the literature, see e.g. [9].) Up to now, such a proof has always required a "fairness" assumption on the scheduler: if a philosopher is continuously hungry (i.e., trying to eat) then he must eventually be scheduled. Fairness guarantees that there exist *rounds*, intervals in which each philosopher has been scheduled at least once. It is shown in [10, 9, 13, 11, 14] that within a constant number of rounds, the probability of reaching $\mathcal{L}$ is greater than 0. It follows that the algorithm converges towards a configuration of $\mathcal{L}$ with probability 1.

In contrast here, we consider *arbitrary* schedulers, without any fairness assumption (so we do not use the notion of rounds). We modify the original Lehmann-Rabin's algorithm by removing self-looping actions. We show that the new algorithm still converges towards $\mathcal{L}$ with probability 1. This is done by constructing a measure Δ over configurations that decreases with positive probability at each computation step (that does not reach $\mathcal{L}$). We thus propose a solution to the resource allocation problem for dining philosophers under *arbitrary* scheduler. We also show that the expected time of convergence, in terms of individual actions, is at least exponential in N, for some "malicious" scheduler.

The plan of the paper is as follows. After some preliminaries on Lehmann-Rabin's original algorithm (Section 2), we explain how we modify the algorithm (Section 3). We then prove the convergence of the modified algorithm for an arbitrary scheduler without fairness assumption (Section 4). In Section 5, we show that, for some malicious scheduler, the expected time of convergence is at least exponential in N. We conclude in Section 6.

2. Randomized Distributed Algorithms

2.1. Randomized Uniform Ring Systems

A *randomized uniform ring system* is a triple $(N, \rightarrow, Q)$ where N is the number of processes in the system, $\rightarrow$ is a state transition algorithm, and Q is the *alphabet*, i.e. a finite set of process states. The N processes $P_1, ..., P_N$ form a ring: there is an edge between two consecutive processes, which means that P_i can observe the states q_{i-1} and q_{i+1} of P_{i-1} and P_{i+1} respectively. Let Q be the state set of P_i. The system is *uniform* in the sense that $\rightarrow$ and Q are common to all processes. A *configuration* is an N-tuple of process states (or letters); if the current state of process P_i is $q_i \in Q$, then the configuration of the system is $x = q_1 q_2 \cdots q_N$. The state transition algorithm $\rightarrow$ is given as a set $\mathcal{R}$ of rules, consisting of *deterministic* or *probabilistic* rewrite rules. A deterministic rule is here of one the following forms:

- $q \rightarrow q'$
- $qr \rightarrow q'r$,
- $rq \rightarrow rq'$

where q, q', r denote states of Q. A probabilistic rule is here simply of the form:

$q \rightarrow q'_1$ with probability $1/2$
or q'_2 with probability $1/2$.

where q, q'_1, q'_2 denote states of Q.The letter q of the left-hand side is the *old* letter of the rule. The letter q' (with possible subscripts) of the right-hand side is the *new* letter of the rule. For readability, the old and new letters will often be written in bold font within rules. A rewrite rule $\mathtt{R}$ of left-hand side $\mathbf{q}$ is *applicable* at position i of a configuration x if the i-th letter of x is q. Likewise, a rewrite rule $\mathtt{R}$ of left-hand side $\mathbf{q}r$ (resp. $r\mathbf{q}$) is *applicable* at position i if the i-th letter of x is q and the $(i+1)$-th (resp. $(i-1)$-th) letter is r.

We say that process P_i is *enabled* if at least one rewrite rule is applicable to the i-th letter of x. Let $\mathcal{E}(x)$ be the set of indices of the enabled processes of x.

Given x and an enabled position i of x, a *transition* leads from x to the configuration x' obtained from x by changing the i-th letter of x equal to the old letter of some applicable rule, say $\mathtt{R}$, into the new letter. Such a transition is written $x \xrightarrow[\mathtt{R}]{i} x'$. Notation $x \xrightarrow[\mathcal{R}]{i} x'$ means $x \xrightarrow[\mathtt{R}]{i} x'$ for some rule $\mathtt{R} \in \mathcal{R}$.

A *(central) scheduler* is a mechanism that selects one enabled process at each step. The distributed system corresponds to repeated application of transition rules according to the philosopher chosen by the scheduler at each step. Given a scheduler $\mathcal{A}$, we are interested in proving the following *convergence* property (see [7, 2]): No matter which initial configuration x_0 one starts from, the probability that $\rightarrow$ under $\mathcal{A}$ reaches a legitimate configuration in a finite number of transitions is 1. This will be written: $Pr(x_0 \xrightarrow[\mathcal{R}]{\mathcal{A}}{}^* \mathcal{L})$. (See [4] for a formal definition.)

2.2. Lehmann-Rabin's algorithm

We present Lehmann-Rabin's algorithm [8] along the lines of [11]. Since we are only interested in the time before some philosopher eats, we disregard what follows this event (after state E has been reached).

The state set of each philosopher is $Q = \{T, H, \overleftarrow{W}, \overrightarrow{W}, \overleftarrow{S}, \overrightarrow{S}, \overleftarrow{D}, \overrightarrow{D}, E\}$. The letter T represents thinking, H that a philosopher is hungry, $\overleftarrow{W}$ (resp. $\overrightarrow{W}$) that a philosopher waits in order to attempt to pick up the left (resp. right) fork next time he is scheduled, $\overleftarrow{S}$ (resp. $\overrightarrow{S}$) that he is holding only the left (resp. right) fork, $\overleftarrow{D}$ (resp. $\overrightarrow{D}$) that he will put down the left (resp. right) fork next time he is scheduled and E that he eats. The details relating to the shared forks are omitted. Thus, for example, if P_i is in state $\overleftarrow{S}$ or P_{i-1} is in state $\overrightarrow{S}$, it means the variable representing the shared fork (between P_i and P_{i-1}) has been set to a value 'taken'. In this modeling, not all configurations are possible. This is because a fork can be taken by at most one process. More generally, we say that a configuration is *admissible* iff it does not contain any substring of the form $\overrightarrow{S}\overleftarrow{S}$, $\overrightarrow{S}\overleftarrow{D}$, $\overrightarrow{D}\overleftarrow{S}$ or $\overrightarrow{D}\overleftarrow{D}$. Henceforth, we will always implicitly focus on admissible configurations. The set $\mathcal{R}$ of transition rules is:

