# Randomized Finite-State Distributed Algorithms As Markov Chains

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.** Distributed randomized algorithms, when they operate under a memoryless scheduler, behave as finite Markov chains: the probability at $n$-th step to go from a configuration $x$ to another one $y$ is a constant $p$ that depends on $x$ and $y$ only. By Markov theory, we thus know that, no matter where the algorithm starts, the probability for the algorithm to be after $n$ steps in a "recurrent" configuration tends to 1 as $n$ tends to infinity. In terms of *self-stabilization* theory, this means that the set $\mathcal{R}ec$ of recurrent configurations is included into the set $\mathcal{L}$ of "legitimate" configurations. However in the literature, the convergence of self-stabilizing randomized algorithms is always proved in an elementary way, without explicitly resorting to results of Markov theory. This yields proofs longer and sometimes less formal than they could be. One of our goals in this paper is to explain convergence results of randomized distributed algorithms in terms of Markov chains theory.
Our method relies on the existence of a non-increasing measure $\varphi$ over the configurations of the distributed system. Classically, this measure counts the number of tokens of configurations. It also exploits a function $D$ that expresses some distance between tokens, for a fixed number $k$ of tokens. Our first result is to exhibit a sufficient condition Prop on $\varphi$ and $D$ which guarantees that, for memoryless schedulers, every recurrent configuration is legitimate. We extend this property Prop in order to handle arbitrary schedulers although they may induce non Markov chain behaviours. We then explain how Markov's notion of "lumping" naturally applies to measure $D$, and allows us to analyze the expected time of convergence of self-stabilizing algorithms. The method is illustrated on several examples of mutual exclusion algorithms (Herman, Israeli-Jalfon, Kakugawa-Yamashita).

## 1 Introduction

A randomized distributed system is a network of $N$ finite-state processes whose states are modified via randomized local actions. A process $P$ is *enabled* when its state and the states of its neighbours allow $P$ to be the siege of some action. A *scheduler* is a mechanism which, at each step, selects a subset of enabled processes: all the selected processes then execute synchronously an action and change their state accordingly. A scheduler is *memoryless* when the choice of the enabled processes depends only on the current network configuration $x$, i.e. the current value of the $N$-tuple of process states. Distributed randomized algorithms, when they operate under a memoryless scheduler, behave as finite Markov chains: the probability at $n$-th step to go from a configuration $x$ to another one $y$ is a constant $p$ that depends on $x$ and $y$ only. By Markov theory, we thus know that, no matter where the algorithm starts, the probability for the algorithm to be after $n$ steps in a "recurrent" configuration tends to 1 as $n$ tends to infinity. In terms of *self-stabilization* theory [5,6], this means that the set $\mathcal{R}ec$ of recurrent configurations is included into the set $\mathcal{L}$ of "legitimate" configurations. However, in the literature (see, e.g., [10,11,9,12,13]) the convergence of self-stabilizing randomized algorithms is often proved in an elementary way, without explicitly resorting to results of Markov theory. This yields proofs longer and sometimes less formal than they could be. One of our goals in this paper is to explain convergence results of randomized distributed algorithms in terms of Markov chains theory.

We focus here on $\varphi$-algorithms where $\varphi$ is a measure over configurations that characterizes the set $\mathcal{L}$ of legitimate states, and never increases along any computation. For example, for mutual exclusion algorithms, a typical measure $\varphi$ counts the number of "tokens", the legitimate configurations being those with only one token. Our first contribution is to exhibit a general property of $\varphi$-algorithms, called Prop, which guarantees that, under a memoryless scheduler, every recurrent configuration is legitimate ($\mathcal{R}ec \subseteq \mathcal{L}$). We also show that this property Prop extends naturally to guarantee the convergence of distributed algorithms under arbitrary schedulers, although these algorithms may not behave any longer as Markov chains. Finally we explain how to use the "lumping" method of Markov theory in order to derive a simpler distributed algorithm, which allows us to compute the expected time of convergence of the original algorithm. This gives us a formal justification for the method used, e.g., by Herman [10].

The plan of the paper is as follows. After some preliminaries (Section 2), we define sufficient property Prop that ensures convergence of distributed algorithms under memoryless scheduler (Section 3). Then Prop is extended in order to treat arbitrary schedulers in Section 4. We explain how to use the lumping method in order to analyze the time of convergence in Section 5. We conclude in Section 6.

## 2 Preliminaries

For the sake of simplicity we focus in this paper on the simple topology of linear networks, i.e. rings. We also assume that the communication between processes is done through the reading of neighbours' state.

### 2.1 Randomized Uniform Ring Systems

The following material is borrowed from [13] (cf. [1,4]). A *randomized uniform ring system* is a triple $R = (N, \rightarrow, Q)$ where $N$ is the number of processes in the system, $\rightarrow$ is a state transition algorithm, and $Q$ is a finite set of process states. The $N$ processes $P_0, P_1, \ldots, P_{N-1}$ form a ring: the fact that there is an edge from $P_{i-1}$ to $P_i$ means that $P_i$ can observe the state $q_{i-1}$ of $P_{i-1}$[1]. Calculations on indices $i$ of processes are done modulo $N$. Let $Q$ be the state set of $P_i$. The system is *uniform* in the sense that $\rightarrow$ and $Q$ are common to every process. A *configuration* of $R$ is an $N$-tuple of process states; if the current state of process $P_i$ is $q_i \in Q$, then the configuration of the system is $x = (q_0, q_1, \cdots, q_{N-1})$. We denote by $X$ the set of all configurations, i.e., $X = Q^N$. The state transition algorithm $\rightarrow$ is given as a set of *guarded commands* of the form:

IF <guard$_1$> THEN <command$_1$>
IF <guard$_2$> THEN <command$_2$>
...
IF <guard$_m$> THEN <command$_m$>

Here a guard is a predicate of the form $g(q_{i-1}, q_i)$. A command modifies the state $q_i$ of process $P_i$. It is either a deterministic action of the form $q_i' = h(q_{i-1}, q_i)$, or a probabilistic action of the form $q_i' = h^{\omega}(q_{i-1}, q_i)$ with probability $p^{\omega} > 0$ ($\omega \in \{0,1\}$ and $\sum_{\omega=0,1} p^{\omega} = 1$). We say that process $P_j$ is *enabled* if a guard $g(q_{j-1}, q_j)$ is true. The set of indices of enabled processes of a configuration $x$ is denoted by $\mathcal{E}(x)$. If no process is enabled at configuration $x$ ($\mathcal{E}(x) = \emptyset$), we say that there is a *deadlock*. Otherwise, a *scheduler* $\mathcal{A}$, is a mechanism which selects a nonempty subset $S$ of $\mathcal{E}(x)$. A transition then leads from $x$ to $x' = (q_0', q_1', \cdots, q_N')$ with

$$q_j' = \begin{cases} q_j & \text{if } j \notin S, \\ r_j & \text{if } j \in S. \end{cases}$$