Q0: $\mathbf{T} \to \mathbf{T}$
Q1: $\mathbf{T} \to \mathbf{H}$
R0: $\mathbf{H} \to \overleftarrow{\mathbf{W}}$ with probability 1/2
or $\overrightarrow{\mathbf{W}}$ with probability 1/2.
R1: $\neg\overrightarrow{SD}\ \overleftarrow{\mathbf{W}} \to \neg\overrightarrow{SD}\ \overleftarrow{\mathbf{S}}$
R2: $\overrightarrow{SD}\ \overleftarrow{\mathbf{W}} \to \overrightarrow{SD}\ \overleftarrow{\mathbf{W}}$
R3: $\overrightarrow{\mathbf{W}}\ \neg\overleftarrow{SD} \to \overrightarrow{\mathbf{S}}\ \neg\overleftarrow{SD}$
R4: $\overrightarrow{\mathbf{W}}\ \overleftarrow{SD} \to \overrightarrow{\mathbf{W}}\ \overleftarrow{SD}$
R5: $\overleftarrow{\mathbf{S}}\ \neg\overleftarrow{SD} \to \mathbf{E}\ \neg\overleftarrow{SD}$
R6: $\overleftarrow{\mathbf{S}}\ \overleftarrow{SD} \to \overleftarrow{\mathbf{D}}\ \overleftarrow{SD}$
R7: $\neg\overrightarrow{SD}\ \overrightarrow{\mathbf{S}} \to \neg\overrightarrow{SD}\ \mathbf{E}$
R8: $\overrightarrow{SD}\ \overrightarrow{\mathbf{S}} \to \overrightarrow{SD}\ \overrightarrow{\mathbf{D}}$
R9: $\overleftarrow{\mathbf{D}} \to \mathbf{H}$
R10: $\overrightarrow{\mathbf{D}} \to \mathbf{H}$

where $\neg\overrightarrow{SD}$ (resp. $\neg\overleftarrow{SD}$) denotes any letter of Q distinct from $\overrightarrow{S}$ and $\overrightarrow{D}$ (resp. $\overleftarrow{S}$ and $\overleftarrow{D}$), and $\overrightarrow{SD}$ (resp. $\overleftarrow{SD}$) denotes $\overrightarrow{S}$ or $\overrightarrow{D}$ (resp. $\overleftarrow{S}$ or $\overleftarrow{D}$).

The rules describe the behaviour of a selected philosopher as follows: initially he thinks "repeatedly" (Q0); he becomes hungry (Q1); he decides randomly which fork to pick up first (R0); next he persists with his decision (R2 or R4) until he finally picks it up when available (R1 or R3), only putting it down later if he finds that his other fork is already held by his neighbour (R6 followed by R9, or R8 followed by R10); if he finds that his other fork is not held, he takes it and eats (R5 or R7).

This behaviour is depicted on figure 1 (drawn from [13]).

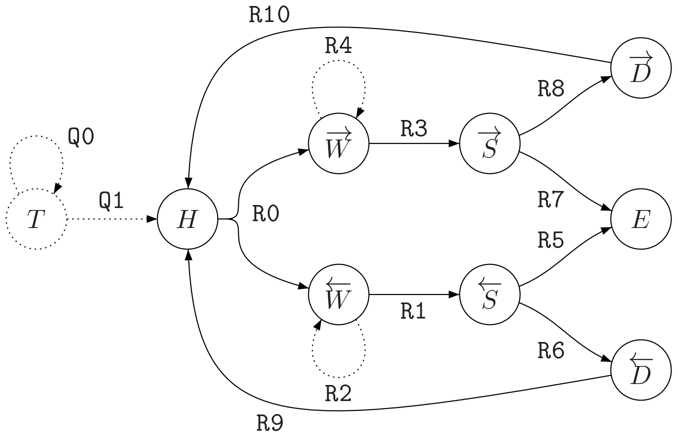

Figure 1. Illustration of the original algorithm. The dashed state and transitions are the ones removed in our variant.

3. Removal of stuttering rules

Let us observe that rule Q0 (resp. R2, R4) is "stuttering" in the sense that the old and new letters of the rule coincide. When a selected philosopher is thinking (resp. waiting for picking up a first fork held by a neighbour), a transition that does not change the configuration, may occur. This is depicted by a self-loop on state T (resp. $\overleftarrow{W}$, $\overrightarrow{W}$) in figure 1. We modify Lehmann-Rabin's algorithm by removing stuttering rules Q0, R2 and R4:

- without Q0, when a philosopher in state T is selected by the scheduler, his state always becomes H via Q1. States T and H then play the same role and will be merged together in the following;

- without R2 (resp. R4), when a philosopher waits for a first fork that is held by a neighbour, i.e., is in state $\overleftarrow{W}$ (resp. $\overrightarrow{W}$) and his left (resp. right) neighbour in state $\overrightarrow{SD}$ (resp. $\overleftarrow{SD}$), he is no longer enabled: no rule applies to him. In such a situation, the philosopher cannot be selected by the scheduler. Note that this differs from Lehmann-Rabin's original algorithm where every process can always be selected.

The rewrite system $\mathcal{R}$ is transformed into $\mathcal{R}' = \mathcal{R} - \{\mathtt{Q0}, \mathtt{Q1}, \mathtt{R2}, \mathtt{R4}\}$. The behaviour of $\mathcal{R}'$ is depicted on figure 1 without the dashed node and transitions. The new state set Q' is $Q - \{T\}$. Accordingly, the new legitimate set $\mathcal{L}'$ is $Q'^*EQ'^*$.