Here $r_j$ is the result of executing $\texttt{command}_{k(j)}$, where $k(j)$ is the smallest index $\ell$ such that $\texttt{guard}_{\ell}$ holds for $q_{j-1}$ and $q_j$. More precisely: $r_j = h_{k(j)}^{\omega(j)}(q_{j-1}, q_j)$ with probability $p_{k(j)}^{\omega(j)}$ if $\texttt{command}_{k(j)}$ is probabilistic; $r_j = h_{k(j)}(q_{j-1}, q_j)$ with probability 1 otherwise. Such a transition is written $x \xrightarrow{S} x'$, or sometimes more simply $x \rightarrow x'$. The probability associated to this transition, written $p(x \xrightarrow{S} x')$, is always positive. More precisely, $p(x \xrightarrow{S} x') = \prod_{j \in S^*} p_{k(j)}^{\omega(j)}$ where $S^*$ is the subset of indices $j$ of $S$ such that $\texttt{command}_{k(j)}$ is probabilistic. Applying $\rightarrow$ $\ell$ times is written $\rightarrow^{\ell}$. The reflexive transitive closure of $\rightarrow$ is denoted $\rightarrow^*$. If the scheduler $\mathcal{A}$ always selects exactly one enabled process (i.e., $|S|$ is always equal to 1), then it is a *central scheduler*. Otherwise, it is a *distributed scheduler*.

*Example 1.* In Herman's mutual exclusion algorithm [10], the scheduler $\mathcal{A}$ is distributed and "maximal": at each step of computation $x \xrightarrow{S} y$, the set $S$ of selected processes is exactly the set $\mathcal{E}(x)$ of *all* the enabled processes.

The set of states is $Q = \{0, 1\}$, and the number of processes is odd. The expression $\overline{q}$ means $q + 1$ where $+$ is addition modulo 2. For each word $u =$

[1] One can also model similarly systems where $P_i$ observe not only the state of $P_{i-1}$ but also that of $P_{i+1}$, see Israeli-Jalfon's algorithm (example 11).

$q_1 q_2 \cdots q_k \in Q^k$ the expression $\overline{u}$ denotes $\overline{q}_1 \overline{q}_2 \cdots \overline{q}_k$. The transition system $\rightarrow$ is defined by:

IF $q_i \neq q_{i-1}$ THEN $q'_i = \overline{q}_i$
IF $q_i = q_{i-1}$ THEN $q'_i = \overline{q}_i$ with probability $1/2$
or $q_i$ with probability $1/2$.

For every configuration $x$, all the letter positions are enabled. When a process $P_j$ has a state $a$ equal to the state of its left neighbour, it is enabled for a probabilistic action, and $j \in S^*$. For example, consider the configuration $x = \mathbf{0111}01010$. $S^*$ is here the set of indices of the 3 letters in bold. We have $x \rightarrow x'$ with probability $(1/2)^4$, where $x'$ is the "opposite" string $\overline{x} = 100010101$.

Without loss of understanding, we will abbreviate henceforth $R : (N, \rightarrow, Q)$ as $\rightarrow$. We now arbitrarily fix a configuration $x_0$ as the initial configuration and a scheduler $\mathcal{A}$. A computation of $\rightarrow$ under $\mathcal{A}$ is a (possibly infinite) sequence of configurations $x_0, x_1, \cdots$, such that $x_0$ is the initial configuration and $x_i \xrightarrow{S_i} x_{i+1}$ for all $i \geq 0$, where $S_i$ is the set of processes selected by $\mathcal{A}$ at $i$-th step. Such a computation of $\rightarrow$ under $\mathcal{A}$ is not deterministic because of the existence of probabilistic actions. The *computation tree* associated with $\rightarrow$ under $\mathcal{A}$ starting from $x_0$, denoted by $T(\mathcal{A}, x_0)$, is a rooted tree such that:

1. $x_0$ is the root,
2. every directed path starting from the root corresponds to a possible computation of $\rightarrow$ under $\mathcal{A}$, and
3. every vertex $v$ is labeled with the probability $\pi(v)$ that $\rightarrow$ under $\mathcal{A}$ follows the path connecting $x_0$ to $v$. (If the path connecting $x_0$ to $v$ is of the form $x_0 \xrightarrow{S_0} x_1 \xrightarrow{S_1} \cdots \xrightarrow{S_{k-1}} x_k$, then $\pi(v) = \prod_{j=0}^{k-1} p(x_j \xrightarrow{S_j} x_{j+1})$.)

### 2.2 $\varphi$-algorithms

We consider now $\varphi$-algorithms, i.e., distributed algorithms for which there exists a measure $\varphi$ over configurations that never increases, whatever the randomized actions do. This situation is typical of mutual exclusion algorithms, where $\varphi$ counts the number of tokens ($\varphi$ never increases as tokens are never created during algorithm computations, and decrease when tokens collide.) This is also the case of Israeli-Jalfon *directing* protocol (see [12], Section 4.1) where $\varphi$ is the number of non-directed edges, or Dolev-Israeli-Moran *leader election* protocol (see [7], Section 5.3) where $\varphi$ is the number of processes that hold 1 in their leader variables. Formally, a measure $\varphi$ from $X$ to $\mathbb{N}$ is *non-increasing* iff: $x \xrightarrow{S} y$ implies $\varphi(y) \leq \varphi(x)$, for all $x, y \in X$ and every subset $S$ of enabled processes.

For a $\varphi$-algorithm, we define the set $\mathcal{L} \subseteq X$ of *$\varphi$-legitimate configurations* as: $\mathcal{L} = \{x \in X \mid \varphi(x) \leq c\}$ where $c$ is an integer constant. Since $\varphi$ is non-increasing, it is easy to show that $\mathcal{L}$ is *closed*, i.e: For any $\varphi$-legitimate configuration $x \in \mathcal{L}$, any configuration $y \in X$ and any set $S$ of enabled processes, $x \xrightarrow{S} y$ implies $y \in \mathcal{L}$.

We assume furthermore that there is *no deadlock* ($\forall x \in X \quad \mathcal{E}(x) \neq \emptyset$). In practice, for mutual exclusion algorithms, the no deadlock property often means that there exists always at least one token ($\forall x \ \varphi(x) \geq 1$). The set $\mathcal{L}$ of $\varphi$-legitimate configurations then actually corresponds to configurations with exactly one token (i.e.: $x \in \mathcal{L}$ iff $\varphi(x) = c = 1$).

Given a scheduler $\mathcal{A}$, we are interested in proving the following *convergence* property (see [13,3]): No matter which initial configuration one starts from, the probability that $\rightarrow$ under $\mathcal{A}$ reaches a $\varphi$-legitimate configuration in a finite number of transitions is 1. Formally, let $T^{\mathcal{L}}$ be the tree constructed from $T(\mathcal{A}, x_0)$ by cutting the edges going out of vertices corresponding to $\varphi$-legitimate configurations of $\mathcal{L}$; let $Leaf(\mathcal{L})$ be the set of leaves of $T^{\mathcal{L}}$. The convergence condition claims that for all $x_0 \in X$: $\sum_{v \in Leaf(\mathcal{L})} \pi(v) = 1$.
Expression $\sum_{v \in Leaf(\mathcal{L})} \pi(v)$ is the *probability of convergence of* $\rightarrow$ *under* $\mathcal{A}$, i.e., the probability, starting from $x_0$, to reach a $\varphi$-legitimate configuration in a finite number of transitions. It will be abbreviated as: $Pr(x_0 \xrightarrow{\mathcal{A}}{}^{*} \mathcal{L})$, or sometimes more simply as: $Pr(x_0 \rightarrow^{*} \mathcal{L})$.

## 3 Randomized Algorithms As Markov Chains

We assume in this section that scheduler $\mathcal{A}$ is given and is *memoryless*, i.e: for every sequence of transitions $x_0 \xrightarrow{S_0} x_1 \xrightarrow{S_1} \cdots \xrightarrow{S_{\ell-1}} x_\ell$, $\mathcal{A}$ deterministically selects a subset $S_\ell$ of enabled processes of $x_\ell$, which depends on $x_\ell$ only (not on the previous transitions). Since there is no deadlock, $S_\ell$ is nonempty, and the modification of letters at positions of $S_\ell$ randomly changes $x_\ell$ into a set of possible configurations $x^1_{\ell+1}, x^2_{\ell+1}, \cdots$, with associated probabilities $p(x_\ell \rightarrow x^1_{\ell+1}), p(x_\ell \rightarrow x^2_{\ell+1}), \cdots$ of sum 1. More precisely, given $x \in X$, consider the set $I_x$ of all the couples $(y, p)$ of $X \times [0, 1]$ such that $x \rightarrow y$ under $\mathcal{A}$ with probability $p$. Then, for all $x \in X$, $\Sigma_{(y,p) \in I_x} \ p = 1$. So the computation behaves exactly as a Markov chain [14]. Therefore the classical Markov property holds: whatever the initial configuration we start from, the probability to reach a *recurrent* configuration in a finite number of steps is 1. We exploit this result for proving the convergence of $\varphi$-algorithms towards $\mathcal{L}$, using the fact that, under a certain condition, each recurrent configuration is $\varphi$-legitimate. Let us first recall the notions of "recurrence", "transience" and Markov basic theorem in our context.

**Definition 2.** A configuration $x$ is *transient* iff $\exists y \quad x \rightarrow^{*} y \wedge \neg(y \rightarrow^{*} x)$.

**Definition 3.** A configuration $x$ is *recurrent* iff $x$ is non transient, i.e:
$\forall y \quad x \rightarrow^{*} y \Rightarrow y \rightarrow^{*} x$.
The set of recurrent configurations is denoted $\mathcal{R}ec$.

Correspondingly, we define $T^{\mathcal{R}ec}$ as a tree constructed from $T(\mathcal{A}, x_0)$ by cutting the edges going out from vertices corresponding to recurrent configurations of $\mathcal{R}ec$. Let $Leaf(\mathcal{R}ec)$ be the set of leaves of $T^{\mathcal{R}ec}$. The expression

$\sum_{v\in Leaf(\mathcal{R}ec)} \pi(v)$ is the probability, starting from $x_0$, to reach a recurrent configuration in a finite number of transitions. It will be abbreviated as
$Pr(x_0 \xrightarrow{\mathcal{A}}{}^{*} \mathcal{R}ec)$.

**Theorem 4. (Markov)** *Given a scheduler $\mathcal{A}$, we have: for every configuration $x$, $Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{R}ec) = 1$.*

We now assume given a non-increasing measure $\varphi$ on $X$, and exploit Markov's theorem.

**Definition 5.** A configuration $x$ is *$\varphi$-transient* iff $\exists y\ x \to^{*} y \wedge \varphi(y) < \varphi(x)$.

**Lemma 6.** *For every configuration $x$, if $x$ is $\varphi$-transient, then $x$ is transient.*

*Proof.* Suppose that $x$ is $\varphi$-transient. Therefore $x \to^{*} y \wedge \varphi(y) < \varphi(x)$ for some $y$. Let us show that $x$ is transient by proving by reductio ad absurdum that $\neg(y \to^{*} x)$. If $y \to^{*} x$, then $\varphi(x) \leq \varphi(y)$ (since $\varphi$ is non-increasing), which contradicts $\varphi(y) < \varphi(x)$. □

**Theorem 7.** *Given a scheduler $\mathcal{A}$ and a non-increasing measure $\varphi$, we have: if each configuration $x$ with $\varphi(x) > c$ is $\varphi$-transient, then: $\forall x\ \ Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{L}) = 1$.*

*Proof.* Any non $\varphi$-legitimate configuration $x$ is such that $\varphi(x) > c$. So, by assumption, any non $\varphi$-legitimate configuration $x$ is $\varphi$-transient. Hence, by Lemma 6, $x$ is transient, i.e., non recurrent. So: $\neg\mathcal{L} \subseteq \neg\mathcal{R}ec$. Hence: $\mathcal{R}ec \subseteq \mathcal{L}$. Now, by Markov theorem, $Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{R}ec) = 1$. It follows: $Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{L}) = 1$. □

Let us now give a local condition, called "Prop", ensuring that every non-legitimate configuration is $\varphi$-transient. This condition is local in the sense that it involves only one-step reduction $\to$ (instead of $\to^{*}$).

We assume given a scheduler $\mathcal{A}$ and a non-increasing measure $\varphi$. For every value of ring length $N$, we assume given a measure $D$ from $X = Q^N$ to a finite set $\Delta$ and an ordering $\ll$ over $\Delta$. We then define a binary relation $\lhd$ over $X$ as follows:

$$\forall x, y \in X \quad x \lhd y \quad \Leftrightarrow \quad \varphi(x) < \varphi(y) \vee D(x) \ll D(y).$$

Since $<$ and $\ll$ are orderings, $\lhd$ is itself an ordering over the finite set $X = Q^N$. Local condition Prop is defined as:

$$\forall x \quad (\varphi(x) > c \;\Rightarrow\; \exists y \quad (x \to y \;\wedge\; x \lhd y)).$$

It says that, for each configuration $x$ (of $\varphi$-measure $> c$), there exists a transition going from $x$ to a configuration $y$ smaller w.r.t. $\lhd$. As there is no infinite decreasing sequence for $\lhd$ (since $\Delta$ is finite), Prop ensures the $\varphi$-transience of each non-legitimate configuration. Therefore by Theorem 7 it follows:

**Theorem 8.** *Given a scheduler $\mathcal{A}$ and a non-increasing measure $\varphi$, we have: if, for all $N$, there exist a measure $D$ and an ordering $\ll$ such that Prop:*

$$\forall x \quad (\varphi(x) > c \;\Rightarrow\; \exists y \quad (x \to y \;\wedge\; (\varphi(x) < \varphi(y) \vee D(x) \ll D(y))))$$

*holds, then: $\forall x\ \ Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{L}) = 1$.*

Given $\mathcal{A}$ and $\varphi$, the crucial part of the convergence proof consists now in finding appropriate $D$ and $\ll$ that satisfy Prop. In practice, $D$ and $\ll$ will not be defined specifically for each ring length $N$, but in a generic manner with $N$ as a parameter.