Discussion.
In Lehmann-Rabin's algorithm, a non-eating philosopher either thinks (state T) or tries to eat (states $\{H, W, S, D\}$). In our version of the algorithm, as the state T has been dropped, this philosopher can only try to eat. This feature may be seen as a limitation. Actually, since we have no fairness assumption, a philosopher can be indefinitely ignored by the scheduler, thus behaving in state H as he used to do originally in state T (i.e., not trying to pick up a fork). We thus claim that our modified algorithm is similar in spirit to the original one.

The original convergence property of Lehmann-Rabin's algorithm can be stated:
for any *fair* scheduler $\mathcal{A}$ and every $x \in Q^*\{H, W, S, D\}Q^*$, $Pr(x \xrightarrow[\mathcal{R}]{\mathcal{A}}{}^* \mathcal{L}) = 1$.
Surprisingly, as shown in section 4, for our modified version $\mathcal{R}'$ of $\mathcal{R}$, the convergence property holds with *no* fairness assumption on the scheduler, i.e.:

Theorem 1 *For any* arbitrary *scheduler $\mathcal{A}$ and every $x \in \{H, W, S, D\}^*$,*
$Pr(x \xrightarrow[\mathcal{R}']{\mathcal{A}}{}^* \mathcal{L}') = 1.$

4. Convergence of $\mathcal{R}'$

4.1. Scheme of the proof

We are going to prove Theorem 1 by using the following property proved in [4] (cf Theorem 1 of [2] and Theorem 5 of [3]):

Theorem 2 *Suppose that there exists a measure Δ and an ordering $\ll$ such that:*

$$\forall x \notin \mathcal{L}' \ \forall i \in \mathcal{E}(x) \quad \exists x' \ (x \xrightarrow[\mathcal{R}']{i} x' \wedge (\Delta(x') \ll \Delta(x) \vee x' \in \mathcal{L}'))$$

Then, for any (central) scheduler $\mathcal{A}$: $\forall x \quad Pr(x \xrightarrow[\mathcal{R}']{\mathcal{A}}{}^* \mathcal{L}') = 1.$

More precisely, we will find an appropriate "rewriting strategy" for probabilistic rule R0 (i.e., a fixed choice of the new letter $\overleftarrow{W}$ or $\overrightarrow{W}$, depending on the context of the old letter H in x, when rewriting x via R0), and a measure Δ such that

- the application of R0 under this strategy makes Δ decrease,

- the application of any other rule makes Δ decrease or leads to $\mathcal{L}'$.

Actually, a configuration x will be given with two lists π and ψ such that for every choice of the scheduler, there exists a configuration x' and two lists of indices π', ψ' constructed from π and ψ such that $x \to x'$ and $\Delta(x', \pi', \psi') \ll \Delta(x, \pi, \psi)$ (see Sections 4.2 and 4.3 for definitions of π and ψ).

In the following, the configurations are implicitly non-legitimate (i.e, belong to $(Q' - \{E\})^*$). Symbol W denotes a letter of $\{\overleftarrow{W}, \overrightarrow{W}\}$. Likewise, S (resp. D) denotes a letter of $\{\overleftarrow{S}, \overrightarrow{S}\}$ (resp. $\{\overleftarrow{D}, \overrightarrow{D}\}$). Let $\overleftarrow{Q} = \{H, \overleftarrow{W}, \overleftarrow{S}, \overleftarrow{D}\}$ and $\overrightarrow{Q} = \{H, \overrightarrow{W}, \overrightarrow{S}, \overrightarrow{D}\}$. (Note that $\overleftarrow{Q} \cap \overrightarrow{Q} = \{H\}$.)

In order to define Δ, we exploit the fact that any non-legitimate configuration can be decomposed into:
- "bonds", i.e. strings of two letters in $\overleftarrow{Q}\overrightarrow{Q}$.
- "anti-bonds", that are, roughly speaking, strings of two letters in $\overrightarrow{Q}\overleftarrow{Q}$.
- letters belonging to neither a bond nor an anti-bond. (These letters are in $\{W, S, D\}^*$, since every H belongs to a bond or an anti-bond; see Prop. 4 below.)

4.2. Bonds

A *bond* in a configuration x is a substring of x made of two consecutive letters in $\overleftarrow{Q}\overrightarrow{Q}$. The *index* of a bond $\overleftarrow{\alpha}\overrightarrow{\beta}$ is the position of its first letter $\overleftarrow{\alpha}$. Note that, due to letter H, two bonds may overlap: for example, in expression $\overleftarrow{W}H\overrightarrow{S}$ there are two overlapping bonds $\overleftarrow{W}H$ and $H\overrightarrow{S}$. In the following, given a configuration x, we focus on a sequence π of indices of *disjoint* bonds of x, i.e., such that $i + 2 \leq j$, for all consecutive indices i and j of π. We suppose also that π is *maximal*, i.e., such that between two consecutive indices $i, j \in \pi$ there is no bond of index k with $i + 2 \leq k \leq j - 2$. A maximal sequence π of indices of disjoint bonds of x is called a *bond list* of x. Note that such a list is not unique. $Bond(\pi)$ is defined as the set of letters of x at position ℓ such that $\ell = i$ or $\ell = i + 1$ for some $i \in \pi$.

Example: For the configuration $\overleftarrow{W}\overrightarrow{W}\overleftarrow{S}\overleftarrow{W}H\overrightarrow{D}$, there are two possible bond lists $\pi_1 = \{1, 4\}$ and $\pi_2 = \{1, 5\}$. Bonds are $\overleftarrow{W}\overrightarrow{W}$ and $\overleftarrow{W}H$ for π_1, and $\overleftarrow{W}\overrightarrow{W}$ and $H\overrightarrow{D}$ for π_2.

Henceforth, every non-legitimate configuration x will be provided with a bond list π. The bond list π_0 of the initial configuration x_0 is arbitrary. Given a configuration x, a bond list π of x, and a rewriting of x into x' via some rule of $\mathcal{R}'$, the bond list π' associated with x' is constructed from π as follows:

- If the rewriting changes a $H \in Bond(\pi)$ via probabilistic rule `R0`, we apply the following strategy:
  - if H is the first letter of a bond of π, then H is changed into $\overleftarrow{W}$,
  - if H is the second letter of a bond of π, then H is changed into $\overrightarrow{W}$.