*Example 9.* We apply here Theorem 8 for showing that Herman's algorithm is convergent. It is easy to notice (see [10]) that, starting from a configuration with an odd number of processes, $\rightarrow$ has no deadlock. Given a configuration $x = q_0 q_1 \cdots q_{N-1}$ we say there is a token at process $i$ if $q_i = q_{i-1}$. The measure $\varphi$ counts the number of tokens of a configuration $x$, i.e: $\varphi(x) = card(\{i \in \{0, \cdots, N-1\} \mid q_i = q_{i-1}\})$. It is also proved in [10] that $\varphi$ is non-increasing. The set $\mathcal{L}$ of $\varphi$-legitimate configurations is defined as the set of configurations $x$ with at most one token ($\varphi(x) \leq c = 1$).

For configurations $x$ with at least two tokens, we consider the same measure $D$ as the one introduced by Herman [10], i.e. the minimal distance between two consecutive tokens of $x$. The corresponding ordering $\ll$ then coincides with $<$. For example, for $x = \mathbf{0}1010\mathbf{0}101\mathbf{1}01010$, there are three tokens (represented in bold font), and the minimal distance between them is $D(x) = 4$. In the following, we assume that $u$ and $v$ are strings of $Q^*$, $a$ and $b$ are distinct elements of $Q$. If $x$ has a pair of adjacent tokens (i.e., $x$ is of the form $ua\mathbf{a}\mathbf{a}v$), then $D(x) = 1$. Otherwise, for some $n \geq 0$, $x$ may be of the form $ua\mathbf{a}(ba)^{n-1}ba\mathbf{a}v$ with $D(x) = 2n+1$ (odd case) or of the form $ua\mathbf{a}(ba)^n b\mathbf{b}v$, thus $D(x) = 2n+2$ (even case).

Let us show that, for all $x$ with at least two tokens ($\varphi(x) > 1$), there exists $y$ such that $x \rightarrow y$ with $\varphi(y) < \varphi(x)$ or $D(y) < D(x)$. There are three cases:

- If $D(x) = 1$, $x$ is of the form $uaaav$ and $x \rightarrow y = \overline{u}bab\overline{v}$ with $\varphi(y) = \varphi(x) - 2$.
- If $D(x) = 2n+1$ with $n \geq 0$, $x$ is of the form $uaa(ba)^{n-1}baav$ and $x \rightarrow y = \overline{u}ba(ab)^{n-1}abb\overline{v}$ with $D(y) = 2n < D(x)$.
- If $D(x) = 2n+2$ with $n \geq 0$, $x$ is of the form $uaa(ba)^n bbv$ and $x \rightarrow y = \overline{u}ba(ab)^n aa\overline{v}$ with $D(y) = 2n+1 < D(x)$.

By Theorem 8, it follows that: $\forall x \; Pr(x \rightarrow^* \mathcal{L}) = 1$. This proof is much simpler than the original one of [10]. Many other examples can be treated exactly along these lines (e.g., Beauquier-Delaët [2], Flatebo-Datta [9]).

## 4 Randomized Algorithms Under Arbitrary Scheduler

Let us consider now the case where the scheduler $\mathcal{A}$ is *not* memoryless, but has unlimited resources, and chooses the next enabled processes using the full information of the execution so far. In such a case, for the same configuration $x$, the selection of enabled processes can vary with the different sequences of transitions that led to $x$. The computation under such a scheduler $\mathcal{A}$ does not behave any longer as a Markov chain. Property Prop of Theorem 8 (dependent on a specific scheduler) must be strengthened in order to take into account all possible schedulers: see Prop' below. For the sake of simplicity, we focus on the case where the scheduler is central (the selected subset of enabled processes is a singleton). The counterpart of Theorem 8 is the following:

**Theorem 10.** *Given a non-increasing measure $\varphi$, suppose that there exist $D$ and $\ll$ such that:*
Prop': $\forall x\ \forall i \in \mathcal{E}(x) \quad (\varphi(x) > c\ \Rightarrow\ \exists y\ (x \xrightarrow{i} y \wedge (\varphi(x) < \varphi(y) \vee D(x) \ll D(y))))$
*Then, for any central scheduler $\mathcal{A}$:* $\forall x \quad Pr(x \xrightarrow{\mathcal{A}}{}^* \mathcal{L}) = 1.$

The proof is given in appendix A. The result extends to the distributed case in the natural way (by replacing $\mathcal{E}(x)$ with $2^{\mathcal{E}(x)}$). Theorem 10 can be seen as a restricted version of Theorem 1 of [3] (see also theorem 5 of [7]). In [8], we prove the convergence of Kakugawa-Yamashita's algorithm [13] using Theorem 10. For lack of space, we present below an application to a simpler algorithm.

*Example 11.* Let us consider Israeli-Jalfon's algorithm [11]. The scheduler is central and arbitrary. A minor difference with the framework presented in Section 2.1 is that each guard takes into account not only the state $q_{i-1}$ of the left neighbour of $P_i$, but also the state $q_{i+1}$ of the right neighbour. Also each command modifies not only the state $q_i$ of $P_i$, but also the states of the left and right neighbours. (There is no conflict between transitions because the scheduler is central, and only one enabled process $P_i$ is selected at each step.) The set of states is $Q = \{0, 1\}$. The transition algorithm $\rightarrow$ is:

IF $q_{i-1}q_iq_{i+1} = 111$ THEN $q'_{i-1}q'_iq'_{i+1} = 101$
IF $q_{i-1}q_iq_{i+1} = 011$ THEN $q'_{i-1}q'_iq'_{i+1} = 101$ with probability 1/2
or 001 with probability 1/2.
IF $q_{i-1}q_iq_{i+1} = 110$ THEN $q'_{i-1}q'_iq'_{i+1} = 101$ with probability 1/2
or 100 with probability 1/2.
IF $q_{i-1}q_iq_{i+1} = 010$ THEN $q'_{i-1}q'_iq'_{i+1} = 100$ with probability 1/2
or 001 with probability 1/2.

Given a configuration $x$ we say there is a token at process $i$ if $q_i = 1$. The measure $\varphi$ counts the number of tokens of a configuration $x$. We focus on initial configurations $x$ with at least one token ($\varphi(x) > 0$). Then all the subsequent configurations keep always at least one token and $\rightarrow$ has no deadlock. It is also obvious that $\varphi$ is non-increasing (no '1' is created). The set $\mathcal{L}$ of $\varphi$-legitimate configurations is defined as the set of configurations $x$ with at most one token ($\varphi(x) \leq c = 1$).

Given a configuration with a fixed number of tokens, say $k > 1$, we consider the measure $D$ that maps any configuration $x$ with the $k$-tuple of distances between two tokens ordered by increasing order. For example, for $x =$ 000110001010, $k = 4$ and $D(x) = (1, 2, 4, 5)$.