Bonds of π are thus preserved, and we let: $\pi' = \pi$.

- If the rewriting creates a new bond of index k disjoint from every bond of π, then π' is $\pi \cup \{k\}$.
- In all other cases, we let $\pi' = \pi$.

Let $\Delta_1(x, \pi)$ be N minus the number of elements of π. The *bond coefficient* is 3 for D, 2 for H, 1 for W and 0 for S. The *weight of a bond* $\overleftarrow{\alpha}\overrightarrow{\beta}$ is the sum of the bond coefficients of $\overleftarrow{\alpha}$ and $\overrightarrow{\beta}$: for example, the weight of bond HH is 4. Then $\Delta_2(x, \pi)$ is the sum of the weights of all the bonds of x indexed by π.
Example: In the configuration $\overleftarrow{W}\overrightarrow{W}\overleftarrow{S}\overleftarrow{W}H\overrightarrow{D}$ of the previous example, we have $\Delta_1 = 4$, $\Delta_2 = 5$ for $\pi_1 = \{1, 4\}$, and $\Delta_1 = 4$, $\Delta_2 = 7$ for $\pi_2 = \{1, 5\}$.

$\Delta_3(x)$ is defined as the number of two-letter strings of the form $\overleftarrow{D}H$, $\overleftarrow{D}\overleftarrow{W}$, $H\overrightarrow{D}$ or $\overrightarrow{W}\overrightarrow{D}$ of x.

4.3. Anti-bonds

Given a configuration x and a bond list π of x, an *anti-bond* is a substring of two letters of x of one of the three forms:

1. $\overrightarrow{\gamma}\overleftarrow{\delta}$ with $\overrightarrow{\gamma} \notin Bond(\pi)$, $\overleftarrow{\delta} \notin Bond(\pi)$ and $\overrightarrow{\gamma} \in \overrightarrow{Q}$, $\overleftarrow{\delta} \in \overleftarrow{Q}$.
2. $\overrightarrow{\gamma}\overleftarrow{\alpha}$ with $\overrightarrow{\gamma} \notin Bond(\pi)$, $\overrightarrow{\gamma} \in \overrightarrow{Q}$ and $\overleftarrow{\alpha}$ 1st letter of a bond of π,
3. $\overrightarrow{\beta}\overleftarrow{\delta}$ with $\overleftarrow{\delta} \notin Bond(\pi)$, $\overleftarrow{\delta} \in \overleftarrow{Q}$ and $\overrightarrow{\beta}$ 2nd letter of a bond of π.

The *index of an anti-bond* of x is the position of its first letter ($\overrightarrow{\gamma}$ in cases 1-2, $\overrightarrow{\beta}$ in case 3).

Consider two bonds $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$ indexed by consecutive index i and i' of π. Then either:
- $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$ are contiguous (i.e: $i' = i + 2$) and there is no anti-bond between them (i.e: no anti-bond indexed by j with $i + 1 \leq j \leq i' - 1$), or
- $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$ are not contiguous (i.e: $i' \geq i + 3$).

In the latter case, it is easy to see that, between $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$, there is no substring of the form $\cdots\overleftarrow{\gamma}\cdots\overrightarrow{\delta}\cdots$ with $\overleftarrow{\gamma} \in \overleftarrow{Q}$ and $\overrightarrow{\delta} \in \overrightarrow{Q}$. (Otherwise, there would be a disjoint bond between $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$, and π would not be maximal.) Hence the substring between $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$ is of the form $\overrightarrow{Q}^*\overleftarrow{Q}^*$ with either no H (case H0) or just one H (case H1). More precisely, the substring delimited by the two bonds is of the form:
- H0: $\overleftarrow{\alpha}\overrightarrow{\beta}\overrightarrow{I}\overleftarrow{I}\overleftarrow{\alpha}'\overrightarrow{\beta}'$, or
- H1: $\overleftarrow{\alpha}\overrightarrow{\beta}\overrightarrow{I}H\overleftarrow{I}\overleftarrow{\alpha}'\overrightarrow{\beta}'$,

with $\overrightarrow{I} \in \{\overrightarrow{W}, \overrightarrow{S}, \overrightarrow{D}\}^*$ and $\overleftarrow{I} \in \{\overleftarrow{W}, \overleftarrow{S}, \overleftarrow{D}\}^*$. Let $\overrightarrow{\lambda}$ and $\overleftarrow{\mu}$ be the last letter of $\overrightarrow{\beta}\overrightarrow{I}$ and the first letter of $\overleftarrow{I}\overleftarrow{\alpha}'$ respectively. Between these bonds, there is:
- H0: a single anti-bond, viz: $\overrightarrow{\lambda}\overleftarrow{\mu}$, or
- H1: two overlapping anti-bonds, viz: $\overrightarrow{\lambda}H$ and $H\overleftarrow{\mu}$.

Given π, we construct an *anti-bond list* ψ of x as a set of indices obtained by putting, for every couple of non-contiguous consecutive bonds indexed by π:
- the index of $\overrightarrow{\lambda}\overleftarrow{\mu}$ in case H0,
- the index of either $\overrightarrow{\lambda}H$ or $H\overleftarrow{\mu}$ in case H1.

Given a configuration x and a bond list π of x, an anti-bond list ψ of x, is thus a maximal set of indices j of anti-bonds of (x, π). More precisely:

Proposition 3 *Given a configuration x and a bond list π of x, an anti-bond list ψ of x, is such that, for any couple of consecutive indices $i, i' \in \pi$, either:*
- the bonds indexed by i and i' are contiguous ($i' = i + 2$), in which case no anti-bond of ψ lies between them (i.e., no $j \in \psi$ such that $i + 1 \leq j \leq i' - 1$), or
- they are not contiguous ($i' \geq i + 3$), in which case exactly one anti-bond indexed by ψ lies between them (i.e., $\exists!\ j \in \psi : \ i + 1 \leq j \leq i' - 1$).