Let us show that, for all $x$ with at least two tokens ($\varphi(x) > 1$), and every position $i$ of token (or '1') in $x$, there exists $y$ such that $x \xrightarrow{i} y$ with $\varphi(y) < \varphi(x)$ or $D(y) \ll D(x)$, where $\ll$ is the lexicographic order. There are four cases:

- If $q_{i-1}q_iq_{i+1} = 111$, then $x$ is of the form $u111v$ and $x \xrightarrow{i} y = u101v$ with $\varphi(y) = \varphi(x) - 1$.

– If $q_{i-1}q_iq_{i+1} = 011$, then $x$ is of the form $u011v$ and $x \xrightarrow{i} y = u001v$ (with probability 1/2) with $\varphi(y) = \varphi(x) - 1$.
– If $q_{i-1}q_iq_{i+1} = 110$, then $x$ is of the form $u110v$ and $x \xrightarrow{i} y = u100v$ (with probability 1/2) with $\varphi(y) = \varphi(x) - 1$.
– If $q_{i-1}q_iq_{i+1} = 010$, then $x$ is of the form $u010v$. Let $\ell$ (resp. $r$) be the distance between the token at position $i$ and the closest token at his left (resp. right). Such a left (resp. right) closest token exists because $\varphi(x) > 1$. (Note that the left and right closest tokens coincide if $\varphi(x) = 2$.) If $\ell \leq r$ then $x \xrightarrow{i} y = u100v$ (with probability 1/2) with $D(y) \ll D(x)$. Otherwise, $x \xrightarrow{i} y = u001v$ (with probability 1/2) with $D(y) \ll D(x)$.

By Theorem 10, it follows that, for any scheduler $\mathcal{A}$: $\forall x \; Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{L}) = 1$.

## 5 Expected Time of Convergence by *D*-Lumping

We assume again in this section that scheduler $\mathcal{A}$ is given and is *memoryless*. So the computation via $\rightarrow$ under $\mathcal{A}$ behaves exactly as a Markov chain [14]. Let $k$ be an integer greater than $c$, $X_k$ the set of configurations of $\varphi$-measure $k$, and $X_{<k}$ the set of configurations of $\varphi$-measure less than $k$. Let $X_k^d$ be the subset of configurations of $X_k$ of $D$-measure $d$, and $\Delta_k$ the image of $X_k$ via $D$. We have:

$X_k = \{x \in X \mid \varphi(x) = k\}$.
$X_{<k} = \{x \in X \mid \varphi(x) < k\}$.
$X_k^d = \{x \in X \mid \varphi(x) = k \wedge D(x) = d\}$.
$\Delta_k = \{d \in \Delta \mid \exists x \in X_k \; D(x) = d\}$.

Note that $X_{<k}$ is closed under $\rightarrow$ (since $\varphi$ is non-increasing). We assume:

$\forall x \in X_k \; \exists y \in X_{<k} \quad x \rightarrow^* y$.

This means that for every configuration $x$ with $k$ tokens ($k > c$), there exists a computation that goes from $x$ to a configuration with less than $k$ tokens. Condition Prop of Theorem 8 is a sufficient condition that guarantees the existence of such a computation. We want now to get quantitative information about the expected time of convergence of $\rightarrow$ under $\mathcal{A}$ from $x_0 \in X_k$ to $X_{<k}$. Formally, given $x_0 \in X_k$, let $T^{X_{<k}}$ be the tree obtained from $T(\mathcal{A}, x_0)$ by cutting the edges going out from vertices corresponding to configurations of $X_{<k}$. Let $Leaf(X_{<k})$ be the set of leaves of $T^{X_{<k}}$. We are interested in computing (an upper bound for) the expected time of $x_0$ to reach $X_{<k}$, that is $\sum_{v \in Leaf(X_{<k})} \delta(v)\pi(v)$ where $\delta(v)$ and $\pi(v)$ are the depth and probability of $v$ in $T^{X_{<k}}$ respectively. This will be abbreviated as $E(x_0 \xrightarrow{\mathcal{A}}{}^{*} X_{<k})$, or more simply as: $E(x_0 \rightarrow^* X_{<k})$. We now explain how to compute this quantity, under certain conditions, by "lumping" using measure $D$. In the Markov theory, it is indeed common to lump together configurations, in order to get a smaller Markov chain which gives information about the original chain. Given $x \in X_k$ and $e \in \Delta_k$, consider the expression:

$\xi(x, e) = \sum_{y \in X_k^e} p(x \rightarrow y)$.

This is the probability of moving via $\rightarrow$ under $\mathcal{A}$ from an element $x$ of $X_k$ into the set $X_k^e$ of configurations of $D$-measure $e$. Likewise, consider the expression:

$\xi(x,\perp) = \sum_{y\in X_{<k}} p(x \rightarrow y)$,
where $\perp$ is a new symbol. This is the probability of moving via $\rightarrow$ under $\mathcal{A}$ from an element $x$ of $X_k$ into set $X_{<k}$ of configurations of $\varphi$-measure less than $k$. Finally, let:
$\xi(\perp,\perp) = 1$.
This expresses the fact that $X_{<k}$ is a set which once entered is never left. We say that Markov chain $\rightarrow$ is *$D$-lumpable* if, for all $x \in X_k$ and all $e \in \Delta_k \cup \{\perp\}$, probability $\xi(x,e)$ depends only on the $D$-measure $d$ of $x$, i.e:
$\forall d \in \Delta_k\ \forall e \in \Delta_k \cup \{\perp\}\ \forall x, x' \in X_k^d \quad \xi(x,e) = \xi(x',e)$.
We then write such a probability $\xi(d,e)$. Given a $D$-lumpable transition system $\rightarrow$, the associated *$D$-transition system*, written $\rightsquigarrow$, is the binary relation over $\Delta_k \cup \{\perp\}$ defined as follows:

- for all $d, e \in \Delta_k$, $d \rightsquigarrow e$ with probability $\xi(d,e)$.
- for all $d \in \Delta_k$, $d \rightsquigarrow \perp$ with probability $\xi(d,\perp)$,
- $\perp \rightsquigarrow \perp$ with probability $\xi(\perp,\perp) = 1$.

By Markov theory, if $\rightarrow$ is a lumpable Markov chain, then the lumped transition system $\rightsquigarrow$ is also a Markov chain. The *$D$-computation tree* associated with $\rightsquigarrow$ starting from $d_0 \in \Delta_k \cup \{\perp\}$, denoted by $U(\mathcal{A}, d_0)$, is a rooted tree such that:

1. $d_0$ is the root,
2. every directed path starting from the root corresponds to a possible sequence of transitions via $\rightsquigarrow$, and
3. every vertex $w$ is labeled with the probability $\psi(w)$ corresponding to the path connecting $d_0$ to $w$. (If the path connecting $d_0$ to $w$ is of the form $d_0 \rightsquigarrow d_1 \rightsquigarrow \cdots \rightsquigarrow d_\ell$, then $\psi(w) = \prod_{j=0}^{\ell-1} \xi(d_j, d_{j+1})$.)

Let $U^\perp$ be the tree constructed from $U(\mathcal{A}, d_0)$ by cutting the edges going out from vertices corresponding to $\perp$. Let $Leaf(\perp)$ be the set of leaves of $U^\perp$. Let $\psi(w)$ and $\varepsilon(w)$ be the probability and the depth of $w$ respectively, for every vertex $w$ in $U^\perp$. The expression $\sum_{w\in Leaf(\perp)} \varepsilon(w)\psi(w)$ is the expected number of $D$-transitions, starting from $d_0$, to reach $\perp$. It will be abbreviated as: $E_k(d_0 \rightsquigarrow^* \perp)$. We now explain, by Markov theory, how to compute this quantity and how it relates to $E(x_0 \rightarrow^* X_{<k})$.

The *$D$-transition matrix* is the square matrix of size $(|\Delta_k|+1)$ having $\xi(d,e)$ on the row and column corresponding to $d$ and $e$ respectively, for all $d, e \in \Delta_k \cup \{\perp\}$.

**Definition 12.** An *absorbing element* $\alpha$ is an element such that, for all $d \in \Delta_k \cup \{\perp\}$, $\alpha \rightsquigarrow^* d \Rightarrow d = \alpha$. The set of absorbing elements is denoted by $\mathcal{A}bs$. A Markov chain is *absorbing* iff all recurrent elements are absorbing and conversely ($\mathcal{R}ec = \mathcal{A}bs$).

Note that, by definition, all absorbing elements are recurrent ($\mathcal{A}bs \subseteq \mathcal{R}ec$). Furthermore, in our case, it is clear that $\perp \in \mathcal{A}bs$ (since $\perp \rightsquigarrow \perp$ with probability

1). Besides, from the assumption that: $\forall x \in X_k \; \exists y \in X_{<k} \;\; x \rightarrow^* y$, it follows by lumpability that: $\forall d \in \Delta_k \;\; d \rightsquigarrow^* \perp$ (convergence towards $\perp$). Markov theory states the following (see [14], p.59) :

**Theorem 13. (Markov2)** *For every absorbing Markov chain, we have:* $E_k(d \rightsquigarrow^* \mathcal{A}bs) < \infty$. *Furthermore,* $(E_k(d \rightsquigarrow^* \mathcal{A}bs))_{d\in\Delta_k} = (\mathcal{I} - \mathcal{Q}_k)^{-1}\mathbf{1}$, *where* $\mathcal{Q}_k$ *is the matrix obtained by truncating the* $D$*-transition matrix onto the set of non-absorbing elements,* $\mathcal{I}$ *the identity matrix of size* $|\Delta_k|$, *and* $\mathbf{1}$ *is the column matrix made of* $|\Delta_k|$ *elements 1.*

Besides, we have:

**Lemma 14.** *For the* $D$*-transition system,* $\mathcal{A}bs = \mathcal{R}ec = \{\perp\}$. *So the* $D$*-transition system is an absorbing Markov chain.*

*Proof.* We have always $\mathcal{A}bs \subseteq \mathcal{R}ec$. Let us prove $\mathcal{R}ec \subseteq \mathcal{A}bs$ by showing that if $d \notin \mathcal{A}bs$, then $d \notin \mathcal{R}ec$. Suppose $d \notin \mathcal{A}bs$. Then $d \neq \perp$. Hence $\neg(\perp \rightsquigarrow^* d)$. On the other hand, by convergence towards $\perp$, we have: $d \rightsquigarrow^* \perp$. Since $d \rightsquigarrow^* \perp$ and $\neg(\perp \rightsquigarrow^* d)$, $d \notin \mathcal{R}ec$. □

From Theorem 13 and Lemma 14, it follows:

**Corollary 15.** *For all* $d \in \Delta_k$, $E_k(d \rightsquigarrow^* \perp) < \infty$. *Furthermore,* $(E_k(d \rightsquigarrow^* \perp))_{d\in\Delta_k} = (\mathcal{I} - \mathcal{Q}_k)^{-1}\mathbf{1}$, *where* $\mathcal{Q}_k$ *is the matrix obtained by truncating the* $D$*-transition matrix onto* $\Delta_k$ *(i.e, removing the* $\perp$*-row and* $\perp$*-column).*

In addition, since the original Markov chain $\rightarrow$ is $D$-lumpable, Markov theory says that the lumped $D$-transition system $\rightsquigarrow$ is such that (see [14]):
$\forall d \in \Delta_k \; \forall x \in X_k^d \quad E(x \rightarrow^* X_{<k}) = E_k(d \rightsquigarrow^* \perp)$.
It is interesting to compute $E_k(d \rightsquigarrow^* \perp)$ rather than $E(x \rightarrow^* X_{<k})$ because the transition matrix of the lumped Markov chain is much smaller than the matrix of the original chain. For example, in Herman's example in case $k = 2$, the matrix of the lumped chain is of size $\lfloor N/2 \rfloor$ while the original one is of size $2^N$. The computation of $E_k(d \rightsquigarrow^* \perp)$ in Herman's example is explained below.

*Example 16.* Let us first show that the Markov chain corresponding to Herman's algorithm is $D$-lumpable for $k = 2$. This is due to the invariance by rotation of the minimal distance of configurations. Formally let $x$ be a given configuration with 2 tokens of minimal distance $D(x) = d$. Let $i$ be the position of the first token, and $j$ the position of the second one. We write $x = (i, j)$. The distance $d$ is $min(i - j, j - i)$ where $-$ is subtraction modulo $N$, and ranges over $\Delta_2 = \{1, 2, \cdots, \lfloor N/2 \rfloor\}$ (as $N$ is odd, $\lfloor N/2 \rfloor = (N-1)/2$). For $1 < d < \lfloor N/2 \rfloor$, $x$ moves either to $x_0 = (i, j), x_1 = (i+1, j+1), x_2 = (i+1, j)$ or $x_3 = (i, j+1)$ with equal probability 1/4. Therefore, $\xi(x, d) = 1/2$ $(= p(x \rightarrow x_0) + p(x \rightarrow x_1))$, and $\xi(x, d+1) = \xi(x, d-1) = 1/4$. If $d = 1$, then $\xi(x, 1) = 1/2$, $\xi(x, 2) = 1/4$ and $\xi(x, \perp) = 1/4$ (case where the 2 tokens collide). If $d = (N-1)/2$, $\xi(x, d) = 3/4$ and $\xi(x, d-1) = 1/4$. Given $d$ and $e$, the value of each probability $\xi(x, e)$ is constant, whatever the choice of $x \in X_2^d$. Hence the lumpability property.