Note that, in any case, the occurrence of H (if any) between $\overleftarrow{\alpha}\overrightarrow{\beta}$ and $\overleftarrow{\alpha}'\overrightarrow{\beta}'$ always belongs to an anti-bond indexed by ψ. Formally:

Proposition 4 *For a given configuration $x \notin \mathcal{L}'$, a bond list π of x and an anti-bond list ψ, every H of x belongs to a bond of π or an anti-bond of ψ.*

Example: For the configuration $\overrightarrow{W} H \overleftarrow{S} \overrightarrow{D} \overrightarrow{W} \overleftarrow{W} \overleftarrow{D}$, we have one bond list $\pi = \{3, 7\}$ and two possible anti-bond lists $\psi_1 = \{1, 5\}$ and $\psi_2 = \{2, 5\}$. Anti-bonds are $\overrightarrow{W} H$ and $\overrightarrow{W}\overleftarrow{W}$ for ψ_1, $H\overleftarrow{S}$ and $\overrightarrow{W}\overleftarrow{W}$ for ψ_2.

Henceforth, every configuration x coupled with a bond list π, will be provided with an anti-bond list ψ. The anti-bond list ψ_0 associated with the initial couple (x_0, π_0) is arbitrary. Given a couple (x, π) and an associated anti-bond list ψ, the rewriting of x into x' via probabilistic rule `R0` preserves π when a bond is rewritten, using the strategy described in section 4.2. Rewriting via `R0` also preserves ψ using the following strategy:
- if H is the 1st letter of an anti-bond of ψ, then H is changed into $\overrightarrow{W}$;
- if H is the 2nd letter of an anti-bond of ψ, then H is changed into $\overleftarrow{W}$.

It is easy to see that this strategy is compatible with the one for bonds: if H is shared by a bond of π and an anti-bond of ψ, both strategies agree for rewriting H either into $\overleftarrow{W}$ or $\overrightarrow{W}$. For example, if H is in $\overleftarrow{S}\mathbf{H}\overleftarrow{S}$, the expression rewrites to $\overleftarrow{S}\overrightarrow{\mathbf{W}}\overleftarrow{S}$. The rewriting of x into x' via the other rules transforms π into π' as explained in section 4.2, and ψ into ψ' where $\psi' = \psi$ except in some cases where D is replaced by H via `R9` or `R10` (see [5], for details).

We say that an anti-bond is *oriented leftwards* (resp. *oriented rightwards*) if it is of the form $\{\overrightarrow{S}, \overrightarrow{D}\}\{\overleftarrow{W}, H\}$ (resp. $\{\overrightarrow{W}, H\}\{\overleftarrow{S}, \overleftarrow{D}\}$).

Given a bond list π and an anti-bond A of index k oriented leftwards (resp. rightwards), the π-*distance* of A is $k - i$ (resp. $i - k$) where i is the index of the closest bond of π to the left (resp. right) of A.

Let $\Delta_4(x, \psi)$ be N minus the number of oriented anti-bonds of x indexed by ψ and $\Delta_5(x, \pi, \psi)$ be the sum of π-distances of the oriented anti-bonds of ψ.

The *anti-bond coefficient* is 3 for H, 2 for W, 1 for S and 0 for D. The *weight of an anti-bond* $\overrightarrow{\alpha}\overleftarrow{\beta}$ is the sum of the anti-bond coefficients of $\overrightarrow{\alpha}$ and $\overleftarrow{\beta}$: for example, the weight of $H\overleftarrow{W}$ is 5.

Let $\Delta_6(x, \psi)$ be the sum of the weights of the anti-bonds of x indexed by ψ.

Example: In the configuration $\overrightarrow{W}H\overleftarrow{S}\overrightarrow{D}\overrightarrow{W}\overleftarrow{W}\overleftarrow{D}$ of the previous example, we have $\Delta_4 = 7 (= N)$, $\Delta_5 = 0$, $\Delta_6 = 9$ for $\psi_1 = \{1, 5\}$, and $\Delta_4 = 6$, $\Delta_5 = 1$, $\Delta_6 = 8$ for $\psi_2 = \{2, 5\}$. In ψ_1 no anti-bond is oriented. In ψ_2, the anti-bond $H\overleftarrow{S}$ is oriented rightwards. Its π-distance with token $\overleftarrow{S}\overrightarrow{D}$ is 1.

4.4. Measure Δ

The *WSD-coefficient* is 2 for W, 1 for S and 0 for D. Let $\Delta_7(x)$ be the sum of the WSD-coefficients of all the letters of x distinct from H.

Given a configuration $x \notin \mathcal{L}'$, a bond list π and an anti-bond list ψ, measure Δ is defined as 7-tuple $(\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5, \Delta_6, \Delta_7)$. The evolution of Δ along a computation is illustrated on an example in the first appendix.

By Proposition 4, any configuration can be decomposed into bonds, anti-bonds and W, S, D-letters. Using this fact, it can be shown by case analysis that, under the strategies defined above for `R0`, Δ decreases whenever x rewrites to x' for lists π' and ψ' constructed from π and ψ as described in sections 4.2 and 4.3. (The full proof is given in [5].) Formally, let $\ll$ be the lexicographic extension of $<$; we have:

Proposition 5 *For every $x \notin \mathcal{L}'$, every bond list π and anti-bond list ψ of x, and every position i in $\mathcal{E}(x)$, there exists a configuration x', a bond list π' and an anti-bond list ψ' of x' such that:*

$$x \xrightarrow[\mathcal{R}']{i} x' \wedge (x' \in \mathcal{L}' \ \vee \ \Delta(x', \pi', \psi') \ll \Delta(x, \pi, \psi)).$$

Theorem 1 then follows from proposition 5 and Theorem 2.

5. Expected Time of Convergence

In traditional approaches (see e.g. [9]) the time is measured in terms of rounds (intervals in which each process has been scheduled at least once). The time is never evaluated as a number of transitions.

With our approach, we do not make any assumption on this round time. We evaluate the expected time of convergence as a number of transitions. It turns out that, for some "malicious" scheduler such a time can be "very" long. In 2nd appendix, we show indeed that, for some configuration x_0 (of the form $\overrightarrow{S}\overrightarrow{S}\cdots\overrightarrow{S}$) and some scheduler $\mathcal{A}_0$, one can stay out of $\mathcal{L}'$ during an expected time at least exponential in N. This gives us an exponential lower bound for the expected time of convergence. In other words, the expected number of transitions in a round can be exponential.

6. Conclusion

We have shown in this paper that a modified version of Lehmann-Rabin's algorithm always converges, whatever the (possibly unfair) scheduler is. We claim that our modified algorithm preserves the spirit of the original one.

With our approach, we are also able to evaluate the expected time of convergence as a number of transitions. We have shown that this time may be exponential for some malicious scheduler.

An interesting further work would be to remove the fairness assumption with more complex connection topologies of dining philosophers as, e.g., in [12].

References

[1] T. Bonald and L. Massouli. Impact of fairness on Internet performance. In *Proc. of Joint International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS/Performance 2001)*, Cambridge, USA, 2001, pp. 82–91.

[2] J. Beauquier, J. Durand-Lose, M. Gradinariu, and C. Johnen. Token based self-stabilizing uniform algorithms. *J. of Parallel and Distributed Systems*, To appear.

[3] S. Dolev, A. Israeli, and S. Moran. Analyzing expected time by scheduler-luck games. *IEEE Transactions on Software Engineering*, 21(5), 1995, pp. 429–439.

[4] M. Duflot, L. Fribourg, and C. Picaronny. Randomized distributed algorithms as Markov chains. In *Proc. 15th Int. Conf. on Distributed Computing (DISC'01)*, Lisbon, 2001, pp. 240–254. (Extended version available on `http://www.lsv.ens-cachan.fr/Publis/`).

[5] M. Duflot, L. Fribourg, and C. Picaronny. Randomized dining philosophers without fairness assumption. Research Report LSV-01-11, Cachan, France, December 2001. Available on `http://www.lsv.ens-cachan.fr/Publis/`

[6] P. Gevros, F. Risso, and P. Kirstein. Analysis of a Method for Differential TCP Service. In *Proc. of the IEEE GLOBECOM'99*, Rio de Janeiro, 1999.

[7] H. Kakugawa and M. Yamashita. Uniform and Self-Stabilizing Token Rings Allowing Unfair Daemon. *IEEE Trans. Parallel and Distributed Systems*, 8(2), 1997, pp. 154–163.

[8] D. Lehmann and M.O. Rabin. The advantages of free choice: a symmetric and fully-distributed solution to the dining philosophers problem. In *Proc. 8th Annual ACM Symposium on Principles of Programming Languages(POPL'81)*, Williamsburg, U.S.A., 1981, pp. 133–138.

[9] N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., 1996.

[10] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proc. 13th Annual ACM Symposium on Principles of Distributed Computing (PODC'94)*, Los Angeles, USA, 1994, pp. 314–323.

[11] A.K. McIver. Quantitative program logic and efficiency in probabilistic distributed algorithms. Tech. report, Computing Lab., Oxford U., UK, 1998. (Extended version of "Quantitative program logic and performance", Proc. of 5th Int AMAST Workshop, ARTS '99).

[12] O.M. Herescu and C. Palamidessi. On the Generalized Dining Philosophers Problem. In Proc. 20th Annual ACM Symposium on Principles of Distributed Computing (PODC'01), Newport, USA, 2001, pp. 81–89.

[13] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1), 1986, pp. 53–72.

[14] A. Pogosyants and R. Segala. Formal verification of timed properties for randomized distributed algorithms. In *Proc. 14th Annual ACM Symposium on Principles of Distributed Computing (PODC'95)*, Ontario, Canada, 1995, pp. 174–183.

Appendix: An example of computation

The general behaviour of bonds should be clear: with our strategy, once a bond is created, it never disappears (Δ_1 never increases) and stays in fixed position. On the example below, the two leftmost and two rightmost letters of each configuration are the bonds of π (π is here the only possible bond list).

Let us explain the general evolution of an anti-bond A located between two consecutive (non contiguous) bonds B_1 and B_2. Once oriented, say rightwards, A moves towards B_2 until it overlaps with it. It may then lose its orientation and become oriented later in the other direction. Such a U-turn is possible only if the left letter of B_2 is rewritten. A thus oscillates between B_1 and B_2. The total number of U-turns is finite as each bond coefficient can decrease at most 3 times. For example the first letter of a bond can change from $\overleftarrow{D}$ to H then to $\overleftarrow{W}$ and to $\overleftarrow{S}$. Once it is $\overleftarrow{S}$, if this process is selected by the scheduler and as the right fork is not taken (the right neighbour is in state $\overrightarrow{D}$, H, $\overrightarrow{W}$ or $\overrightarrow{S}$) the philosopher enters state E and $\mathcal{L}$ is reached. A computation with Δ_1 and Δ_3 constant illustrated on Figure A.1.