Let us now explain the computation of the $D$-transition matrix $\mathcal{Q}_k$ for $k = 2$ in Herman's example [10]. $\Delta_2 = \{1, 2, \cdots, m\}$ with $m = N/2$. $Q_2$ is the $m \times m$ matrix of components $\xi(d, e)$ (with $d, e \in \Delta_2$), of the form:

$$\begin{pmatrix} 1/2 & 1/4 & & & & \\ 1/4 & 1/2 & 1/4 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \\ & & & 1/4 & 1/2 & 1/4 \\ & & & & 1/4 & 3/4 \end{pmatrix}$$

Note that, component $\xi(1, \perp) = 1/4$ is excluded from the matrix $Q_2$, since the $\perp$-column has been truncated. We then compute $B_2 = (\mathcal{I} - \mathcal{Q}_2)^{-1}$, which gives:

$$\begin{pmatrix} 4 & 4 & . & . & . & . & 4 \\ 4 & 8 & 8 & . & . & . & 8 \\ . & 8 & 12 & . & . & . & 12 \\ . & . & . & . & & . & . \\ . & . & . & & . & & . \\ . & . & . & & & . & . \\ 4 & 8 & 12 & . & . & . & 4m \end{pmatrix}$$

By Corollary 15, we know that the result of applying $B_2$ to $\mathbf{1}$ gives a column vector of $d$-component $2d(N - d)$, for $d \in \{1, \cdots, \lfloor N/2 \rfloor\}$. Therefore $E_2(d \rightsquigarrow^* \perp) = 2d(N - d)$. The maximal expected time corresponds to $d = \lfloor N/2 \rfloor = m$, and is $2m(m + 1) \simeq N^2/2$. This corresponds to $E(x \rightarrow^* \mathcal{L})$ for $x \in X_2$. We thus retrieves directly what Herman obtained in [10] in a more complicated way. Using this result, Herman explains how to infer an expected time $N^2\lceil \log N \rceil/2$ for $E(x \rightarrow^* \mathcal{L})$ in the general case where $x \in X_k$ with $k > 2$ (see [10]). Israeli-Jalfon's algorithm is analyzed similarly in appendix B.

## 6 Final Remarks And Further Work

We exploited Markov chains theory in order to simplify the proofs of convergence of randomized self-stabilizing algorithms and justify more formally their performance analysis. Our method relies on the existence of a non-increasing measure $\varphi$ over the configurations of the distributed system. Classically, this measure counts the number of tokens of configurations. It also exploits a function $D$ that expresses some distance between tokens, for a fixed number $k$ of tokens. Our first result was to exhibit a sufficient condition Prop that exploits $\varphi$ and $D$ in order to guarantee that, under a memoryless scheduler, every non-legitimate configuration is "transient" in the sense of Markov theory. Roughly speaking, Prop says that, under a given scheduler $\mathcal{A}$, for every configuration $x$, there exists a transition of non-null probability that applies to $x$ and decreases $\varphi$ or $D$. We extended this property Prop in order to handle arbitrary schedulers although they may induce non Markov chain behaviours. We thus retrieve (a particular case of) a result due to [3]. We then explain how Markov's notion of lumping naturally applies to measure $D$, and allows to analyze the expected time of convergence of self-stabilizing algorithms. Let us point out that the crucial step for proving convergence of such distributed algorithms is still the discovery of an appropriate function $D$ satisfying Prop or Prop' (which may be intricated, see e.g., example of Kakugawa-Yamashita in [8]), but our work identifies some weak properties of $D$ which suffice to entail the algorithm convergence and justify the performance analysis as done, e.g., by Herman. We only treated the case of finite-state reading model of communication as well as ring topologies. The finite-state assumption is a basic requirement of our work, which seems difficult to be relaxed. On the other hand, we believe that our results apply to other network topologies than rings. In the future, we plan to compare our lumping-based analysis method with the scheduler-luck game technique of [7].

**Acknowledgements.** We are most grateful to Joffroy Beauquier and anonymous referees for helpful comments on the earlier version of this paper.

## References

1. J. Beauquier, S. Cordier, and S. Delaët. Optimum probabilistic self-stabilization on uniform ring. In *Proc. of the 2nd Worshop on Self-Stabilizing Systems*, 1995.
2. J. Beauquier and S. Delaët. Probabilistic self-stabilizing mutual exclusion in uniform ring. In *Proc. of the 13th Annual ACM Symposium on Principles of Distributed Computing (PODC'94)*, page 378, 1994.
3. J. Beauquier, J. Durand-Lose, M. Gradinariu, and C. Johnen. Token based self-stabilizing uniform algorithms. *J. of Parallel and Distributed Systems*, To appear.
4. J. Beauquier, M. Gradinariu, and C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *Proc. of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*, pages 199–208, 1999.
5. E.W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, Nov. 1974.
6. S. Dolev. *Self-Stabilization*. MIT Press, 2000.
7. S. Dolev, A. Israeli, and S. Moran. Analyzing expected time by scheduler-luck games. *IEEE Transactions on Softare Engineering*, 21(5):429–439, May 1995.
8. M. Duflot, L. Fribourg, and C. Picaronny. Randomized distributed algorithms as markov chains. Technical report, Lab. Specification and Verification, ENS de Cachan, Cachan, France, May 2001. Available on `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/`.
9. M. Flatebo and A.K. Datta. Two-state self-stabilizing algorithms for token rings. *IEEE Transactions on Software Engineering*, 20(6):500–504, June 1994.
10. T. Herman. Probabilistic self-stabilization. *IPL*, 35(2):63–67, June 1990.
11. A. Israeli and M. Jalfon. Token management schemes and random walks yield self-stabilizing mutual exclusion. In *Proc. of the 9th Annual ACM Symposium on Principles of Distributed Computing (PODC'90)*, pages 119–131, 1990.
12. A. Israeli and M. Jalfon. Uniform self-stabilizing ring orientation. *Information and Computation*, 104(2):175–196, 1993.
13. H. Kakugawa and M. Yamashita. Uniform and self-stabilizing token rings allowing unfair daemon. *IEEE Trans. Parallel and Distributed Systems*, 8(2), 1997.
14. J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. D. van Nostrand Co., 1969.

## Appendix A: Proof of Theorem 10

Before proving this theorem, it is convenient to introduce the notion of "$\lhd$-decreasing sequence" and some related properties.

**Definition 17.** Given $x \in X$, a sequence $\sigma = (i_1, \cdots, i_\ell)$ of process indices is said to be $\lhd$*-decreasing for* $x$ iff:

$x = x_0 \xrightarrow{i_1} x_1 \xrightarrow{i_2} \cdots \xrightarrow{i_\ell} x_\ell$ for some $x_1, \cdots, x_\ell \in X$ such that

- $x_\ell \lhd x_{\ell-1} \lhd \cdots \lhd x_1 \lhd x_0$, or
- $x_i \lhd x_{i-1} \lhd \cdots \lhd x_1 \lhd x_0$ and $\varphi(x_i) \leq c$, for some $i$ $(0 \leq i < \ell)$.

For any integer $\ell$, the finite set of $\lhd$-decreasing sequences for $x$ of length $\ell$ is written $Dec_\ell(x)$.