$$
\begin{array}{llllllll}
 & \overleftarrow{W}\underset{\Delta_5=3}{\underline{\overrightarrow{W}\overleftarrow{\mathbf{D}}}}\overleftarrow{D}\overleftarrow{D}\overrightarrow{S} & \xrightarrow{\Delta_5} & \overleftarrow{W}\overrightarrow{W}\underset{\Delta_5=2}{\underline{\mathbf{H}\overleftarrow{D}}}\overleftarrow{D}\overrightarrow{S} & \xrightarrow{\Delta_6} & \overleftarrow{W}\overrightarrow{W}\underset{\Delta_5=2}{\underline{\overrightarrow{W}\overleftarrow{\mathbf{D}}}}\overleftarrow{D}\overrightarrow{S} & \xrightarrow{\Delta_5} & \overleftarrow{W}\overrightarrow{W}\overrightarrow{W}\underset{\Delta_5=1}{\underline{H\overleftarrow{\mathbf{D}}}}\overrightarrow{S} \\
\xrightarrow{\Delta_2} & \overleftarrow{W}\overrightarrow{W}\overrightarrow{W}\underset{\Delta_5=0}{\underline{H\mathbf{H}}}\overrightarrow{S} & \xrightarrow{\Delta_2} & \overleftarrow{W}\overrightarrow{W}\overrightarrow{W}\underset{\Delta_5=0}{\underline{\mathbf{H}\overleftarrow{W}}}\overrightarrow{S} & \xrightarrow{\Delta_6} & \overleftarrow{W}\overrightarrow{W}\overrightarrow{W}\underset{\Delta_5=0}{\underline{\overrightarrow{\mathbf{W}}\overleftarrow{W}}}\overrightarrow{S} & \xrightarrow{\Delta_4} & \overleftarrow{W}\overrightarrow{W}\mathbf{W}\underset{\Delta_5=3}{\underline{\overrightarrow{S}\overleftarrow{W}}}\overrightarrow{S} \\
\xrightarrow{\Delta_7} & \overleftarrow{W}\overrightarrow{W}\overrightarrow{S}\underset{\Delta_5=3}{\underline{\overrightarrow{\mathbf{S}}\overleftarrow{W}}}\overrightarrow{S} & \xrightarrow{\Delta_6} & \overleftarrow{W}\overrightarrow{W}\overrightarrow{S}\underset{\Delta_5=3}{\underline{\overrightarrow{\mathbf{D}}\overleftarrow{W}}}\overrightarrow{S} & \xrightarrow{\Delta_5} & \overleftarrow{W}\mathbf{W}\underset{\Delta_5=2}{\underline{\overrightarrow{S}H}}\overleftarrow{W}\overrightarrow{S} & \xrightarrow{\Delta_2} & \overleftarrow{W}\overrightarrow{S}\underset{\Delta_5=2}{\underline{\mathbf{S}H}}\overleftarrow{W}\overrightarrow{S} \\
\xrightarrow{\Delta_6} & \overleftarrow{W}\overrightarrow{S}\underset{\Delta_5=2}{\underline{\overrightarrow{D}\mathbf{H}}}\overleftarrow{W}\overrightarrow{S} & \xrightarrow{\Delta_6} & \overleftarrow{W}\overrightarrow{S}\underset{\Delta_5=2}{\underline{\overrightarrow{\mathbf{D}}\overleftarrow{W}}}\overleftarrow{W}\overrightarrow{S} & \xrightarrow{\Delta_5} & \overleftarrow{W}\underset{\Delta_5=1}{\underline{\overrightarrow{S}\mathbf{H}}}\overleftarrow{W}\overleftarrow{W}\overrightarrow{S} & \xrightarrow{\Delta_6} & \overleftarrow{W}\underset{\Delta_5=1}{\underline{\overrightarrow{S}\overleftarrow{W}}}\overleftarrow{W}\overleftarrow{W}\overrightarrow{S}
\end{array}
$$

Figure A.1. A computation: each configuration has a single anti-bond, which is underlined and labelled with the value of the Δ_5-distance; the bold letter is the letter to be changed; the arrow depicting each transition is labelled with the first decreasing Δ-component.

Each transition is represented by an arrow labelled with the first decreasing component of Δ. For each configuration, we also give the corresponding measure Δ_5. As there is just one anti-bond A, $\Delta_5(x, \pi, \psi)$ is either the π-distance of A when it is oriented, or 0 otherwise. Note that, in the last configuration, the anti-bond $\overrightarrow{S}\overleftarrow{W}$ cannot be rewritten without reaching $\mathcal{L}'$.

Appendix: Expected Time of Convergence

Let us show on a example that the expected time of convergence is at least exponential in the number N of processes for some "malicious" scheduler. We exhibit a scheduler which, starting from the uniform configuration $x_0 = \overrightarrow{S}^N$, goes to $x_1 = \overrightarrow{S}^{N-2}\overleftarrow{W}\overleftarrow{S}$ within a constant expected time. Then, from x_1, it can stay in the set of configurations $\{\overrightarrow{S}^i\overleftarrow{W}\overleftarrow{S}^j | i+j+1 = N\} \cup \{\overrightarrow{S}^i\overrightarrow{W}\overleftarrow{S}^j | i+j+1 = N\}$ during an expected time exponential in N.

Consider $x_0 = \overrightarrow{S}^N$ as a starting configuration. Let us first show that the scheduler may reach the configuration $x_1 = \overrightarrow{S}^{N-2}\overleftarrow{W}\overleftarrow{S}$ in a finite amount of time: It selects a $\overrightarrow{S}$ and applies `R8.R10.R0` to obtain $\overrightarrow{S}^{N-1}\overrightarrow{W}$ or $\overrightarrow{S}^{N-1}\overleftarrow{W}$. In the first case, it applies `R3` and goes on. In the second case, it selects the last $\overrightarrow{S}$ and applies `R8.R10.R0` to obtain $\overrightarrow{S}^{N-2}\overrightarrow{W}\overleftarrow{W}$ or $\overrightarrow{S}^{N-2}\overleftarrow{W}\overleftarrow{W}$. In the first case, it applies `R3` to $\overrightarrow{W}$ and begins again. In the second case, it applies `R2` to the right $\overleftarrow{W}$. The expected time E of going from x_0 to x_1 may be easily computed: $E = 15$.

Now, we describe two possible choices of the scheduler on a configuration of the form $\overrightarrow{S}^{i}\overleftarrow{W}\overleftarrow{S}^{j}$ with $i \geq 2$ and $i + j + 1 = N$. These two choices are the application of four consecutive rules, one of which is a probabilistic one. The three first rules are the same: select the rightmost $\overrightarrow{S}$ and applies `R8.R10.R0`. This yields: $\overrightarrow{S}^{i-1}\overleftarrow{W}\overleftarrow{W}\overleftarrow{S}^{j}$ or $\overrightarrow{S}^{i-1}\overrightarrow{W}\overleftarrow{W}\overleftarrow{S}^{j}$.
From $\overrightarrow{S}^{i-1}\overleftarrow{W}\overleftarrow{W}\overleftarrow{S}^{j}$, apply `R1` to the rightmost $\overleftarrow{W}$ to get $\overrightarrow{S}^{i-1}\overleftarrow{W}\overleftarrow{S}^{j+1}$.
From $\overrightarrow{S}^{i-1}\overrightarrow{W}\overleftarrow{W}\overleftarrow{S}^{j}$, the scheduler can choose $\overrightarrow{W}$ or $\overleftarrow{W}$, which leads to the two cases:

(A) The scheduler applies `R1` to $\overleftarrow{W}$ and yields $\overrightarrow{S}^{i-1}\overrightarrow{W}\overleftarrow{S}^{j+1}$,

(B) The scheduler applies `R3` to $\overrightarrow{W}$ and yields $\overrightarrow{S}^{i}\overleftarrow{W}\overleftarrow{S}^{j}$.

Symmetrically, using `R6.R9.R0`, on a configuration $\overrightarrow{S}^{i}\overrightarrow{W}\overleftarrow{S}^{j}$ (with $j \geq 2$ and $i + j + 1 = N$), one goes to $\overrightarrow{S}^{i}\overrightarrow{W}\overleftarrow{W}\overleftarrow{S}^{j-1}$ or $\overrightarrow{S}^{i}\overrightarrow{W}\overrightarrow{W}\overrightarrow{S}^{j-1}$.
From $\overrightarrow{S}^{i}\overrightarrow{W}\overrightarrow{W}\overrightarrow{S}^{j-1}$, apply `R3` to the leftmost $\overrightarrow{W}$, to get $\overrightarrow{S}^{i+1}\overrightarrow{W}\overleftarrow{S}^{j-1}$.
From $\overrightarrow{S}^{i}\overrightarrow{W}\overleftarrow{W}\overleftarrow{S}^{j-1}$, the scheduler can choose $\overrightarrow{W}$ or $\overleftarrow{W}$, which leads to the two cases:

(A') The scheduler applies `R3` to $\overrightarrow{W}$ and yields $\overrightarrow{S}^{i+1}\overleftarrow{W}\overleftarrow{S}^{j-1}$.

(B') The scheduler applies `R1` to $\overleftarrow{W}$ and yields $\overrightarrow{S}^{i}\overrightarrow{W}\overleftarrow{S}^{j}$.

The scheduler can iterate such application of four consecutive rules until the system reaches an "end" configuration, i.e, a configuration of the form:

$x_{end} = \overrightarrow{S}^{N-2}\overrightarrow{W}\overleftarrow{S}$ or $y_{end} = \overrightarrow{S}\overleftarrow{W}\overleftarrow{S}^{N-2}$.

Let us consider the following "malicious" scheduler: From $\overrightarrow{S}^{i}\overleftarrow{W}\overleftarrow{S}^{j}$ (with $i \geq 2$ and $i + j + 1 = N$), it chooses (A) or (B) according to the compared values of i and j. Precisely: it chooses (A) if $j \geq i$, and (B) if $j < i$.

Symmetrically, from $\overrightarrow{S}^{i}\overrightarrow{W}\overleftarrow{S}^{j}$ (with $j \geq 2$ and $i + j + 1 = N$): it chooses (A') if $i \geq j$, and (B') if $i < j$.

For this scheduler, let us now compute the expected time for reaching x_{end} or y_{end}, starting from a configuration $\overrightarrow{S}^{i}\overrightarrow{W}\overleftarrow{S}^{j}$ (resp. $\overrightarrow{S}^{i}\overleftarrow{W}\overleftarrow{S}^{j}$), with $j \geq 2$ (resp. $i \geq 2$) and $i + j + 1 = N$. Let us abbreviate this expected time as $E[\overrightarrow{i}]$ (resp. $E[\overleftarrow{i}]$). We have:

$E[\overrightarrow{i}] = 4 + 1/2\ E[\overrightarrow{i+1}] + 1/2\ E[\overleftarrow{i+1}]$, for $1 \leq i$ and $2i \geq N-2$.
$E[\overrightarrow{i}] = 4 + 1/2\ E[\overrightarrow{i+1}] + 1/2\ E[\overrightarrow{i}]$, for $1 \leq i$ and $2i < N-2$.
$E[\overleftarrow{i}] = 4 + 1/2\ E[\overrightarrow{i-1}] + 1/2\ E[\overleftarrow{i-1}]$, for $2 \leq i$ and $2i \leq N-2$.
$E[\overleftarrow{i}] = 4 + 1/2\ E[\overleftarrow{i-1}] + 1/2\ E[\overleftarrow{i}]$, for $2 \leq i$ and $2i > N-2$.
$E[\overrightarrow{N-2}] = 0$.
$E[\overleftarrow{1}] = 0$.

We then solve this linear system. The symmetry shows that $E[\overrightarrow{i}] = E[\overrightarrow{N-1-i}]$, for all $1 \leq i < N-1$. Let m be the integral part of $(N-1)/2$. The result is:

$E[\overrightarrow{i}] = 8(3 \times 2^{m-1} + 5m - 6 - i)$, for $1 \leq i < m$.
$E[\overrightarrow{i}] = 8(3(2^{m-1} - 2^{i-m}) - 2m + i + 1))$ for $m \leq i < N-1$.

In particular, the time to go from x_1 to x_{end} (hence x_0 to x_{end}) is $E[\overleftarrow{N-2}] = E[\overrightarrow{1}] = 8(3 \times 2^{m-1} + 5m - 7) \simeq 3 \times 2^{m+2}$.
This shows that we can stay out of $\mathcal{L}'$ an exponential expected number of steps, hence the upper bound for the expected time of convergence for a general scheduler is at least exponential.