We will write henceforth $x \xrightarrow{\sigma} \mathcal{L}$ with $\sigma = (i_1, \cdots, i_\ell)$ as an abbreviation of: $\exists x_1, \cdots, x_\ell \;\; x \xrightarrow{i_1} x_1 \wedge x_1 \xrightarrow{i_2} x_2 \wedge \cdots \wedge x_{\ell-1} \xrightarrow{i_\ell} x_\ell \in \mathcal{L}$.

**Lemma 18.** *Relation $\lhd$ is an ordering over $X = Q^N$. Furthermore, there exists $M > 0$ such that every sequence of configurations decreasing for $\lhd$ is of length $\leq M$.*

*Proof.* First, it is easy to see that $\lhd$ is an ordering because $<$ and $\ll$ are orderings. Let us now show that, for $m \geq M = |X|$, any sequence $x_0, \cdots, x_m$ of elements in $X$ cannot be ordered by $\lhd$. By the pigeonhole principle, there exists $i, j$ with $0 \leq i < j \leq m$ such that $x_i = x_j$. Hence, we cannot have $x_m \lhd \cdots x_j \lhd \cdots \lhd x_i \lhd \cdots \lhd x_0$. $\square$

**Lemma 19.** *For $M$ defined as above, we have: $\forall x \in X \; \forall \sigma \in Dec_M(x) \;\; x \xrightarrow{\sigma} \mathcal{L}$.*

*Proof.* By definition, given $x \in X$, for all $\sigma = (i_1, \cdots, i_M) \in Dec_M(x)$, there exist $x_1, \cdots, x_M$ such that $x = x_0 \xrightarrow{i_1} x_1 \xrightarrow{i_2} \cdots \xrightarrow{i_M} x_M$ with

$x_M \lhd x_{M-1} \lhd \cdots \lhd x_1 \lhd x_0$, or

$x_i \lhd x_{i-1} \lhd \cdots \lhd x_1 \lhd x_0$ and $\varphi(x_i) \leq c$, for some $i$ $(0 \leq i < M)$.

But the first case $x_M \lhd x_{M-1} \lhd \cdots \lhd x_1 \lhd x_0$ is impossible according to Lemma 18. So: $\varphi(x_i) \leq c$, for some $i$ $(0 \leq i < M)$. Hence $x \xrightarrow{\sigma} \mathcal{L}$. $\square$

**Proof of Theorem 10:**

Given $x \in X$ and $\sigma$ such that $x \xrightarrow{\sigma} \mathcal{L}$, let $Pr(x \xrightarrow{\sigma} \mathcal{L})$ be the probability associated with $x \xrightarrow{\sigma} \mathcal{L}$. Let us now show

$\mathrm{P}_1: \quad \exists p \in ]0,1] \; \forall x \in X \;\; \forall \sigma \in Dec_M(x) \quad Pr(x \xrightarrow{\sigma} \mathcal{L}) \geq p,$

By iteratively applying Prop', it is easy to show that, for all $x \in X$ and all $\ell \in \mathbb{N}^*$, there exists a $\lhd$-decreasing sequence $\sigma \in Dec_\ell(x)$. For all $x \in X$, we know by applying Lemma 19 to $x$ that, for all $\sigma \in Dec_M(x)$, $Pr(x \xrightarrow{\sigma} \mathcal{L}) > 0$. Since $Dec_M(x)$ is nonempty and finite, we can define $p_x$ as the minimum of $Pr(x \xrightarrow{\sigma} \mathcal{L})$ when $\sigma$ ranges over $Dec_M(x)$. We have: $\forall \sigma \in Dec_M(x) \quad Pr(x \xrightarrow{\sigma} \mathcal{L}) \geq p_x > 0$.

Now by taking the minimum $p$ of all the $p_x$ when $x$ ranges over $X$, we get: $\forall x \in X,\ \forall \sigma \in Dec_M(x)\ \ Pr(x \xrightarrow{\sigma} \mathcal{L}) \geq p > 0$. Hence $P_1$.

Observe that, by Prop', for every configuration $x$ and every scheduler $\mathcal{A}$, there is a path in the computation tree $T(\mathcal{A}, x)$ along which the probabilistic actions make the successive configurations decrease for $\lhd$. So from $P_1$ it follows that the probability, starting from $x$, to reach $\mathcal{L}$ under $\mathcal{A}$ in $M$ transitions is $\geq p$. This writes: $Pr(x \xrightarrow{\mathcal{A}}{}^{M} \mathcal{L}) \geq p$. Alternatively, given a scheduler $\mathcal{A}$ and a starting configuration $x$, the probability of not being in $\mathcal{L}$ is less than $1-p$ after the $M$ first transitions. It is less than $(1-p)^2$ after the $2M$ first transitions, and so on. The probability of not reaching $\mathcal{L}$ under $\mathcal{A}$ after $\ell$ transitions tends to 0 as $\ell$ tends to $\infty$. In other words: $\forall x \in X$, $lim_{\ell \to \infty}\ Pr(x \xrightarrow{\mathcal{A}}{}^{\ell} \mathcal{L}) = 1$. So, for any central scheduler $\mathcal{A}$: $\quad \forall x\ \ Pr(x \xrightarrow{\mathcal{A}}{}^{*} \mathcal{L}) = 1$. □

## Appendix B: Expected Time of Israeli-Jalfon's Algorithm

As mentioned earlier, the scheduler in Israeli-Jalfon's example is arbitrary and the algorithm does not behave *a priori* as a Markov chain. However in the case where there are only two tokens ($k = 2$), one can suppose that the scheduler is memoryless, and always selects the same token, say $A$, since any move of the other one can be simulated by a symmetrical move on $A$. It is also easy to show that Markov chain corresponding to Israeli-Jalfon's algorithm is $D$-lumpable for $k = 2$. The $D$-chain corresponds to a random walk, and the expected time $E_2(d \rightsquigarrow^* \perp)$ corresponds to the expected time that one token at a distance $d$ from the origin, animated by a random walk, meets that origin. For $N = 2m+1$, we have:

$$\mathcal{Q}_2 = \begin{pmatrix} 0 & 1/2 & & & & \\ 1/2 & 0 & 1/2 & & & \\ & . & . & . & & \\ & & . & . & . & \\ & & & . & . & \\ & & & 1/2 & 0 & 1/2 \\ & & & & 1/2 & 1/2 \end{pmatrix}$$

$$(\mathcal{I} - \mathcal{Q}_2)^{-1} = \begin{pmatrix} 2 & 2 & . & . & . & . & 2 \\ 2 & 4 & . & . & . & . & 4 \\ . & . & 6 & . & . & . & 6 \\ . & . & . & . & & & . \\ . & . & . & & . & & . \\ . & . & . & & & . & . \\ 2 & 4 & 6 & . & . & . & 2m \end{pmatrix}$$

Then $E_2(d \rightsquigarrow^* \perp) = d(N-d)$ with a maximum of $m(m+1) \simeq (N/2)^2$ when $d = m$. Using our general matrix method, we thus retrieve the quadratic complexity result found by Israeli and Jalfon [11].